

# Pick your Poison: Undetectability versus Robustness in Data Poisoning Attacks against Deep Image Classification

Nils Lukas  
 nlukas@uwaterloo.ca  
 University of Waterloo  
 Ontario, Canada

Florian Kerschbaum  
 florian.kerschbaum@uwaterloo.ca  
 University of Waterloo  
 Ontario, Canada

## ABSTRACT

Deep image classification models trained on large amounts of web-scraped data are vulnerable to data poisoning, a mechanism for backdooring models. Even a few poisoned samples seen during training can entirely undermine the model's integrity during inference. While it is known that poisoning more samples enhances an attack's effectiveness and robustness, it is unknown whether poisoning too many samples weakens an attack by making it more detectable. We observe a fundamental detectability/robustness trade-off in data poisoning attacks: Poisoning too few samples renders an attack ineffective and not robust, but poisoning too many samples makes it detectable. This raises the bar for data poisoning attackers who have to balance this trade-off to remain robust and undetectable. Our work proposes two defenses designed to (i) detect and (ii) repair poisoned models as a post-processing step after training using a limited amount of trusted image-label pairs. We show that our defenses mitigate all surveyed attacks and outperform existing defenses using less trusted data to repair a model. Our defense scales to joint vision-language models, such as CLIP, and interestingly, we find that attacks on larger models are more easily detectable but also more robust than those on smaller models. Lastly, we propose two adaptive attacks demonstrating that while our work raises the bar for data poisoning attacks, it cannot mitigate all forms of backdooring.

## CCS CONCEPTS

- Security and privacy → Software and application security;
- Computing methodologies → Machine learning.

## KEYWORDS

deep image classification, data poisoning, robustness, detectability

### ACM Reference Format:

Nils Lukas and Florian Kerschbaum. 2023. Pick your Poison: Undetectability versus Robustness in Data Poisoning Attacks against Deep Image Classification. In *Preprint*. ACM, New York, NY, USA, 18 pages. <https://doi.org/XXXXXX.XXXXXXX>

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

*Preprint*, May 5, 2023

© 2023 Association for Computing Machinery.  
 ACM ISBN 978-1-4503-XXXX-X/18/06...\$15.00  
<https://doi.org/XXXXXX.XXXXXXX>

## 1 INTRODUCTION

The integrity of deep image classification models is essential for their safe and effective use in real-world applications [57]. A model with integrity should make accurate, transparent, and robust decisions that are resistant to manipulation. Training models with integrity is challenging, which limits the applicability of models in domains with impactful decision-making like healthcare [3, 62], credit scoring [5], and access control systems [87].

One form of manipulation that can undermine a model's integrity is *backdooring* - an attack during training enabling an attacker to influence the model's predictions during inference by stamping a secret *trigger* pattern on an image [10]. Backdooring can be accomplished through *data poisoning*, a technique in which the attacker injects a limited number of poisoned samples into the model's training data, thereby creating spurious correlations [64] between the trigger and an attacker-chosen *target* class. Given that modern deep learning models rely on vast amounts of web-scraped data for optimal accuracy [60], poisoning attacks are considered among the most worrisome threats to a model's integrity in practice [40].

**Problem.** Consider a provider who trains a deep image classifier using web-scraped data and deploys their model for a downstream task such as content moderation, like OpenAI with CLIP [60]. The provider, being trustworthy, must balance the trade-off between high model accuracy and integrity but needs to train on large, web-scraped datasets to achieve the former. The threat is an attacker who poisons a small number of samples during training to evade content moderation during inference. With CLIP, it has been demonstrated that backdooring succeeds by poisoning only 400 out of 400 million image-label pairs [60]. The provider needs a method to mitigate data poisoning while training an accurate model.

To counter backdooring, the provider can (i) sanitize the training dataset [4, 8, 70, 74] or (ii) use algorithmic defenses during [15, 16, 31, 43, 46] or after training [44, 46, 79] or during inference [12, 86]. However, none of these methods can withstand all attacks in practice [84]. Existing data sanitation methods have been broken by more covert attacks [14, 38], and existing algorithmic defenses during or after training substantially deteriorate the model's accuracy on clean data [13, 84]. While poisoning attacks have been tested against large models such as CLIP [7], the evaluation of defenses is lagging behind. No existing defense shows effectiveness against all existing data poisoning attacks, which has called into question whether web-scraped training is at all desirable [7].

**Post-Training Defenses.** Existing post-training defenses [45, 46, 79, 85] are designed to (i) detect whether the model contains a backdoor [79] or (ii) disable the backdoor [45, 46, 85] by *repairing* the model using a limited set of trusted (non-poisoned) data. However, these defenses have studied undetectability and robustness

in isolation [44, 46, 79], whereas an attacker wants to design an attack that is both undetectable and robust. We make fundamental observations on the undetectability/robustness trade-off for a data poisoning attack and show that attackers who *over-poison* by injecting too many samples become detectable, whereas attackers who *under-poison* are not robust. Our findings raise the bar for all data poisoning attackers who need precise control over the number of samples injected during training to maximize both goals.

During training, the model learns a lower-dimensional, latent representation of images known as the *latent space*. The latent space encodes salient characteristics relevant to the classification task and can be visualized to interpret the model [61]. While there are methods to enhance latent representations through regularization [18] or latent space alignment [54], no method exists to maximize the *dissimilarity* between two latent spaces to achieve model repair. Our idea is to find a repaired model with latent representations that are orthogonal to the poisoned model’s representations in order to weaken spurious correlations between the trigger and target class using a limited set of clean, trusted data.

**Our Defenses.** We design two defenses for (i) model repair and (ii) backdoored model detection. Our model repair works by maximizing latent dissimilarity, which we optimize using a known regularization method called *Pivotal Tuning* [63]. Pivotal Tuning clones the pre-trained model, referred to as the Pivot and regularizes the poisoned model to (i) maximize the dissimilarity between both latent spaces using a loss function we propose and (ii) minimize the deviation of the repaired model from the Pivot in the parameter space. Since a data poisoning attacker has no control over the model’s parameters, we hypothesize that repaired parameters exist with high proximity to the Pivot’s parameters.

Our backdoor detection method reverse-engineers the secret trigger pattern, similar to Neural Cleanse [79], through iterative optimization and, as a result, outputs anomaly scores for each class. We observe that Neural Cleanse easily gets stuck in local minima due to the model’s vulnerability to universal adversarial examples [53] and has a high chance of failing at faithfully reconstructing the trigger even for patch-based triggers [24] it was designed to defend against. Our method uses the repaired model to guide the optimization and avoids getting stuck in local minima. We demonstrate improved detection rates of our method over Neural Cleanse.

**Results.** Our empirical evaluation shows that Pivotal Tuning is effective and repairs all surveyed backdoored models on CIFAR-10 [39] using only 1% of trusted data and 2.5% on ImageNet [11] with a maximum deterioration in test accuracy of 2%, outperforming all other surveyed post-training defenses. By ablating over the number of poisoned samples, we observe a fundamental trade-off between robustness and undetectability that allows crafting more potent defenses. Interestingly, we find that detection-based defenses such as Neural Cleanse [79] are successful at detecting backdoors in larger models but often fail to reconstruct the exact trigger needed to repair the model. In contrast, on smaller models, Neural Cleanse often fails to detect an attack but can reconstruct the trigger with high fidelity when it is successful (which is useful for repairing the model). We expand our evaluation to state-of-the-art joint vision-language models [60], and demonstrate the effectiveness and data efficiency of our post-training defenses.

Finally, we propose two adaptive attacks that reveal the limitations of our defenses. The first attack, called Parameter-Controlled Backdoor (PCB), assumes the attacker has control over the entire training process and embeds the malicious backdoor functionality into a small subset of the model’s parameters. In the second data poisoning attack, the attacker uses a technique we call *trigger scattering* to maximize robustness (at the expense of higher detectability).

## 1.1 Contributions

Our key contributions can be summarized as follows.

- (1) We raise the bar for all data poisoning attacks by demonstrating an undetectability/robustness trade-off in the number of poisoned samples. Attacks without control over the number of poisoned samples are substantially less threatening.
- (2) We propose a method to classify whether a model contains a backdoor using at most 5% of clean, trusted data.
- (3) We propose a backdoor repair method via Pivotal Tuning [63] and latent space orthogonalization. Our defense is the most data efficient and repairs backdoored models on CIFAR-10 and ImageNet using only 1% and 2.5% of the training data with a deterioration of less than 2% test accuracy.
- (4) We are the first to experiment with post-training backdoor defenses on large pre-trained models such as OpenAI’s CLIP model [60]. Our results indicate that attacks on large models are easier to detect but more robust.
- (5) We propose rigorous, game-based definitions of robustness and detectability for data poisoning.
- (6) We propose two adaptive attacks showing that data poisoning attackers can balance undetectability and robustness using precise control over the number of injected samples.
- (7) We provide source code to reproduce all our experiments, including all hyper-parameters and ablation studies.

## 2 BACKGROUND

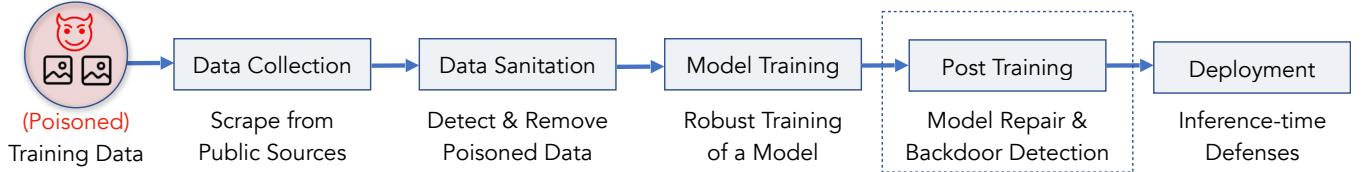
In this section, we review related work on backdooring deep image classifiers. We consider a dataset represented by  $D \subseteq \mathcal{X}$ , where  $\mathcal{X}$  denotes the set of images, and a labeling function  $O : \mathcal{X} \rightarrow \mathcal{Y}$  that maps images to labels in the set  $\mathcal{Y}$ . The goal of the model is to replicate the labeling function for any image dataset accurately.

### 2.1 Data Poisoning Attacks

Data poisoning attacks manipulate a subset of the training data to undermine a model’s integrity during inference and can be divided into two categories: poison label and clean label attacks. Poison label attacks give the attacker limited control over images and the labeling function [24], while clean label attacks assume an attacker can only control the images [69].

**Poison Label.** Badnets [24] uses a visible *trigger* during training and inference and assigns the target class label to all poisoned images. Adaptive Blend (A-Blend) and Adaptive Patch (A-Patch) [58] assign the target label only to a subset of all poisoned images with the goal of increasing the undetectability of their attack.

**Clean Label.** Shafahi et al. [69] propose a clean label attack that stamps poisoned images with an adversarially crafted trigger, assuming access to the attacked model’s parameters. Turner et



**Figure 1: An exemplary training pipeline for deep image classification models from data collection to deployment. Our detection and model repairing defenses are applied post-training.**

al.[76] present an attack that does not require knowledge of the attacked model’s parameters by applying adversarially crafted triggers using a different model trained for the same task. Refool [47] uses naturally occurring trigger patterns, such as reflections, while WaNet [55] uses imperceptible warping of the image. The surveyed clean label attacks differ in only the generation of the trigger pattern but are based on the same principle of poisoning only samples from the target class.

In total, we survey six data poisoning attacks from related work [24, 47, 55, 58, 76] (Qi et al. [58] propose two backdoors). For brevity, we moved detailed descriptions of all surveyed attacks, their hyperparameters, and illustrations of the triggers to Appendix A.2.

## 2.2 Data Poisoning Defenses

Figure 1 illustrates the training pipeline for a deep image classification model using web-scraped data. The model’s training data is scraped from public sources and can contain a small number of poisoned images [6]. A dataset sanitation process aims to filter or augment poisoned images [8, 26, 67] before training the model using a robust training approach [4, 31, 43, 80]. After training, the defender can choose to inspect or repair the model using a limited amount of trustworthy data [44, 46, 79] before deployment when they suspect backdooring. The last line of protection includes defenses implemented during inference, such as anomaly detection systems [77], input purification [12, 52] or certifiable defenses [36, 71, 86, 90].

**Post-Training Defenses.** We summarize four existing post-training defenses for deep image classification models. Weight Decay [48] is a known regularization method that penalizes the model for learning weights with a large magnitude. Fine-Pruning [46] is a defense against backdoors in convolutional neural networks that uses channel-wise pruning of convolutional filters if the absolute activation of the filter on clean data is low. Neural Attention Distillation [45] (NAD) fine-tunes a teacher model using the poisoned model as a starting point and then distills [28] its activations to a student model. Neural Cleanse [79] reconstructs the trigger pattern by backpropagating through the model and optimizing for a mask that behaves like a malicious trigger. The backdoor can be removed by fine-tuning the model on the reconstructed trigger.

## 3 THREAT MODEL

This section describes our threat model and provides rigorous security games for the robustness and detectability of an attack. To this end, we introduce the following functions resembling the training pipeline of a deep classification model depicted in Figure 1.

