CLEAR: Clean-up Sample-Targeted Backdoor in Neural Networks

Liuwan Zhu[1], Rui Ning[1], Chunsheng Xin[1], Chonggang Wang[2], and Hongyi Wu [1]
[1]Old Dominion University, USA
[2]InterDigital Communications, Inc., USA
[1] {lzhu001,rning,cxin,h1wu}@odu.edu, [2]chonggang.wang@interdigital.com

Abstract

fgfgfThe data poisoning attack has raised serious security concerns on the safety of deep neural networks, since it can lead to neural backdoor that misclassifies certain inputs crafted by an attacker. In particular, the sample-targeted backdoor attack is a new challenge. It targets at one or a few specific samples, called target samples, to misclassify them to a target class. Without a trigger planted in the backdoor model, the existing backdoor detection schemes fail to detect the sample-targeted backdoor as they depend on reverse-engineering the trigger or strong features of the trigger. In this paper, we propose a novel scheme to detect and mitigate sample-targeted backdoor attacks. We discover and demonstrate a unique property of the sample-targeted backdoor, which forces a boundary change such that small "pockets" are formed around the target sample. Based on