- $\text{SANITIZE}(D, Y)$ : Returns a filtered and augmented set of images  $\hat{D}$  and labels  $\hat{Y}$  given a set of images  $D$  and labels  $Y$ .
- $\text{TRAIN}(D, Y; \theta_{\text{init}})$ : A stochastic training algorithm that tunes a model on image-label pairs. This function optionally accepts pre-trained weights  $\theta_{\text{init}}$ .
- $\text{POST-TRAINING}(D, Y, \theta)$ : Given a model  $\theta$  and labelled samples, this method returns modified model parameters  $\hat{\theta}$ .
- $\text{DEPLOY}(\theta)$ : Given a model  $\theta$ , this method modifies the inference function of the model.
- $\text{DETECT}(\theta, D, Y)$ : Outputs 0 for backdoored models and 1 for clean models, given a labelled dataset  $D, Y$  and model  $\theta$ .
- $\mathcal{A}(D, O)$ : Given images  $D$  and a ground-truth labeling function  $O$ , this function returns poisoned images and the attack’s target labels for each poisoned image.
- $\text{Acc}(D, Y; \theta)$ : Given a set of image-label pairs  $(D, Y)$  and a labelling function on  $D$  parameterized by  $\theta$ , this function computes the accuracy of the predicted labels against  $Y$ .

Table 1 summarizes our notation.

**Table 1: Summary of Notation**

Notation	Description
$\theta$	Parameters of a deep image classification model
$\mathcal{D}$	A distribution over images
$D \sim \mathcal{D}^n$	Draw $n$ independent images $D$ from $\mathcal{D}$
$O$	A ground-truth labelling function for images
$\mathcal{A}$	A procedure denoting an adversary
$m/n$	Ratio between poisoned and clean samples
$r$	The number of trusted, clean image-label pairs
$\eta$	The Attack Success Rate (ASR)
$y \leftarrow \mathcal{P}(\vec{x})$	Call $\mathcal{P}$ with arguments $\vec{x}$ and assign result to $y$

**Adversary’s Capabilities and Goals.** We consider an attacker capable of poisoning a limited number of  $m$  samples in the defender’s training dataset. The attacker’s goal is to provoke a misclassification of a sample during inference into a pre-defined *target* class by stamping the secret trigger pattern on the image. The adversary succeeds when the model predicts the target class deviating from the ground truth (backdoor functionality) while correctly predicting clean images not containing the trigger (benign functionality). In such a targeted attack, the adversary has the incentive to preserve the model’s accuracy to (i) remain stealthy and (ii) maximize the chance that the defender deploys this backdoored classifier.

**Defender’s Capabilities and Goals.** Our defender controls the entire training pipeline of their model from data collection to deployment as depicted in Figure 1. We also assume the defender has access to a limited set of trustworthy data with ground-truth labels, which is however insufficient to train an accurate model from scratch. The availability of limited trustworthy data is a realistic and often-made [44, 46, 79] assumption in the context of training large machine learning models on web-scraped data where a subset of sources are more trustworthy than others (e.g., by restricting who can post content to these platforms). The defender wants to train models with high (i) accuracy and (ii) integrity. A primary goal of the defender is to (i) repair models suspected of backdooring while limiting the impact on its accuracy and (ii) reliably detect a backdoored model. A secondary goal of the defender is to minimize the amount of trusted data used in the defense.

### 3.1 Robustness

The robustness of a data poisoning attack describes its integrity/accuracy trade-off curve: How much loss in utility does repairing a backdoor incur to the model? In the context of deep image classification, we consider the model’s accuracy on an unseen, clean test set and the model’s integrity as its accuracy on a set of poisoned images with attacker-chosen target labels. A defender can mitigate backdooring by modifying multiple steps of the model training pipeline illustrated in Figure 1, such as the (i) dataset sanitation, (ii) model training, (iii) post-training processes, and (iv) model deployment. Algorithm 1 encodes the data poisoning robustness game for a given data distribution  $\mathcal{D}$ , an attack procedure  $\mathcal{A}$  and some ratio of poisoned and clean samples  $m/n$ .

**Algorithm 1** Data Poisoning Robustness Game

---

```

1: procedure COLLECT( $\mathcal{D}, \mathcal{A}, n, m$ ) ▷ (e.g., web-scraping)
2:    $D_A \sim \mathcal{D}^m$ 
3:    $\tilde{D}_A, \tilde{Y}_A \leftarrow \mathcal{A}(D_A, O)$  ▷ Poisoned samples and target labels
4:    $D \sim \mathcal{D}^n$ 
5:    $\tilde{Y} \leftarrow O(D \cup \tilde{D}_A)$  if clean label else  $O(D) \cup \tilde{Y}_A$ 
6:   return  $D \cup \tilde{D}_A, \tilde{Y}$ 
7: experiment ROBUSTNESS( $\mathcal{D}, \mathcal{A}, n, m$ )
8:    $\tilde{D}_{\text{train}}, \tilde{Y}_{\text{train}} \leftarrow \text{COLLECT}(\mathcal{D}, \mathcal{A}, n, m)$ 
9:    $\hat{D}_{\text{train}}, \hat{Y}_{\text{train}} \leftarrow \text{SANITIZE}(\tilde{D}_{\text{train}}, \tilde{Y}_{\text{train}})$  ▷ Filter/Augment
10:   $\theta_0 \leftarrow \text{TRAIN}(\hat{D}_{\text{train}}, \hat{Y}_{\text{train}})$ 
11:   $D_{\text{trust}} \sim \mathcal{D}^r$  ▷ Limited trustworthy data
12:   $\theta_1 \leftarrow \text{POST-TRAINING}(D_{\text{trust}}, O(D_{\text{trust}}), \theta_0)$  ▷ Model repair
13:   $\theta_2 \leftarrow \text{DEPLOY}(\theta_1)$  ▷ Modify inference algorithm
14:   $D_{\text{test}} \sim \mathcal{D}$ 
15:   $\tilde{D}_{\text{test}}, \tilde{Y}_{\text{test}} \leftarrow \mathcal{A}(D_{\text{test}}, O)$  ▷ Poisoned test set
16:   $A_{\text{CDA}} \leftarrow \text{Acc}(D_{\text{test}}, O(D_{\text{test}}); \theta_2)$  ▷ Clean data accuracy
17:   $A_{\text{ASR}} \leftarrow \text{Acc}(\tilde{D}_{\text{test}}, \tilde{Y}_{\text{test}}; \theta_2) - \text{Acc}(\tilde{D}_{\text{test}}, \tilde{Y}_{\text{test}}; O)$ 
18:  return  $A_{\text{CDA}}, A_{\text{ASR}}$ 

```

---

The robustness game can be described as follows. First, the defender collects web-scraped data by sampling  $n$  clean images and  $m$  poisoned images independently at random (lines 1-4). We assume the defender collects ground-truth labels for all images in the case of a clean label attack; otherwise, the attacker injects their poisoned target labels into the defender’s training dataset (line 5).

After collection, the defender can sanitize the dataset (line 9) and train a model (line 10). Once trained, the defender post-processes the model using a limited number of  $r$  independently sampled, trustworthy data elements with ground-truth labels (lines 11-12) before deploying the model (line 13). Finally, the game evaluates and returns the utility  $A_{\text{CDA}}$  and integrity  $A_{\text{ASR}}$  of the model.

**Defense Effectiveness.** The effectiveness of a defense against a data poisoning attack is the expected trade-off between the attack success rate  $A_{\text{ASR}}$  and the model’s accuracy on clean data  $A_{\text{CDA}}$  when playing the game encoded by Algorithm 1.

$$\text{Succ}_{\text{Repair}} = \mathbb{E}[A_{\text{CDA}} - A_{\text{ASR}}]$$

Consider that a defender can always trivially repair a backdoor in their model by resetting the model weights, which also nullifies the model’s utility. We consider the entire trade-off curve and do not make assumptions for the defender about which point of this trade-off curve is optimal for their use case.

### 3.2 Detectability

The detectability of a data poisoning attack describes the ability of a defender to predict whether a model has been trained on poisoned data. Detectability substantially weakens a data poisoning attack because (i) it allows the defender to reject and re-train a model deemed to contain a backdoor, and (ii) it can help a defender in localizing poisoned samples or repair malicious behavior in the model. The backdoor detection game is formalized by Algorithm 2. Note that Algorithm 2 considers post-training detectability of a data poisoning attack which only has access to the pre-trained model weights and a limited amount of trustworthy data (but not the model’s training data).

**Algorithm 2** Data Poisoning Detectability Game

---

```

1: experiment DETECTABILITY( $\mathcal{D}, \theta_{\text{init}}, n, m, r$ )
2:    $\tilde{D}_{\text{train}}, \tilde{Y}_{\text{train}} \leftarrow \text{COLLECT}(\mathcal{D}, \mathcal{A}, n, m)$  ▷ Poisoned Data
3:    $D_{\text{train}}, Y_{\text{train}} \leftarrow \text{COLLECT}(\mathcal{D}, \mathcal{A}, n + m, 0)$  ▷ Clean Data
4:    $\theta_0 \leftarrow \text{TRAIN}(\tilde{D}_{\text{train}}, \tilde{Y}_{\text{train}}; \theta_{\text{init}})$ 
5:    $\theta_1 \leftarrow \text{TRAIN}(D_{\text{train}}, Y_{\text{train}}; \theta_{\text{init}})$ 
6:    $b \sim \{0, 1\}$  ▷ Unbiased coin flip
7:    $D_{\text{trust}} \sim \mathcal{D}^r$ 
8:    $p \leftarrow \text{DETECT}(D_{\text{trust}}, O(D_{\text{trust}}), \theta_b)$ 
9:    $A_{\text{detect}} \leftarrow 1$  if  $p = b$  else 0
10:  return  $A_{\text{detect}}$ 

```

---

The data poisoning detectability game can be described as follows. First, we collect a set of poisoned images with poisoned labels (line 2) and a set of clean images with ground-truth labels (line 3). Then, we train a poisoned and a clean model  $\theta_0, \theta_1$  (lines 4-5), flip an unbiased coin  $b$  (line 6), and instantiate the defender’s detection algorithm given  $r$  clean samples with ground-truth labels (lines 7-8). We measure and return the correctness of the prediction as  $A_{\text{detect}}$  (lines 9-10). The success of the detectability game is its expected classification accuracy  $\mathbb{E}[A_{\text{detect}}]$  when repeating the game.

## 4 CONCEPTUAL APPROACH

This section first describes our Soft Latent Orthogonalization Loss (SLOL) to regularize a model’s latent space and then presents a

method based on Pivotal Tuning [63] to optimize a regularization constraint with minimal loss in model accuracy.

#### 4.1 Model Repair through the Latent Space

**Properties of a Latent Space.** During training, deep image classifiers learn a low-dimensional latent space from high-dimensional images that capture salient image features relevant for the classification task [37]. Latent representations (or short *latents*) have many useful properties, such as the ability to measure perceptual image similarity [89] or guiding image generation models [61]. We focus on two properties of the latent space that can be summarized as follows.

- (1) Images from similar classes have similar latents [89].
- (2) The model learns *latent directions* that control one or more salient features. For example, CLIP [60] learned a latent direction to interpolate the ‘age’ attribute for any image (e.g., by turning a lion cub into an adult lion) [61].

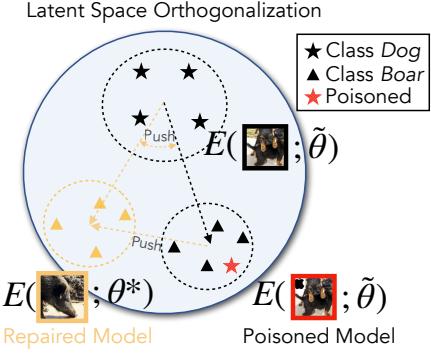
It has been observed that images containing a backdoor trigger form a dense cluster in the latent space, which has been used to detect poisoned samples [8]. However, no defense exists that considers latent directions learned by the model to defend against backdoors. We hypothesize that we can repair the model by enforcing a regularization constraint on the model’s latent space by modifying its latents and latent directions.

**Objective 1 - Parameter Space.** Our objective is to find a repaired model,  $\theta^*$ , given a (possibly) poisoned model,  $\tilde{\theta}$ , and limited trusted data,  $D_{\text{trust}}$ . The repaired model should retain its functionality on clean data while behaving differently on poisoned data. We assume that since a data poisoning attacker cannot control the poisoned model’s parameters,  $\tilde{\theta}$ , there likely exists a set of repaired parameters,  $\theta^*$ , for a repaired model that is close to the poisoned model’s parameters, such that  $\|\theta^* - \tilde{\theta}\|_2 < \epsilon$  for a small positive  $\epsilon$ . To preserve functionality on clean data, our first objective is to minimize the perturbation in the parameter space applied to the repaired model.

**Objective 2 - Latent Space.** Our second objective is to regularize the model’s latent space to disable functionality not present in the trusted data. Regularizing the latent space is a known and effective method to improve a model’s latent representations [18]. The idea is to maximize the dissimilarity between the latent spaces of the (poisoned) model before tuning and the repaired model. Since studies have shown that large models such as CLIP encode meaningful functionalities through latent directions [19], we impose two regularization constraints on the latent space.

- (1) We orthogonalize the clusters of the latent space within the repaired model by using a pairwise cosine similarity loss.
- (2) We orthogonalize latent directions across both models. More precisely, given two centroids in the latent space of the poisoned model  $\vec{Z}_1, \vec{Z}_2$ , and the same two centroids (derived using the same images) in the repaired model  $\hat{Z}_1, \hat{Z}_2$ , we (i) first compute the pairwise latent directions  $\vec{A} = \vec{Z}_1 - \vec{Z}_2$  and  $\hat{A} = \hat{Z}_1 - \hat{Z}_2$  and then (ii) assign a high loss if they are non-orthogonal by measuring their cosine similarity.

Figure 2 conceptually illustrates our two constraints for the latent space of a poisoned model with parameters  $\tilde{\theta}$ . We initialize the



**Figure 2: A conceptual illustration of our defense in the latent space. Black and yellow colors refer to the model before and after repair. The repaired model’s latent cluster for the class *Boar* is orthogonalized and pushed away from its origin.**

repaired model’s parameters  $\theta^*$  by cloning  $\tilde{\theta}$  and regularize its latent space to find new latents for each class that are orthogonal to the latents from the poisoned model.

**SOL.** We refer to this latent regularization method as the *Soft Latent Orthogonalization Loss* (SOL) which is encoded by Algorithm 3. Algorithm 3 defines the following three procedures. The procedure **CENTROID** groups images by their ground-truth label and computes their latents, denoted by  $E(\cdot; \theta)$  (lines 2-3). Then, it returns the centroids for each class (line 4). **ORTHOGONALITY** computes the aforementioned regularization over all class pairs (lines 7-8) by orthogonalizing latent directions across both models (line 10) and orthogonalizing the centroids within the repaired model (line 11). Finally, SOL takes two models  $\tilde{\theta}, \theta^*$  and trustworthy image-label pairs and returns the orthogonality loss (lines 14-16). We include PyTorch code outlining the matrix operations needed to compute both terms in the **ORTHOGONALITY** procedure efficiently.

---

#### Algorithm 3 Soft Latent Orthogonalization Loss / Model Repair

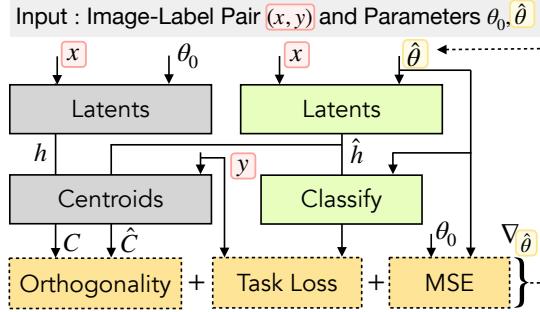
---

```

1: procedure CENTROIDS( $\theta, D, O$ )
2:   for  $i \leftarrow 1$  to  $|Y|$  do
3:      $H_i \leftarrow \{E(x; \theta) | x \in D \text{ and } O(x) = i\}$   $\triangleright$  Group Latents
4:   return  $\{\frac{1}{|H_i|} \sum_{h \in H_i} h | i \in 1$  to  $|Y|\}$   $\triangleright$  Class Centroids
5: procedure ORTHOGONALITY( $C, \hat{C}$ )
6:    $l \leftarrow 0$ 
7:   for  $i \leftarrow 1$  to  $|Y|$  do
8:     for  $j \leftarrow 1$  to  $|Y|$  do
9:       if  $i < j$  then
10:          $l \leftarrow l + \cos(C_i - C_j, \hat{C}_i - \hat{C}_j)$ 
11:          $l \leftarrow l + \cos(\hat{C}_i, \hat{C}_j)$ 
12:   return  $l$ 
13: procedure SOL( $\tilde{\theta}, \theta^*, D, O$ )
14:    $C \leftarrow \text{CENTROIDS}(\tilde{\theta}, D, O)$   $\triangleright$  Frozen model
15:    $\hat{C} \leftarrow \text{CENTROIDS}(\theta^*, D, O)$   $\triangleright$  Trainable model
16:   return ORTHOGONALITY( $C, \hat{C}$ )

```

---



**Figure 3: An overview of our Pivotal Tuning method for model repair. See Algorithm 3 for a definition of the orthogonality loss. Grey, green, and yellow boxes, respectively, indicate frozen modules, tunable modules, and loss terms.**

**Pivotal Tuning.** We optimize both objectives (in the parameter and latent space) by using a known regularization method for pre-trained models called Pivotal Tuning [63]. Pivotal Tuning has been shown to enforce a regularization constraint in generative models while limiting the deterioration of the model’s utility [50]. The frozen model before tuning is also referred to as the *Pivot* around which the tunable model is optimized. Figure 3 illustrates the optimization procedure given the Pivot  $\tilde{\theta}$ , the tunable model  $\theta^*$  and one image-label pair  $(x, y) \in D$ . We compute the SLOL loss, a task-specific loss (we use the cross-entropy loss for image classification), and the distance in the parameter space between the frozen pivot and the tunable model. The combined loss can be written as follows for a set of frozen parameters  $\tilde{\theta}$  (the Pivot), a set of tunable parameters  $\theta^*$ , and a set of trusted images  $D$ .

$$\mathcal{L} = \text{CE}(M(E(D; \theta^*); \theta^*) + \lambda_S \text{SLOL}(\tilde{\theta}, \theta^*, D, O(D)) + \lambda_P \|\tilde{\theta} - \theta^*\|_2$$

We refer to Appendix A.5 for the hyper-parameters  $\lambda_S, \lambda_P$  that we used for our defense in the evaluation.

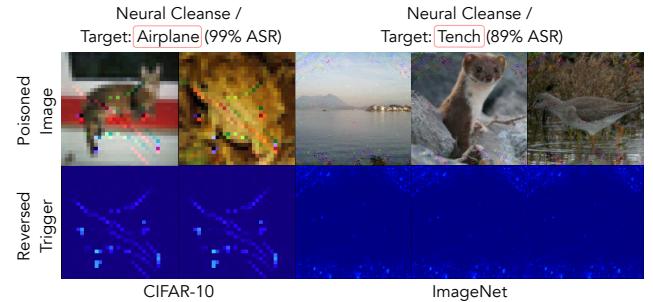
**Contrasting with Existing Work.** We highlight that our work is different from previous latent-space repair methods because our method maximizes the *dissimilarity* of the latent space before and after repairing the model. Existing work, specifically Neural Attention Distillation (NAD) [44], is based on the idea of maximizing *similarity* between a student and a (potentially poisoned) teacher. They create a teacher model by cloning the poisoned model and fine-tuning it for a limited number of epochs on trustworthy data. Then, they fine-tune the poisoned model with a regularization loss to align (i.e., maximize the similarity) the teacher’s and student’s latent representations. However, it has been shown that a teacher containing a backdoor can transfer that backdoor to the student through its latent representations [21, 65]. In summary, while NAD maximizes similarity in the latent space and minimizes similarity in the parameter space, we do the opposite.

## 4.2 Backdoor Model Detection

Neural Cleanse [79] is a method to detect backdoors in deep image classification models. This method works by attempting to reverse-engineer a trigger pattern and calculating an anomaly score for each class based on the trigger’s norm. The trigger, represented

as  $T^*$ , is added to an image using an operation such as element-wise addition, represented by  $\oplus$ . The goal of Neural Cleanse is to update the trigger through an optimization loop and make the model misclassify any image into a target class when adding a trigger. A high anomaly score indicates that the reverse-engineered trigger has a low L1 norm. To determine whether an anomaly is detected, the authors suggest using a constant value to threshold the trigger’s L1-norm. We refer to Algorithm 6 in the Appendix for an implementation of Neural Cleanse using our notations.

**Issues with Neural Cleanse.** Using the L1-norm as a measure of anomaly can be exploited by attackers who design triggers with L1-norms above the threshold. For instance, natural reflections [47] create triggers with low perceptual distance from the original image but a large L1-norm distance, allowing the attack to remain undetected even when the trigger is successfully reconstructed. Another issue is that Neural Cleanse can get stuck in local minima, rendering it unable to detect patch-based backdoor attacks [24] it was designed to defend against [84]. Interestingly, this can be linked to the vulnerability of deep models to universal adversarial perturbations [53], a type of adversarial attack optimizing the same loss as Neural Cleanse. Figure 4 illustrates this failure case for Neural Cleanse on CIFAR-10 and ImageNet, where it is able to find a trigger with a high success rate for any class, including non-backdoored ones.



**Figure 4: Issues with Neural Cleanse related to universal adversarial perturbations [53] where a reverse-engineered trigger achieves near-perfect success for a non-poisoned class.**

**Adding Calibration.** We propose a simple improvement over Neural Cleanse to remedy both aforementioned issues. Our idea is first to repair a model, for instance, using Pivotal Tuning presented in the previous section, and then use the repaired model to guide the optimization procedure used to reverse-engineer a trigger. We compute the anomaly score as the difference in the success rates of the reverse-engineered trigger causing a misclassification in the model before repair and preserving the correct classification in the model after repair. Since the models before and after repair are highly similar, we expect that they are vulnerable to many similar universal adversarial perturbations, decreasing the probability that the optimization gets stuck at a local minimum. Furthermore, we believe that representing an anomaly score as a difference in success rates is more readily interpretable for practitioners and makes it easier to choose a threshold for flagging anomalies.

Algorithm 4 implements our Calibrated Trigger Inversion method to detect backdoored models. First, we repair the model (line 2), and

**Algorithm 4** Calibrated Trigger Inversion / Backdoor Detection

---

```

1: procedure CNC( $\tilde{\theta}$ ,  $D_{\text{trust}}$ ,  $Y_{\text{trust}}$ ,  $D_{\text{test}}$ ,  $N$ ,  $\alpha$ )
2:    $\theta^* \leftarrow \text{PIVOTALTUNING}(\tilde{\theta}, D_{\text{trust}}, Y_{\text{trust}})$   $\triangleright \text{Model Repair}$ 
3:    $S \leftarrow \{\}$ 
4:   for  $c \leftarrow 1$  to  $|\mathcal{Y}|$  do
5:      $T^* \leftarrow \text{Random Initialization}$   $\triangleright \text{Shape of an image}$ 
6:     for  $1$  to  $N$  do
7:        $(x, y) \sim D_{\text{trust}} \times Y_{\text{trust}}$   $\triangleright \text{Sample image-label pair}$ 
8:        $\tilde{y} \leftarrow M(E(x \oplus T^*; \tilde{\theta}); \tilde{\theta})$   $\triangleright \text{Pivotal model}$ 
9:        $y^* \leftarrow M(E(x \oplus T^*; \theta^*); \theta^*)$   $\triangleright \text{Repaired model}$ 
10:       $g_T \leftarrow \nabla_{T^*} (\text{CE}(\tilde{y}, c) + \text{CE}(y^*, y))$ 
11:       $T^* \leftarrow T^* - \alpha \cdot \text{Adam}(T^*, g_T)$   $\triangleright \text{Update Trigger}$ 
12:       $D_{\text{psn}} \leftarrow \{x \oplus T^* | x \in D_{\text{test}}\}$ 
13:       $S \leftarrow S \cup \{\text{Acc}(D_{\text{psn}}, c; \tilde{\theta}) - \text{Acc}(D_{\text{psn}}, Y; \theta^*)\}$ 
14:   return  $\max(S)$   $\triangleright \text{Return highest anomaly score}$ 

```

---

then, for each class (line 4), we randomly initialize a trigger in the shape of an image (line 5). We optimize the trigger for  $N$  steps by sampling an image-label pair (line 7) and computing the predictions from the model before (the Pivot) and after repair (lines 8-9). Then we compute the gradient with respect to the trigger to optimize for misclassification in the Pivot and a correct classification in the repaired model and update the trigger using this gradient (lines 10-11). After  $N$  steps, we compute the difference in success rate as the anomaly score (lines 12-13) and return the largest anomaly score to predict backdoored models (line 14).

### 4.3 Adaptive Attacks

We design two adaptive attacks against our defenses called the (i) Trigger-Scattering Backdoor (TSB) and (ii) Parameter-Controlled Backdoor (PCB). Both attacks are designed with the goal of maximizing robustness while remaining undetectable.

**Trigger-Scattering Backdoor (TSB).** Our first adaptive attack uses a technique we refer to as *trigger scattering*. The technique involves breaking a single, large trigger into multiple smaller segments and then poisoning each image with only one of these segments during training but utilizing multiple segments during inference. The trigger segments should be pairwise, perceptually distinct to make the model learn a separate filter for each segment. This technique does not require the model to learn sample-specific triggers [42] and is different from other work [47, 58, 76] in that, we require the model to recognize each individual segment by itself as opposed to showing a large, complex trigger pattern. For brevity, we move Algorithm 7 implementing TSB into the Appendix.

**Parameter-Controlled Backdoor (PCB):** This attack assumes an elevated threat model compared to data poisoning. Here, the attacker has control over the entire training process, not just injecting poisoned samples, which allows them to make arbitrary modifications to any model parameter. These attacks are sometimes referred to as *code poisoning* [88] or, in certain cases, supply-chain attacks [29]. We investigate this elevated threat model with the goal of evaluating whether an attacker who controls the training process can execute more powerful attacks to evade our defenses.

**Algorithm 5** Parameter-Controlled Backdoor / Adaptive Attacks

---

```

1: procedure PCB( $\theta$ ,  $D_{\text{train}}$ ,  $Y_{\text{train}}$ ,  $y_{\text{target}}$ ,  $p$ ,  $N$ )  $\triangleright \text{Code Posioning}$ 
2:    $\tilde{\theta} \leftarrow \text{SAMPLEPARAMS}(\theta, p)$   $\triangleright \text{Randomly sample parameters}$ 
3:    $T \leftarrow \text{Random Initialization}$ 
4:   for  $i \leftarrow 1$  to  $N$  do
5:      $(x, y) \sim D_{\text{train}} \times Y_{\text{train}}$ 
6:      $\theta^* \leftarrow \tilde{\theta}$  if  $i$  is even else  $\theta$ 
7:      $\hat{x} \leftarrow (x \oplus T)$  if  $i$  is even else  $x$   $\triangleright \text{Poison if } i \text{ is even}$ 
8:      $\hat{y} \leftarrow y_{\text{target}}$  if  $i$  is even else  $y$ 
9:      $y_{\text{pred}} \leftarrow M(E(\hat{x}; \theta); \theta)$   $\triangleright \text{Classify image}$ 
10:     $g \leftarrow \nabla_{\theta^*} \text{CE}(y_{\text{pred}}, \hat{y})$   $\triangleright \text{Gradients w.r.t parameters}$ 
11:     $\theta^* \leftarrow \theta^* - \alpha \cdot \text{Adam}(\theta^*, g)$ 
12:   return  $\theta$   $\triangleright \text{Return poisoned model}$ 

```

---

The core idea behind PCB is to embed malicious backdoor functionality only within a smaller, randomly sampled subset of the poisoned model’s parameters. PCB exploits the first assumption made by our Pivotal Tuning-based defense that there exists a set of parameters for a repaired model with a low distance to the poisoned model’s parameters. By embedding malicious functionality only in a subset of parameters, we expect that this results in a stronger perturbation to these parameters (because fewer parameters need to encode the same malicious functionality). To repair such a backdoor, our Pivotal Tuning-based defense would have to target specifically these parameters and apply stronger perturbations (on average) to them than to other parameters.

Algorithm 5 encodes our PCB attack. The input is a set of model parameters  $\theta$  (can be randomly initialized), image-label pairs  $D_{\text{train}}$ ,  $Y_{\text{train}}$ , a target class  $y_{\text{target}} \in \mathcal{Y}$ , a fraction for the sampled parameters  $p$  and the number of optimization steps  $N$ . First, we randomly sample a fraction of  $p$  poisoned parameters from the model (line 2) and a trigger pattern (line 3). Then, we use alternating optimization on the model’s parameters. In a for-loop (line 4), we update the poisoned parameters on poisoned data when the iteration counter is even, and otherwise, we update all model parameters (including the poisoned ones) on clean, non-poisoned data (lines 5-9). Finally, we return the poisoned model’s parameters.

## 5 EXPERIMENTS

We begin by describing the experimental setup, including datasets, model architectures we use to conduct our experiments, and measured quantities. Then we present results on the detectability of backdoored models by (i) showing how Neural Cleanse [79] compares with our Calibrated Trigger Inversion and (ii) demonstrating a detectability/attack effectiveness trade-off. Then we show results on the robustness of existing attacks by (i) contrasting our defense with other defenses and (ii) demonstrating a robustness/attack effectiveness trade-off. Finally, we evaluate two adaptive data poisoning attacks against our defenses.

### 5.1 Experimental Setup

**Datasets.** We experiment with two image classification datasets: CIFAR-10 [39] and ImageNet [11]. CIFAR-10 consists of 50k images and 10 class labels with a resolution of  $32^2$  pixels. ImageNet consists

of 1.23m images and 1k class labels, which we resize and then center-crop to a resolution of 224<sup>2</sup> pixels.

**Model Architectures.** We experiment with two model architectures: ResNet-18 [27] and a pre-trained CLIP [60] model checkpoint released by OpenAI. We train all models from scratch on CIFAR-10. Due to the high computational demands for re-training even the smallest versions of CLIP (tens of hours on hundreds of GPUs [33]) from scratch, we instead re-use pre-trained checkpoints and perform backdooring by fine-tuning the models on poisoned data. Torchvision provides checkpoints for ResNet-18<sup>1</sup> and Huggingface provides checkpoints for CLIP<sup>2</sup>. All models achieve a high test accuracy: 95% on CIFAR-10, 70% on ImageNet with a ResNet-18, and 73% test accuracy by CLIP.

**Implementation.** We re-implement all surveyed attacks and defenses from scratch in PyTorch 1.13 and make our framework and pre-trained, backdoored models publicly available to foster reproducibility. All experiments are evaluated on A100 GPUs with 80Gbyte VRAM on a server with 1TByte of RAM.

## 5.2 Measured Quantities.

Like related work [45, 79], we compute the following metrics.

- **Clean Data Accuracy (CDA):** The CDA measures a model’s utility in terms of its accuracy on clean, unseen data.
- **Attack Success Rate (ASR):** The ASR measures the model’s integrity in terms of its accuracy on poisoned data with attacker-chosen labels. Models with integrity have low ASR.

**Attack Quantities.** We measure the following quantities of a data poisoning attack in relation to the number of poisoned samples an attacker injected into the poisoned model’s training data.

- **Attack Effectiveness:** To measure the effectiveness of an attack, denoted by  $\eta_m$ , we estimate the expected ASR  $\eta$  when the attacker can poison at most  $m$  samples.
- **Detectability:** For detectability, denoted by  $\rho_m$ , we calculate the expected ROC AUC  $\rho$  of predicting backdoored/non-backdoored models from a balanced set when the attacker can poison at most  $m$  samples.
- **Robustness:** We measure the robustness  $\eta$  of an attack-defense pair by its ASR after using a defense.

**Defense Quantities.** We study a defender who is limited in the availability of trustworthy data because – as outlined in Section 3, a secondary goal of the defender is to minimize the use of this data. We measure effectiveness and efficiency for a defender as follows.

- **Defense Effectiveness:** We measure the effectiveness  $\eta$  of an attack-defense pair by the lowest expected ASR when limiting the maximum allowed degradation of the model’s accuracy to  $\Delta$ . Throughout our paper, we treat effectiveness as a utility/integrity trade-off and refer to a model as *repaired* when  $\eta \leq 5\%$ , likely preventing an attacker from executing their attack successfully in practice.
- **Data Efficiency:** The efficiency of a defense against an attack, denoted  $v_m$  is the attack’s robustness  $\eta$  when limiting the defender’s data to at most  $m$  clean, labeled images.

<sup>1</sup><https://pytorch.org/vision/stable/models.html>

<sup>2</sup><https://huggingface.co/openai/clip-vit-base-patch32>

Throughout our paper, we limit the maximum acceptable degradation in the CDA that a defense incurs to  $\Delta = 2\%$ . This means that if the model has an accuracy of 95% of CIFAR-10, we terminate the defense if it deteriorates the model’s test accuracy below 93%.

## 5.3 Hyper-Parameter Optimization

To ensure a fair comparison between all defenses, we perform a systematic grid search to determine optimal hyper-parameters for all defenses, simulating how defenders would identify them in practice. To this end, we use the limited trusted data available to the defender to poison their own model with a **BadNets** [24] attack. By knowing the trigger, the defenders can measure ASR/CDA pairs while using their defense. We define a space of valid hyper-parameters, run a grid search and select hyper-parameters yielding the highest defense effectiveness (i.e., lowest remaining ASR  $\eta$  with  $\Delta = 2\%$ ). We run a separate grid search for each (i) dataset, (ii) model architecture, and (iii) number of trusted data  $r$  available to the defender. From this point forward, we fix the same optimal defense hyper-parameters for every experiment. Appendix A.5 provides detailed descriptions of all hyper-parameters and their configurations. To the best of our knowledge, our approach is the first to systematically determine defense hyper-parameters for evaluating the robustness and detectability of data poisoning attacks.

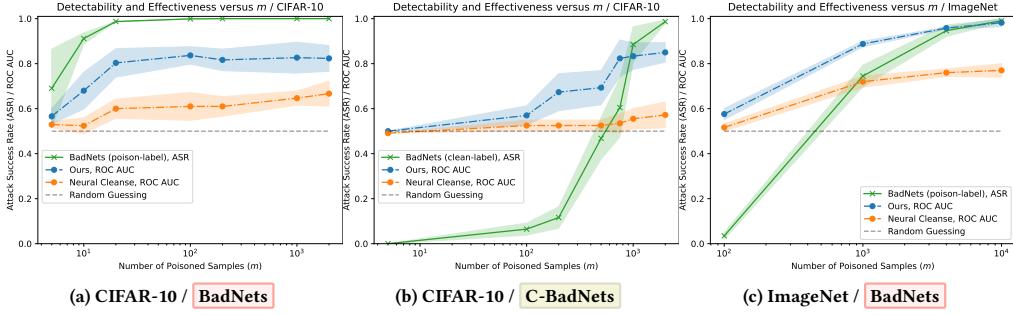
## 5.4 Detectability

We begin by studying the detectability of data poisoning attacks relative to the number of injected, poisoned samples. Our hypothesis is that attackers who increase their attack effectiveness by over-poisoning also increase the detectability of their attack. To study detectability, we compare Neural Cleanse [79] and our Calibrated Trigger Inversion (see Algorithm 4) against a clean label and poison label attack. For brevity, we moved Algorithm 6 describing Neural Cleanse into the Appendix.

**5.4.1 Ablation Study for Patch-based Attacks.** We first study the detectability of patch-based poisoning attacks, which Neural Cleanse and our method were designed to protect against. Patch-based attacks stamp a trigger within a restricted region of the image, such as a little square on the image’s corner [24]. We create a variant of **BadNets** that only stamps the trigger on samples from the target class, meaning that the label remains unmodified, and refer to this attack as **C-BadNets**. Both attack variants use the same trigger.

**Setup.** Our goal is to ablate over the number of poisoned samples  $m$  to evaluate detectability. For each  $m$ , we train 20 separate models on clean and poisoned data for CIFAR-10 and 3 models for ImageNet. We train fewer models on ImageNet due to the higher computational intensity for model training. Since clean label attacks are less effective [68] and each ImageNet class is limited to about 1k samples, we *boost* poisoned samples with a factor  $b$  by artificially repeating them  $b$  times during training. We only have to boost on ImageNet due to the low effectiveness of all clean label attacks in order to meaningfully study the robustness/detectability trade-off. The trigger is a patch of a downscaled apple (🍎) covering less than 1% of the image’s total area (e.g., 3<sup>2</sup> pixels for CIFAR-10).

**Result Overview.** Figure 5 shows the number of poisoned samples ( $m$ ) on the x-axis and the attack success rate (ASR) and detectability in ROC AUC on the y-axis. The green, solid line shows



**Figure 5: The effectiveness and detectability of a poison label and clean label attack on CIFAR-10 and ImageNet.** The shaded area denotes 1 standard deviation. We train 10 separate, poisoned models for each  $m$  to measure the attack’s detectability in ROC AUC and repeat the experiment three times. On ImageNet, due to high computational intensity, we only train three models and repeat the experiment three times. C-BadNets is a clean label variant of the BadNets attack.

the attack’s ASR, whereas the blue and orange dotted lines show the detectability of each attack using (i) our Calibrated Trigger Inversion and (ii) Neural Cleanse [79]. We evaluate detectability against a ResNet-18 model [27]. All plots show that more effective attacks are more easily detectable and that our Calibrated Trigger Inversion strictly outperforms Neural Cleanse in the ability to detect backdoors for every  $m$ . We observe that Neural Cleanse performs substantially worse than our approach because it often reverse-engineers a generic trigger that barely resembles the real trigger. Since many unrelated trigger patterns can already perfectly satisfy the optimization criterion (to achieve a high ASR), Neural Cleanse often gets stuck in local minima. Our detection strictly improves over Neural Cleanse, and thus when referring to detectability, we only refer to an attack’s detectability using our method.

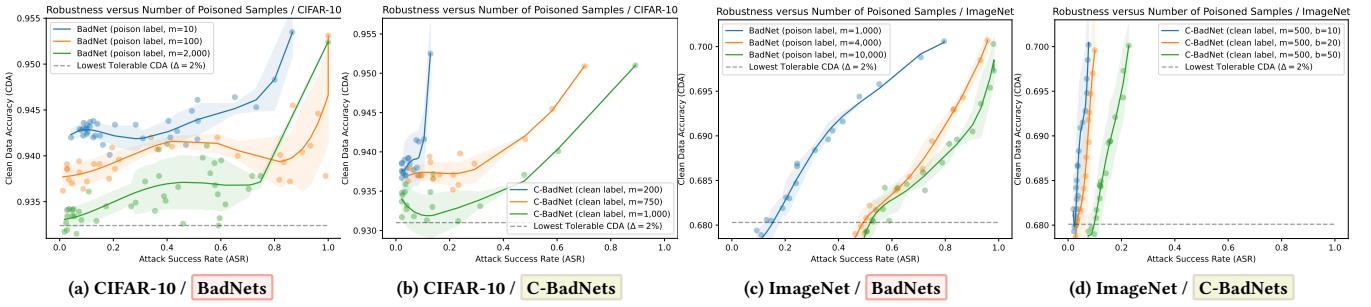
**CIFAR-10.** Figure 5a shows that BadNets achieves a high ASR of  $\eta_5 = 0.69$  with a low detectability  $\rho_5 = 0.56$  on CIFAR-10. As the attacker poisons more samples, the ASR increases to a near-perfect value  $\eta_{20} = 0.99$ , but also the attack’s detectability  $\rho_{20} = 0.80$  increases. Interestingly, the attack’s detectability does not substantially increase once it achieves a near-perfect ASR. We observe the same phenomenon for C-BadNets, which requires injecting more poisoned samples to achieve a similar effectiveness as its poison label counterpart. When equalizing for effectiveness, we observe a slightly higher detectability for clean label attacks than for poison label attacks. For instance, in Figure 5b, we measure an effectiveness/detectability pair of  $\eta_{750} = 0.60/\rho_{750} = 0.81$  for C-BadNets, but only  $\eta_5 = 0.69/\rho_5 = 0.56$  for BadNets on CIFAR-10.

**ImageNet.** Figure 5c shows the effectiveness and detectability of BadNets on ImageNet. For BadNets, we measure an effectiveness  $\eta_{4000} = 0.93$  and a detectability of  $\rho_{4000} = 0.96$  (i.e., a near-perfect attack detectability). Even for relatively low effectiveness  $\eta_{100} = 0.03$ , we observe a detectability of  $\rho_{100} = 0.57$ . Recall that on CIFAR-10, we measure a similar detectability of  $\rho_5 = 0.56$  - but at a much higher attack effectiveness of  $\eta_5 = 0.69$ . We conclude that the same attacks are substantially easier to detect on ImageNet than on CIFAR-10 when equalizing for effectiveness.

$\mathcal{D}$	Attack	$m$	$b$	CDA	ASR	Detectability
CIFAR-10 [39]	No Backdoor	0	0	0.953	0.00	0.50
	BadNets [24]	50	1	0.952	1.00	0.76
	A-Blend [58]	50	1	0.952	0.94	0.65
	A-Patch [58]	50	1	0.951	1.00	0.73
	Adv [76]	500	1	0.947	0.98	0.68
	Refool [47]	500	1	0.954	0.38	0.70
	WaNet [55]	500	1	0.954	0.29	0.65
ImageNet [11]	No Backdoor	0	0	0.702	0.00	0.50
	BadNets [24]	4000	1	0.702	0.97	0.97
	A-Blend [58]	4000	1	0.702	0.56	0.95
	A-Patch [58]	4000	1	0.701	0.75	1.00
	Adv [76]	500	50	0.705	0.46	0.55
	Refool [47]	500	50	0.694	0.74	1.00
	WaNet [55]	500	50	0.699	0.20	0.63

**Table 2: Listed are all surveyed poison label and clean label attacks, the number of poisoned samples  $m$ , the boosting factor  $b$ , the CDA/ASR pair, and the detectability in ROC AUC measured using our Calibrated Trigger Inversion method.**

**5.4.2 Detectability of Complex Triggers.** Table 2 shows the clean data accuracy (CDA), effectiveness, and detectability on CIFAR-10 and ImageNet for all six surveyed data poisoning attacks. As described in 2.1, these attacks differ from BadNets and C-BadNets in the complexity of the trigger. By complexity, we refer to the function used to apply the trigger (e.g., computing reflections [47] or warping [55]). Since both Neural Cleanse and our Calibrated Trigger Inversion search for a constant trigger that, when added to any image, produces a misclassification, we expect that complex triggers evade detection. Surprisingly, this is not the case against many complex triggers, and our method yields high detection rates against these types of attacks.



**Figure 6: The integrity/utility trade-offs on CIFAR-10 [39] and ImageNet [11] against a **BadNets** [24] attacker who injected a varying number of poisoned samples. We record measurements using Pivotal Tuning as a model repair method with  $r = 1\%$  clean data while measuring the ASR/CDA pair every fixed number of iterations. For ImageNet, we *boost* the number of poisoned samples by repeating them  $b$  times to obtain a higher attack effectiveness. The shaded region represents one standard deviation.**

From Table 2, we see that all poison label attacks achieve a high, near-perfect ASR on CIFAR-10, whereas the clean label attacks achieve a much lower ASR, despite poisoning 10× more samples. A notable exception among the clean label attacks is the **Adv** [76] attack, which uses adversarially crafted triggers and achieves an ASR of 0.96 on CIFAR-10. Our findings corroborate with findings by Fowl et al. [17] that adversarial examples [23] are effective triggers to use in data poisoning. We find that our Calibrated Trigger Inversion method has improved detection rates against all attacks on CIFAR-10 of at least 0.65. This detectability is even higher on ImageNet, where we observe a near-perfect detectability against all attacks except against **WaNet** [55] and **Adv** [76].

The lower detectability against **WaNet** can be explained by its relatively low effectiveness of only 0.20. In contrast, **Adv** achieves a much higher ASR of 0.46 but has a substantially lower detectability of only 0.55. We conclude that even complex triggers such as natural reflections [47] or warping [55] are detectable by approaches that reverse-engineer a constant trigger. We observe this is because images from the poisoned class are more sensitive to perturbations, enabling reverse-engineered triggers with a high success rate.

## 5.5 Robustness

We evaluate the data efficiency of the five surveyed model repair defenses by limiting the number of trusted image-label pairs available to the defender to  $r \in \{1\%, 2.5\%, 5\%\}$  relative to the training dataset. We study the robustness of the same poisoned model checkpoints used to evaluate detectability in Table 2. Since every surveyed post-training defense involves an iterative fine-tuning procedure, we record the ASR/CDA pairs every 50<sup>th</sup> and 200<sup>th</sup> iteration on CIFAR-10 and ImageNet to plot the trade-off curves between the model’s integrity and its test accuracy. We terminate a defense if the CDA deteriorates by more than  $\Delta = 2\%$ .

**5.5.1 Ablation Study.** Figure 6 shows the integrity/accuracy curves on CIFAR-10 and ImageNet for the **BadNets** and **C-BadNets** attacks introduced in Section 5.4. The x- and y-axes show the ASR and CDA. We show the results of our Pivotal Tuning-based defense

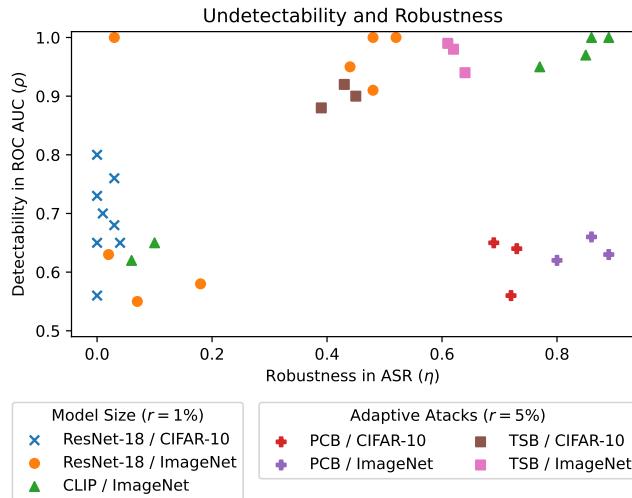
against three attacks that vary only in the number of poisoned samples  $m$ , and the shaded region represents one standard deviation.

**CIFAR-10.** Figure 6a shows that an **BadNets** attacker who injected more poisoned samples will enhance their attack’s robustness. This means that an attacker who injected  $m = 2\,000$  instead of only 100 samples can expect a higher remaining ASR in the repaired model for the same tolerable deterioration of CDA chosen by the defender. For instance, Figure 6a shows that repairing a model that has been poisoned with only 10 samples can be repaired with a drop of only about 1% CDA, whereas a model poisoned with 2000 samples experience a deterioration in CDA of nearly 2%. We observe a similar phenomenon for **C-BadNets** in Figure 6b, where attackers can increase their attack’s robustness by over-poisoning.

**ImageNet.** Figures 6c and 6d show the robustness of attacks on ImageNet using  $m \in \{1\,000, 2\,000, 10\,000\}$  when the defender is limited to  $r = 1\%$  trusted image-label pairs. As for CIFAR-10, we observe that over-poisoning increases the attack’s robustness. Interestingly, Figure 6d shows that the defender fails to repair the **C-BadNets** attack using the most poisoned samples ( $m = 500, b = 50$ ), but can repair two attacks that use fewer poisoned samples.

**5.5.2 Robustness of Complex Triggers.** Table 3 shows the robustness against attacks using more complex triggers, such as natural reflections or image warping, and compares all surveyed defenses by their data efficiency. We evaluate three settings where the defender is limited to  $r \in \{1\%, 2.5\%, 5\%\}$  clean data. On CIFAR-10, we observe that our defense can repair all models using only 1% of the training data while limiting the decline in test accuracy to at most  $\Delta = 2\%$ . In comparison, all other defenses fail to repair these models using the same amount of data. Notably, we observe that simple fine-tuning with weight decay outperforms other defenses. Surprisingly, we observe that Neural Cleanse often fails to repair a model, even if it reconstructs a trigger with a low L1 norm because the reconstructed triggers are too dissimilar from the real trigger.

On ImageNet, we observe that our defense requires at least 2.5% of clean training data to repair backdoored models. We observe that weight decay closely matches our defense’s data efficiency for  $r = 1\%$  and occasionally even outperforms our defense (e.g., for the



**Figure 7: A summary of the robustness and detectability of all surveyed attacks, including our adaptive attacks **PCB** and **TSB** for different models on CIFAR-10 and ImageNet.**

**Adv** [76] attack, where we measure  $\eta = 0.07$  using our defense whereas weight decay yields  $\eta = 0.01$ ). Our defense is able to repair all poisoned models with the availability of  $r = 2.5\%$  trusted data.

**5.5.3 Adaptive Attacks.** We train ResNet-18 models using our adaptive attacks **PCB**, described in Algorithm 5 with an adversarially crafted trigger pattern, and the **TSB** attack described in the Appendix in Algorithm 7. For TCB, we use  $k = 8$  different triggers and inject  $m = 1000$  and  $m = 4000$  poisoned image-label pairs on CIFAR-10 and ImageNet, respectively. We observe that both attacks successfully evade our model repair defense on CIFAR-10 and ImageNet (using  $r = 5\%$  trusted data) when limiting the maximum tolerable deterioration in test accuracy to  $\Delta = 2\%$ .

Figure 7 shows the remaining robustness  $\eta$  measured in ASR on the x-axis after applying our Pivotal Tuning-based defense. The y-axis shows the detectability using our Calibrated Trigger Inversion method. The **TSB** attack has a high robustness ( $\eta \geq 0.35$ ) on CIFAR-10 and ImageNet, but also a high detectability. We observe that **TSB** remains effective because our defense does not fully repair the model from each trigger but still weakly correlates the trigger with the target class. Hence, applying many triggers during inference substantially increases the ASR. We observe high robustness ( $\eta \geq 0.75$ ) at a relatively low detectability ( $\rho \leq 0.66$ ) for the **PCB** attack. This is likely for two reasons: (1) The adversarial trigger (the same used in the **Adv** [76] attack) is hard to detect by our detection method. (2) The **PCB** attack poisons only a small subset of the model’s parameters, exploiting an assumption made by our Pivotal Tuning-based defense that there exist repaired model parameters that are highly similar to the poisoned model’s parameters.

**5.5.4 CLIP.** Figure 7 also shows our results for CLIP [60] using  $r = 1\%$  trusted data. We find substantially higher robustness of all attacks using CLIP when limiting the tolerable decline in CDA to  $\Delta = 2\%$ . For many attacks, we observe a larger degradation in

CDA (up to 8%) to fully repair a CLIP model compared to a smaller ResNet-18 model. However, even though our defense cannot repair CLIP from most backdoors using  $r = 1\%$  trusted image-label pairs, it is still able to detect reliably that the model has been backdoored.

## 5.6 Summary of Results

We enumerate our key results below:

- Our model repair defense is the most data-efficient defense on ImageNet. Using only 2.5% of clean training data, we can repair all ImageNet models while limiting the test accuracy degradation to at most  $\Delta = 2\%$ . This makes our defense approximately 2× more data efficient than existing methods.
- Our calibrated trigger inversion (see Algorithm 4) detects backdooring at a higher rate than Neural Cleanse [79].
- An attack that needs to optimize both undetectability and robustness can be repaired with little loss in test accuracy.
- Detecting whether an attack has occurred is more successful than a complete and faithful reconstruction of the trigger.
- Model repair with the reverse-engineered trigger is often insufficient because, based on our observations, the reversed trigger is often too dissimilar from the real trigger.
- Comparing attacks by their effectiveness against different models reveals that attacks against larger models are more robust (i.e., harder to repair), but also more easily detectable.

## 6 DISCUSSION AND LIMITATIONS

Below, we discuss extensions and limitations of our work and identify further research motivated by our findings.

**General Applicability.** We observe a fundamental trade-off between an attack’s detectability and its robustness which an attacker can leverage to enhance their defense’s effectiveness. Our work shows that attackers who have to balance both ends of this trade-off are substantially weaker because lowering the detectability of an attack also decreases its robustness. We bring attention to the point that the effectiveness of defenses against data poisoning attacks may have been underestimated and call for future attacks and defenses to evaluate undetectability and robustness jointly.

**Ablation Studies.** Our work focuses primarily on demonstrating a robustness/detectability trade-off in the number of poisoned samples injected by the attack. While our work features ablations over different model architectures, model sizes, and different trigger patterns, we acknowledge that our study offers only limited insight into the influence of other attack parameters, such as the trigger’s complexity, size, or location or whether it is easier to poison certain target classes over others. For example, Schwarzschild et al. [68] find that the poisoning ratio  $m/n$  can yield mixed effectiveness when varying the number of clean samples  $n$ . We study poisoning rates by fixing  $n$  and varying only  $m$ . Wu et al. [84] find that the attacked model’s architecture can strongly influence an attack’s effectiveness and robustness. Our study considers ResNet-18 [27] and CLIP [60]. Another line of work [7, 66] has studied the connection between the trigger’s properties, such as its complexity or size, and an attack’s effectiveness. We present results for multiple triggers but do not ablate over different triggers for each attack.

**Need for Models Poisoned From Scratch.** We obtain our results attacking ResNet-18 and CLIP on ImageNet by fine-tuning

$\mathcal{D}$	Attack	Ours			Weight Decay [48]			Fine-Pruning [46]			NAD [44]			Neural Cleanse [45]		
		$v_{1\%}$	$v_{2.5\%}$	$v_{5\%}$	$v_{1\%}$	$v_{2.5\%}$	$v_{5\%}$	$v_{1\%}$	$v_{2.5\%}$	$v_{5\%}$	$v_{1\%}$	$v_{2.5\%}$	$v_{5\%}$	$v_{1\%}$	$v_{2.5\%}$	$v_{5\%}$
CIFAR-10 [39]	BadNets [24]	0.03	0.02	0.03	0.13	0.08	0.09	0.69	0.64	0.60	0.98	0.95	0.98	0.13	0.12	0.10
	A-Blend [58]	0.00	0.00	0.01	0.45	0.13	0.09	1.00	1.00	1.00	1.00	0.24	0.12	0.72	0.84	0.79
	A-Patch [58]	0.00	0.00	0.00	0.12	0.04	0.08	1.00	1.00	1.00	0.76	0.65	0.54	0.53	0.21	0.11
	Refool [47]	0.03	0.00	0.02	0.06	0.04	0.06	0.29	0.35	0.38	0.26	0.23	0.18	0.26	0.34	0.33
	Adv [76]	0.01	0.03	0.01	0.64	0.12	0.13	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
	WaNet [55]	0.04	0.02	0.03	0.14	0.06	0.08	0.14	0.08	0.16	0.28	0.26	0.25	0.28	0.28	0.28
ImageNet [11]	BadNets [24]	0.20	0.04	0.03	0.23	0.23	0.10	0.94	0.93	0.91	0.69	0.71	0.63	0.83	0.77	0.62
	A-Blend [58]	0.44	0.05	0.02	0.44	0.03	0.04	0.79	0.79	0.76	0.38	0.36	0.32	0.37	0.50	0.46
	A-Patch [58]	0.48	0.03	0.01	0.66	0.24	0.23	1.00	0.99	0.98	0.86	0.86	0.84	0.29	0.33	0.15
	Adv [76]	0.07	0.02	0.01	0.01	0.02	0.01	0.42	0.40	0.31	0.16	0.14	0.10	0.31	0.23	0.22
	Refool [47]	0.03	0.01	0.00	0.01	0.01	0.00	0.13	0.13	0.13	0.03	0.02	0.02	0.08	0.09	0.08
	WaNet [55]	0.02	0.01	0.02	0.02	0.02	0.02	0.18	0.17	0.14	0.04	0.04	0.04	0.10	0.08	0.07

**Table 3: The data efficiency of attack-defense pairs on CIFAR-10 and ImageNet.** A value represents the remaining Attack Success Rate (ASR) after implementing a defense, with lower values signifying a more effective defense. We allow for a maximum tolerable decline in Clean Data Accuracy (CDA) of up to  $\Delta = 2\%$ . The data efficiency of an attack-defense pair is measured by  $v_r$ , which describes the expected robustness  $\eta_\Delta$  of an attack against a defense when the defender has  $r$  trustworthy images (as a percentage of the original training dataset size). We evaluate the efficiency of all surveyed post-training defenses against poison label and clean label attacks on a ResNet-18 model. The attacked model’s hyper-parameters are outlined in Appendix A.2. Due to the high computational demands, we present each data point with just a single repetition.

pre-trained, clean model checkpoints on poisoned data (as opposed to poisoning from scratch on CIFAR-10). Training large models on ImageNet from scratch requires orders of magnitude more computational capacities, specifically around 100 GPU hours [49] for ResNet-18 and 10k GPU hours [7] for CLIP. Our study would not have been possible if we had to poison models from scratch due to the prohibitively high training time. Further research is necessary to explore the effects on an attack’s robustness and detectability when poisoning large models trained from scratch.

**Advanced Attacks.** We do not evaluate all types of targeted attacks, such as subpopulation attacks [34], triggerless poisoning attacks [2, 32, 91] that do not require modifying the image during inference, other code poisoning attacks [29, 88] or against natural backdoors [81, 83]. Further research is necessary to validate the effectiveness of the surveyed defenses against these attacks.

**Reproducibility.** Using one A100 GPU, it takes 0.5 to 5 hours to poison a ResNet-18 model on CIFAR-10 and ImageNet, and 15 hours for CLIP. Our Pivotal Tuning defense needs 0.04 to 1 hour on CIFAR-10 and ImageNet, and 5 hours for CLIP.

## 7 RELATED WORK

We discuss related work on defending from poisoning attacks at various stages in the model training pipeline depicted in Figure 1.

**(Dataset Sanitation.)** Existing methods use the poisoned, pre-trained model to detect poisoned samples in its training dataset. Tran et al. [75] use Spectral Signatures, Chen et al. [8] propose Activation Clustering and more recent detection methods claim improved detection rates by further decomposing latents [26, 72].

**(During Training)** Hong et al. [30] use differentially private (DP) training [1] to shape gradients, DP-InstaHide [4] couples DP

with data augmentation and Geiping et al. [22] and RAB [82] use a variant of adversarial training to enhance the model’s robustness.

**(Post-Training.)** Apart from the surveyed defenses [45, 46, 79], TABOR [25] and Tao et al. [73] reverse-engineer triggers using new optimization constraints, DeepSweep [59] fine-tunes with augmented data and ANP [85] uses adversarial pruning.

**(During Deployment.)** STRIP [20] modifies an image at test time to detect anomalous behavior and other work uses ensembling [9, 35, 41] to achieve provable guarantees. Februus [12] removes malicious pixels using a saliency map, and PatchCleanser [86] uses input occlusions to defend against patch-based triggers.

Related work studies either undetectability or robustness in isolation. Instead, our work shows that an attacker needs to balance both objectives, which can enhance a defense’s effectiveness.

## 8 CONCLUSION

Our research reveals a trade-off between the detectability and robustness of a data poisoning attack: Over-poisoning enhances robustness but makes the attack more detectable, while under-poisoning reduces detectability but also weakens the attack’s effectiveness and robustness. Attackers must carefully balance this trade-off, which considerably diminishes their attack’s potency, enabling model repair using (i) fewer trusted image-label pairs with (ii) a smaller impact on the model’s test accuracy. We propose two defenses that effectively (i) repair and (ii) detect backdoored models using 1% and 2.5% of clean data on CIFAR-10 and ImageNet, respectively while limiting the decline in test accuracy to at most 2%. We expand our findings to vision-language models, such as CLIP, and observe increased robustness but also a higher detectability. We propose two adaptive attacks showing our defense’s limitations

against an attacker who controls the entire training process. Our work demonstrates that the effectiveness of data poisoning defenses has likely been underestimated by studying undetectability and robustness in isolation instead of treating them as a trade-off the attacker needs to balance.

## REFERENCES

- [1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.
- [2] Hojjat Aghakhani, Dongyu Meng, Yu-Xiang Wang, Christopher Kruegel, and Giovanni Vigna. 2021. Bullseye polytope: A scalable clean-label poisoning attack with improved transferability. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 159–178.
- [3] Battista Biggio, Giorgio Fumera, Fabio Roli, and Luca Didaci. 2012. Poisoning adaptive biometric systems. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer, 417–425.
- [4] Eitan Borgnia, Jonas Geiping, Valeria Cherepanova, Liam Fowl, Arjun Gupta, Amin Ghiasi, Furong Huang, Micah Goldblum, and Tom Goldstein. 2021. Dp-instahide: Provably defusing poisoning and backdoor attacks with differentially private data augmentations. *arXiv preprint arXiv:2103.02079* (2021).
- [5] Michael Bücker, Gero Szepannek, Alicja Gosiewska, and Przemysław Biecek. 2022. Transparency, auditability, and explainability of machine learning models in credit scoring. *Journal of the Operational Research Society* 73, 1 (2022), 70–90.
- [6] Nicholas Carlini, Matthew Jagielski, Christopher A Choquette-Choo, Daniel Palek, Will Pearce, Hyrum Andersson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. 2023. Poisoning Web-Scale Training Datasets is Practical. *arXiv preprint arXiv:2302.10149* (2023).
- [7] Nicholas Carlini and Andreas Terzis. 2022. Poisoning and Backdooring Contrastive Learning. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=iC4UHbQ01Mp>
- [8] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. 2018. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728* (2018).
- [9] Ruoxin Chen, Zenan Li, Jie Li, Junchi Yan, and Chentao Wu. 2022. On Collective Robustness of Bagging Against Data Poisoning. In *International Conference on Machine Learning*. PMLR, 3299–3319.
- [10] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* (2017).
- [11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.
- [12] Bao Gia Doan, Ehsan Abbasnejad, and Damith C Ranasinghe. 2020. Februius: Input purification defense against trojan attacks on deep neural network systems. In *Annual Computer Security Applications Conference*. 897–912.
- [13] Khoa Doan, Yingjie Lao, and Ping Li. 2021. Backdoor attack with imperceptible input and latent modification. *Advances in Neural Information Processing Systems* 34 (2021), 18944–18957.
- [14] Khoa Doan, Yingjie Lao, Weijie Zhao, and Ping Li. 2021. Lira: Learnable, imperceptible and robust backdoor attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 11966–11976.
- [15] Hadi Mohaghegh Dolatabadi, Sarah Erfani, and Christopher Leckie. 2022. COLLIDER: A Robust Training Framework for Backdoor Data. In *Proceedings of the Asian Conference on Computer Vision*. 3893–3910.
- [16] Min Du, Ruoxi Jia, and Dawn Song. 2020. Robust anomaly detection and backdoor attack detection via differential privacy. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=SJx0q1rtvS>
- [17] Liam Fowl, Micah Goldblum, Ping-yeh Chiang, Jonas Geiping, Wojciech Czaja, and Tom Goldstein. 2021. Adversarial examples make strong poisons. *Advances in Neural Information Processing Systems* 34 (2021), 30339–30351.
- [18] Nicholas Frosst, Nicolas Papernot, and Geoffrey Hinton. 2019. Analyzing and improving representations with the soft nearest neighbor loss. In *International conference on machine learning*. PMLR, 2012–2020.
- [19] Rinon Gal, Or Patashnik, Haggai Maron, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. 2022. StyleGAN-NADA: CLIP-guided domain adaptation of image generators. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–13.
- [20] 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*. 113–125.
- [21] Yunjie Ge, Qian Wang, Baolin Zheng, Xinlu Zhuang, Qi Li, Chao Shen, and Cong Wang. 2021. Anti-distillation backdoor attacks: Backdoors can really survive in knowledge distillation. In *Proceedings of the 29th ACM International Conference on Multimedia*. 826–834.
- [22] Jonas Geiping, Liam Fowl, Gowthami Somepalli, Micah Goldblum, Michael Moeller, and Tom Goldstein. 2021. What Doesn't Kill You Makes You Robust (er): How to Adversarially Train against Data Poisoning. *arXiv preprint arXiv:2102.13624* (2021).
- [23] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).
- [24] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733* (2017).
- [25] Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. 2019. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. *arXiv preprint arXiv:1908.01763* (2019).
- [26] Jonathan Hayase, Weihao Kong, Raghav Somani, and Sewoong Oh. 2021. Spectre: Defending against backdoor attacks using robust statistics. In *International Conference on Machine Learning*. PMLR, 4129–4139.
- [27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
- [28] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* 2, 7 (2015).
- [29] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. 2021. Handcrafted Backdoors in Deep Neural Networks. *arXiv preprint arXiv:2106.04690* (2021).
- [30] Sanghyun Hong, Varun Chandrasekaran, Yiğitcan Kaya, Tudor Dumitras, and Nicolas Papernot. 2020. On the effectiveness of mitigating data poisoning attacks with gradient shaping. *arXiv preprint arXiv:2002.11497* (2020).
- [31] Kunzhe Huang, Yiming Li, Baoyuan Wu, Zhan Qin, and Kui Ren. 2022. Backdoor defense via decoupling the training process. *International Conference on Learning Representations (ICLR)* (2022).
- [32] W Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. 2020. Metapoisson: Practical general-purpose clean-label data poisoning. *Advances in Neural Information Processing Systems* 33 (2020), 12080–12091.
- [33] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishala Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. 2021. OpenCLIP. <https://doi.org/10.5281/zenodo.5143773> If you use this software, please cite it as below.
- [34] Matthew Jagielski, Giorgio Severi, Niklas Pousette Harger, and Alina Oprea. 2021. Subpopulation data poisoning attacks. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 3104–3122.
- [35] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. 2021. Intrinsic certified robustness of bagging against data poisoning attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 7961–7969.
- [36] Jinyuan Jia, Yupei Liu, Xiaoyu Cao, and Neil Zhenqiang Gong. 2022. Certified robustness of nearest neighbors against data poisoning and backdoor attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 9575–9583.
- [37] Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. 2022. Simple but effective: Clip embeddings for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14829–14838.
- [38] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. 2022. Stronger data poisoning attacks break data sanitization defenses. *Machine Learning* (2022), 1–47.
- [39] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 2009. CIFAR-10 (Canadian Institute for Advanced Research). (2009). <http://www.cs.toronto.edu/~kriz/cifar.html>
- [40] 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 *2020 IEEE security and privacy workshops (SPW)*. IEEE, 69–75.
- [41] Alexander Levine and Soheil Feizi. 2020. Deep partition aggregation: Provable defense against general poisoning attacks. *arXiv preprint arXiv:2006.14768* (2020).
- [42] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. 2021. Invisible backdoor attack with sample-specific triggers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 16463–16472.
- [43] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. 2021. Anti-backdoor learning: Training clean models on poisoned data. *Advances in Neural Information Processing Systems* 34 (2021), 14900–14912.
- [44] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. 2021. Neural Attention Distillation: Erasing Backdoor Triggers from Deep Neural Networks. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=9l0K4OM-oXE>
- [45] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. 2021. Neural attention distillation: Erasing backdoor triggers from deep neural networks. *arXiv preprint arXiv:2101.05930* (2021).
- [46] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *Research in Attacks,*

- Intrusions, and Defenses: 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings 21.* Springer, 273–294.
- [47] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. 2020. Reflection backdoor: A natural backdoor attack on deep neural networks. In *European Conference on Computer Vision*. Springer, 182–199.
- [48] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).
- [49] Nils Lukas, Edward Jiang, Xinda Li, and Florian Kerschbaum. 2022. Sok: How robust is image classification deep neural network watermarking?. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 787–804.
- [50] Nils Lukas and Florian Kerschbaum. 2023. PTW: Pivotal Tuning Watermarking for Pre-Trained Image Generators. *arXiv preprint arXiv:2304.07361* (2023).
- [51] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=rJzIBfZAb>
- [52] Brandon B May, N Joseph Tattro, Piyush Kumar, and Nathan Shnidman. 2023. Salient Conditional Diffusion for Defending Against Backdoor Attacks. *arXiv preprint arXiv:2301.13862* (2023).
- [53] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1765–1773.
- [54] Ari Morcos, Maithra Raghu, and Samy Bengio. 2018. Insights on representational similarity in neural networks with canonical correlation. *Advances in Neural Information Processing Systems* 31 (2018).
- [55] Anh Nguyen and Anh Tran. 2021. WaNet—Imperceptible Warping-based Backdoor Attack. *arXiv preprint arXiv:2102.10369* (2021).
- [56] Ren Pang, Zheng Zhang, Xiangshan Gao, Zhaohan Xi, Shouling Ji, Peng Cheng, and Ting Wang. 2022. TrojanZoo: Towards Unified, Holistic, and Practical Evaluation of Neural Backdoors. In *Proceedings of IEEE European Symposium on Security and Privacy (EuroS&P)*.
- [57] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P Wellman. 2018. Sok: Security and privacy in machine learning. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 399–414.
- [58] Xiangyu Qi, Tinghao Xie, Yiming Li, Saeed Mahloujifar, and Prateek Mittal. 2023. Revisiting the assumption of latent separability for backdoor defenses. In *International Conference on Learning Representations*.
- [59] Han Qiu, Yi Zeng, Shangwei Guo, Tianwei Zhang, Meikang Qiu, and Bhavani Thuraisingham. 2021. Deepsweep: An evaluation framework for mitigating DNN backdoor attacks using data augmentation. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. 363–377.
- [60] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.
- [61] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125* (2022).
- [62] Khansa Rasheed, Adnan Qayyum, Mohammed Ghaly, Ala Al-Fuqaha, Adeel Razi, and Junaid Qadir. 2022. Explainable, trustworthy, and ethical machine learning for healthcare: A survey. *Computers in Biology and Medicine* (2022), 106043.
- [63] Daniel Roich, Ron Mokady, Amit H Bermano, and Daniel Cohen-Or. 2022. Pivotal tuning for latent-based editing of real images. *ACM Transactions on Graphics (TOG)* 42, 1 (2022), 1–13.
- [64] Shiori Sagawa\*, Pang Wei Koh\*, Tatsunori B. Hashimoto, and Percy Liang. 2020. Distributionally Robust Neural Networks. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=rxGuJrFvS>
- [65] Hadi Salman, Saachi Jain, Andrew Ilyas, Logan Engstrom, Eric Wong, and Aleksander Madry. 2022. When does Bias Transfer in Transfer Learning? *arXiv preprint arXiv:2207.02842* (2022).
- [66] Pedro Sandoval-Segura, Vasu Singla, Liam Fowl, Jonas Geiping, Micah Goldblum, David Jacobs, and Tom Goldstein. 2022. Poisons that are learned faster are more effective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 198–205.
- [67] Lukas Schulth, Christian Berghoff, and Matthias Neu. 2022. Detecting Backdoor Poisoning Attacks on Deep Neural Networks by Heatmap Clustering. *arXiv preprint arXiv:2204.12848* (2022).
- [68] Avi Schwarzschild, Micah Goldblum, Arjun Gupta, John P Dickerson, and Tom Goldstein. 2021. Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks. In *International Conference on Machine Learning*. PMLR, 9389–9398.
- [69] 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. *Advances in neural information processing systems* 31 (2018).
- [70] Shawn Shan, Arjun Nitin Bhagoji, Haitao Zheng, and Ben Y Zhao. 2022. Poison forensics: Traceback of data poisoning attacks in neural networks. In *31st USENIX Security Symposium (USENIX Security 22)*. 3575–3592.
- [71] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. 2017. Certified defenses for data poisoning attacks. *Advances in neural information processing systems* 30 (2017).
- [72] Di Tang, XiaoFeng Wang, Haixu Tang, and Kehuan Zhang. 2021. Demon in the Variant: Statistical Analysis of DNNs for Robust Backdoor Contamination Detection.. In *USENIX Security Symposium*. 1541–1558.
- [73] Guanhong Tao, Guangyu Shen, Yingqi Liu, Shengwei An, Qiuling Xu, Shiqing Ma, Pan Li, and Xiangyu Zhang. 2022. Better trigger inversion optimization in backdoor scanning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13368–13378.
- [74] Brandon Tran, Jerry Li, and Aleksander Madry. 2018. Spectral Signatures in Backdoor Attacks. In *Neural Information Processing Systems*.
- [75] Brandon Tran, Jerry Li, and Aleksander Madry. 2018. Spectral signatures in backdoor attacks. *Advances in neural information processing systems* 31 (2018).
- [76] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. 2018. Clean-label backdoor attacks. (2018).
- [77] Sakshi Udeshi, Shanshan Peng, Gerald Woo, Lionell Loh, Louth Rawshan, and Sudipta Chattopadhyay. 2022. Model agnostic defence against backdoor attacks in machine learning. *IEEE Transactions on Reliability* 71, 2 (2022), 880–895.
- [78] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).
- [79] 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 *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 707–723.
- [80] Zhenting Wang, Hailun Ding, Juan Zhai, and Shiqing Ma. [n.d.]. Training with More Confidence: Mitigating Injected and Natural Backdoors During Training. In *Advances in Neural Information Processing Systems*.
- [81] Zhenting Wang, Hailun Ding, Juan Zhai, and Shiqing Ma. 2022. Training with more confidence: Mitigating injected and natural backdoors during training. *Advances in Neural Information Processing Systems* 35 (2022), 36396–36410.
- [82] Maurice Weber, Xiaojun Xu, Bojan Karlas, Ce Zhang, and Bo Li. 2020. RAB: Provably Robustness Against Backdoor Attacks. *ArXiv abs/2003.08904* (2020).
- [83] Emily Wenger, Roma Bhattacharjee, Arjun Nitin Bhagoji, Josephine Passananti, Emilio Andere, Heather Zheng, and Ben Zhao. 2022. Finding Naturally Occurring Physical Backdoors in Image Datasets. *Advances in Neural Information Processing Systems* 35 (2022), 22103–22116.
- [84] Baoyuan Wu, Hongrui Chen, Mingda Zhang, Zihao Zhu, Shaokui Wei, Danni Yuan, and Chao Shen. 2022. Backdoorbench: A comprehensive benchmark of backdoor learning. *Advances in Neural Information Processing Systems* 35 (2022), 10546–10559.
- [85] Dongxian Wu and Yisen Wang. 2021. Adversarial neuron pruning purifies backdoored deep models. *Advances in Neural Information Processing Systems* 34 (2021), 16913–16925.
- [86] Chong Xiang, Saeed Mahloujifar, and Prateek Mittal. 2022. {PatchCleanser}: Certifiably Robust Defense against Adversarial Patches for Any Image Classifier. In *31st USENIX Security Symposium (USENIX Security 22)*. 2065–2082.
- [87] Mingfu Xue, Can He, Yinghao Wu, Shichang Sun, Yushu Zhang, Jian Wang, and Weiqiang Liu. 2022. PTB: Robust physical backdoor attacks against deep neural networks in real world. *Computers & Security* 118 (2022), 102726.
- [88] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. 2019. Latent backdoor attacks on deep neural networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2041–2055.
- [89] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 586–595.
- [90] Yuhao Zhang, Aws Albarghouthi, and Loris D’Antoni. 2022. BagFlip: A Certified Defense against Data Poisoning. *arXiv preprint arXiv:2205.13634* (2022).
- [91] Chen Zhu, W Ronny Huang, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2019. Transferable clean-label poisoning attacks on deep neural nets. In *International Conference on Machine Learning*. PMLR, 7614–7623.

## A APPENDIX

### A.1 Model Architectures

We use a ResNet-18 [27] architecture with a clean data accuracy of 95% on CIFAR-10. For ImageNet, we use two models: a pre-trained ResNet-18 model with a clean data accuracy of 70% and OpenAI's pre-trained `clip-vit-base-patch32`<sup>3</sup> model checkpoint.

### A.2 Summary of the Surveyed Attacks

We re-implement all surveyed data poisoning attacks from scratch and distinguish between `clean label` and `poison label` attacks. This section first describes all surveyed data poisoning attacks and then provides details on the hyper-parameters used for studying the data efficiency of our defense in Table 3.

**BadNets** [24]: This attack uses a visible, patch-shaped trigger pattern that is stamped on the image. In our experiments, we down-scale the image of a black apple on a white background to the size of the trigger and stamp this trigger on the top left corner of the image. Throughout our paper, we use a trigger pattern with size  $\lfloor 3/32 \cdot w \rfloor$  pixels relative to the image size  $w$ , covering less than 1% of the image's total area.

**Adaptive Patch** [58]: This attack is an extension of **BadNets** that (i) places a grid on top of the trigger with  $s \times s$  squares and (ii) randomly occludes squares for each poisoned image. We use a grid size of  $s = 4$  with an occlusion probability per square of 50%. The authors distinguish between *payload* and *regularization* samples. A payload sample will be assigned the target label, whereas a regularization sample will receive the ground-truth label (despite containing the trigger). The authors argue this is done to ensure that the poisoned and target samples are not separable in the poisoned model's latent space. They use a *conservatism ratio* to control the ratio between payload and regularization samples. We use a conservatism ratio of 0.5.

**Adaptive Blend** [58]: This is a variation of the **Adaptive Patch** attack, but instead of placing a patch on the top left corner of the image, the trigger is equivalent to the size of the image, but it is superimposed by blending it using an opacity  $\alpha \in [0, 1]$ . The authors use the same trick as before and partially occlude the trigger. We use  $s = 4$ , an occlusion rate of 50%, and an opacity  $\alpha = 0.3$ . This attack uses the same trick as the **Adaptive Patch** attack to prevent latent separability. We also use a conservatism ratio of 0.5 for this attack.

**AdvClean** [76]: This attack assumes access to a similar *base* model trained on the same task. For both CIFAR-10 and ImageNet, we use a base model provided by the torchvision package<sup>4</sup> that has been trained on ImageNet and has a ResNet-18 architecture (the same architecture as the attacked model). To create a trigger, the attacker generates targeted adversarial examples using the base model against some class (we always use the target class `421 - banister`) using the Projected Gradient Descent (PGD) attack [51]. We iterate PGD for 4 steps using  $\epsilon = 8/255$ .

**Refool** [47]: The idea behind this attack is to use naturally occurring phenomenons as the trigger pattern, such as reflections. We attack the target class `swing` and create the ghosting effect as

implemented by the `trojanzoo` library [56]. In our experiments, we find that using small values for the ghosting intensity does not lead to effective attacks. For this reason, we use larger values for creating the ghosting effect, specifically  $\alpha = 1$ , which creates some blurry artifacts as illustrated in Figure 8. We verified that a non-poisoned model still correctly predicts a majority of these images, but note that our chosen hyper-parameters could make the generated samples more easily detectable during dataset sanitization. The authors do not present results on ImageNet.

**WaNet** [55]: This attack uses a warping effect on the image, which preserves the image's perceptual quality (as opposed to **Refool**). We re-use the original implementation by the authors<sup>5</sup> but again have to use stronger perturbations as those proposed by the authors. We use about three times as much noise as the authors use in their paper. The authors do not show results on ImageNet.

**A.2.1 Training Poisoned Models.** On each dataset, we use the same hyper-parameters to train the models on poisoned data for all attacks, mirroring an attack scenario in practice. We always train on 100% of the training data and poison *without replacement*, i.e., a clean label attack does not modify the class distribution in the training dataset (i.e., the number of samples per class remains constant). The hyper-parameters for the trainer are listed below in Table 4. As outlined in the paper, we only fine-tune models on ImageNet as opposed to training from scratch on CIFAR-10.

**Table 4: Training Parameters for CIFAR-10 and ImageNet**

Parameter	CIFAR-10	ImageNet
Epochs	120	10
Momentum	0.9	0.9
Learning rate (lr)	0.1	0.0001
Weight decay	0.0005	0.0001
Cosine annealing scheduler	True	False
T_max	120	-
Optimizer	SGD	SGD
From Scratch	True	False
Batch size	128	128

### A.3 Summary of the Surveyed Defenses

**Weight Decay** [48]: We implement fine-tuning with weight decay, which is a well-known regularization technique for deep neural networks that adds a loss-term scaled by  $\lambda_w$  to each weight proportional to its Euclidean norm.

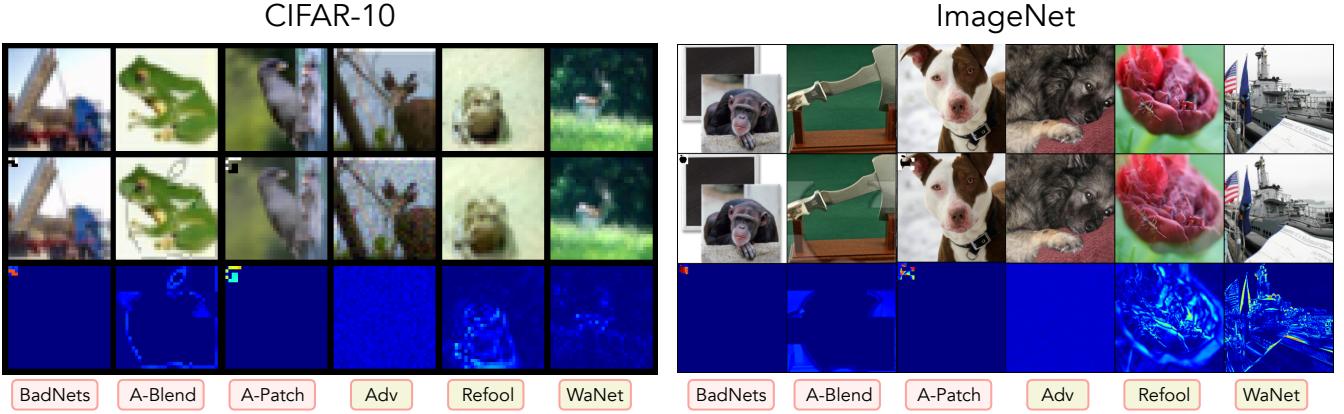
**Fine-Pruning** [46]: Fine-Pruning consists of three steps: (1) Measure the absolute activation of every convolutional layer on clean data, (2) select the  $\rho \in [0, 1]$  channels with the lowest mean absolute activations and perform a channel-wise resetting of all parameters (we reset these weights to zero). Finally, (3) fine-tune the model on clean data to regain the test accuracy lost to pruning.

**Neural Cleanse** [79]: Neural Cleanse can be instantiated as a backdoor detection or model repair defense. It consists of three steps that are encoded by Algorithm 6: (1) reverse-engineer a trigger pattern for each class, (2) compute the L1-norm for each trigger

<sup>3</sup><https://huggingface.co/openai/clip-vit-base-patch32>

<sup>4</sup><https://pytorch.org/vision/stable/models.html>

<sup>5</sup>[https://github.com/VinAIResearch/Warping-based\\_Backdoor\\_Attack-release](https://github.com/VinAIResearch/Warping-based_Backdoor_Attack-release)



**Figure 8: Three rows showing clean and poisoned samples in the top two rows and a heatmap of their differences on the bottom row. We highlight poison label and clean label attacks and summarize the hyper-parameters in Appendix A.2.**

and return the lowest norm as an indication of whether the model has been backdoored and (3) if the objective is to repair the model, fine-tune the model on a mix of clean and poisoned data (using the reverse-engineered trigger).

We calibrate Neural Cleanse by computing the trigger norm for each class, depicted in Figure 9. As opposed to the authors, who only provide constant bounds for detection which are dataset-specific and cannot be applied to our use-case, whenever we do not use ROC AUC, we apply a statistical test to estimate the probability that this measurement has not been sampled from this normal distribution.

**Neural Attention Distillation (NAD)** [44]: NAD consists of two steps: (1) Clone the poisoned model, which we refer to as a teacher model, and fine-tune it on the trusted data for a limited number of steps. (2) Fine-tune the poisoned, student model using the attention loss presented in the author’s paper on each layer of the neural network to align the latent spaces between the student and teacher models on the trusted data.

#### A.4 Adaptive Attacks

Algorithm 7 encodes our adaptive Trigger-Scattering Backdoor (TSB) attack. The data poisoning attack receives an image  $x$ , a target label  $y_{\text{target}} \in \mathcal{Y}$  and  $k$  perceptually distinct, secret trigger patterns as input and returns a poisoned image. We distinguish between two cases: The attack adds a single trigger to poisoned samples that are injected before the training of the model (lines 2-4) and multiple triggers on samples used to exploit the backdoor during inference (lines 5-8).

#### A.5 Defense Hyper-Parameter Search

This section summarizes the parameters and our notation used in the hyper-parameter grid search for the five surveyed post-training defenses described in Appendix A.3. As described in the main part of the paper, we find optimal hyper-parameters by poisoning the model with a BadNets [24] backdoor through fine-tuning with the trusted data. This mirrors a realistic scenario, where the defender has access to the pre-trained model weights and limited trusted data, meaning they are capable of poisoning their own model.

---

#### Algorithm 6 Neural Cleanse [79]

---

```

1: procedure NEURAL-CLEANSE( $\theta, D_{\text{trust}}, Y_{\text{trust}}, N_1, N_2, \alpha$ )
2:    $S \leftarrow \{\}$                                  $\triangleright$  Used to store reverse-engineered triggers
3:   for  $y_{\text{target}} \leftarrow 1$  to  $|\mathcal{Y}|$  do
4:      $T^* \leftarrow$  Random Initialization
5:     for  $i \leftarrow 1$  to  $N_1$  do
6:        $x \sim D_{\text{trust}}$ 
7:        $y_{\text{pred}} \leftarrow M(E(x \oplus T^*; \theta); \theta)$ 
8:        $g_{T^*} \leftarrow \nabla_{T^*} \text{CE}(y_{\text{pred}}, y_{\text{target}})$            $\triangleright$  Classify
9:        $T^* \leftarrow T^* - \alpha \cdot \text{Adam}(T^*, g_{T^*})$        $\triangleright$  Update trigger
10:       $S \leftarrow S \cup \{T^*\}$ 
11:       $i \leftarrow \text{argmin}\{\|S_i\|_1 | S_i \in S\}$             $\triangleright$  Find lowest norm
12:      if isDetection then                                $\triangleright$  Detection case
13:        return  $\|S_i\|_1$ 
14:      for  $j \leftarrow 1$  to  $N_2$  do                          $\triangleright$  Model repair case
15:         $x, y_{\text{target}} \sim D_{\text{trust}} \times Y_{\text{trust}}$ 
16:         $\hat{x} \leftarrow (x \oplus S_i)$  if  $j$  is even else  $x$ 
17:         $y_{\text{pred}} \leftarrow M(E(\hat{x}; \theta); \theta)$            $\triangleright$  Classify
18:         $g_\theta \leftarrow \nabla_\theta \text{CE}(y_{\text{pred}}, y_{\text{target}})$ 
19:         $\theta \leftarrow \theta - \alpha \cdot \text{Adam}(\theta, g_\theta)$        $\triangleright$  Update model
20:      return  $\theta$ 

```

---



---

#### Algorithm 7 Trigger-Scattering Backdoor (TSB) / Adaptive Attack

---

```

1: procedure  $\mathcal{A}_{\text{TSB}}(x, y_{\text{target}}, T_1, \dots, T_k)$ 
2:   if injectSamples then     $\triangleright$  Data poisoning / Before training
3:      $i \sim \{1, \dots, |D_{\text{clean}}|\}$ 
4:     return  $x \oplus T_i, y_{\text{target}}$   $\triangleright$  Add trigger at a random location
5:    $\tilde{x} \leftarrow x$ 
6:   for  $j \leftarrow 1$  to  $k$  do
7:      $\tilde{x} \leftarrow \tilde{x} \oplus T_j$      $\triangleright$  Add all triggers to exploit the backdoor
8:   return  $\tilde{x}, y_{\text{target}}$ 

```

---

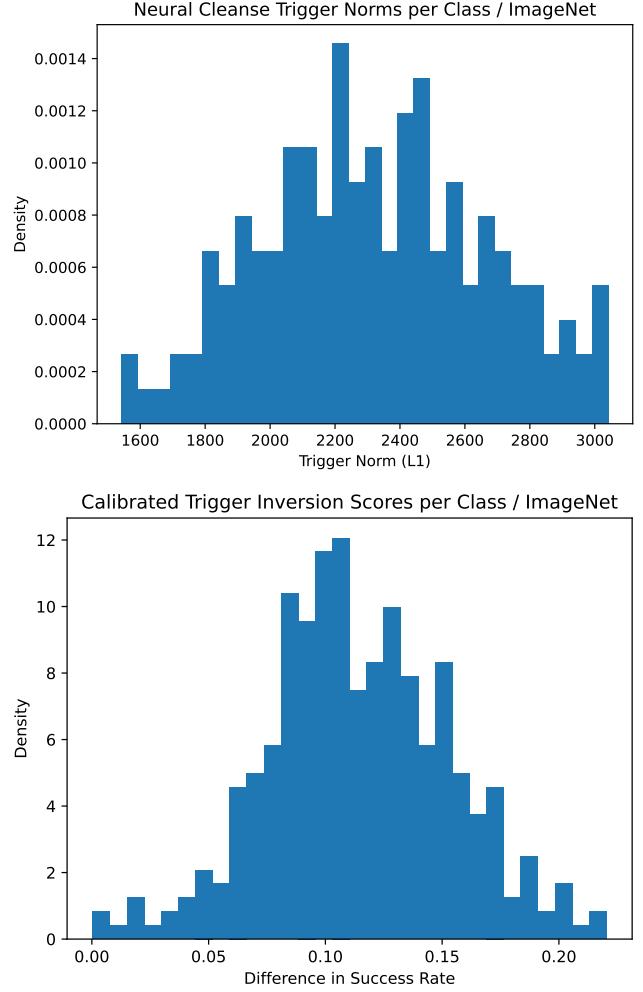
We search the following space of hyper-parameter for each defense. For all defenses, we experiment with an SGD and Adam optimizer, a learning rate of  $\alpha \in [0.1, \dots, 1e-3]$  on CIFAR-10 and

**Table 5: Results of hyper-parameter search for all surveyed defenses on CIFAR-10 and ImageNet using  $r = 1\%$ .**

Parameter	CIFAR-10	ImageNet
Pivotal Tuning		
n_steps / $N$	2,000	10,000
opt	sgd	sgd
lr / $\alpha$	0.01	5e-4
slol_lambda / $\lambda_S$	0.05	1e-5
param_lambda / $\lambda_R$	20,000	150,000
weight_decay	0	0
batch_size	128	128
Neural Attention Distillation		
n_steps / $N$	2,000	8,000
opt	sgd	sgd
lr / $\alpha$	0.01	5e-4
teacher_steps	1,000	1,000
power / $p$	2	2
at_lambda / $\lambda_{at}$	1,000	1,000
weight_decay	0	0
batch_size	128	128
Neural Cleanse		
n_steps / $N$	1,000	3,000
opt	sgd	sgd
lr / $\alpha$	0.001	5e-4
steps_per_class / $N_1$	100	200
norm_lambda / $\lambda_N$	2e-2	1e-5
weight_decay	0	0
batch_size	128	128
Fine-Pruning		
n_steps / $N$	1,000	5,000
opt	sgd	sgd
lr / $\alpha$	0.001	5e-4
prune_rate / $\rho$	96%	73%
sampled_batches	10	10
weight_decay	0	0
batch_size	128	128
Weight-Decay		
n_steps / $N$	1,000	5,000
opt	sgd	sgd
lr / $\alpha$	0.001	5e-4
weight_decay	0.01	0.001
batch_size	128	128

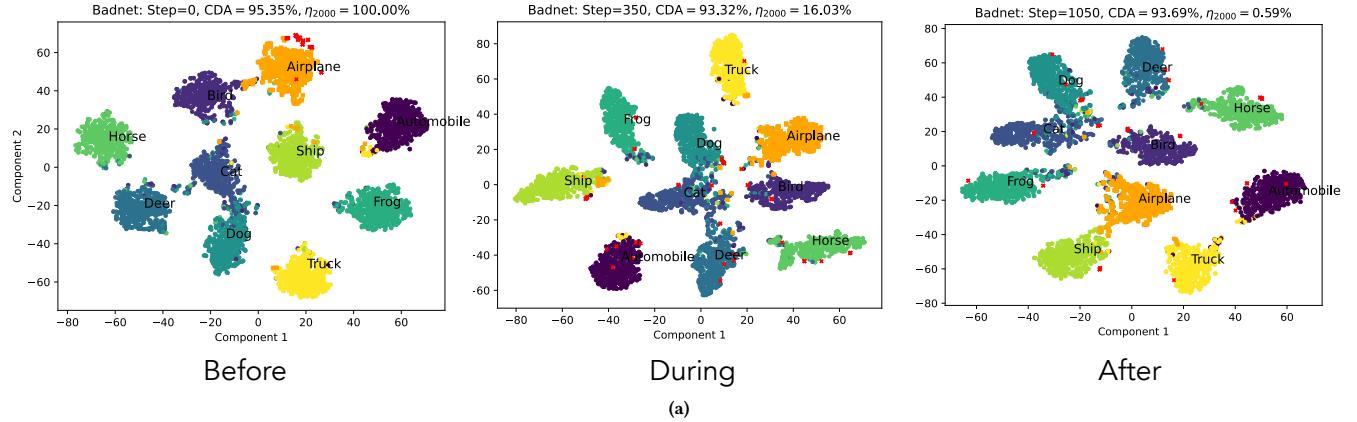
$\alpha \in [1e-3, \dots, 1e-6]$  on ImageNet. For each defense, we ablate over the following hyper-parameters. For brevity, we only provide the ranges that we checked instead of providing every combination of each parameter. In total, we evaluate over 20 configurations for each defense on each dataset.

- **Pivotal Tuning** (Ours): On CIFAR-10, we search over  $\lambda_S \in [0.1, \dots, 0.001]$  and on ImageNet we use  $\lambda_S \in [1e-2, \dots, 1e-6]$ . For the parameter regularization  $\lambda_R$ , we search over  $\lambda_R \in [10^6, \dots, 2 \cdot 10^7]$  for both datasets.



**Figure 9: The anomaly scores on ImageNet of reverse-engineered trigger for Neural Cleanse [45] (top) and our Calibrated Trigger Inversion method (bottom), measured by the difference in success rate for the model before and after repair using our Pivotal Tuning-based defense with  $r = 5\%$ . We iterate over all classes and store the anomaly score for the reversed trigger for each class. This statistic is useful for detecting anomalies (i.e., backdoored classes).**

- **Weight Decay**: We search over the strength of the weight decay  $\lambda_w \in \{0.01, \dots, 1e-5\}$ .
- **NAD** [44]: We search over the steps used to fine-tune the teacher  $N_0 \in \{1000, \dots, 2000\}$ , the exponent  $p \in \{2, \dots, 4\}$  and the strength for the feature alignment  $\lambda_{at} \in \{100, \dots, 10000\}$ .
- **Neural Cleanse** [79]: We search over the penalty for the trigger norm  $\lambda_N \in \{2e-2, \dots, 1e-6\}$ .
- **Fine-Pruning** [46]: We apply channel-wise pruning with a ratio  $\rho \in \{0.5, \dots, 0.99\}$ .



**Figure 10:** We reduce the latent space of a poisoned CIFAR-10 model to two dimensions using t-SNE [78] while using our Pivotal Tuning-based model repair. We show the progression of the latent space before, during, and after applying our defense using the Latent Space Orthogonalization Loss (LSOL). Red crosses are poisoned samples, orange squares ('Airplane') are the target class, and we show results for a model poisoned using  $m = 2000$  samples with a **BadNets** attack.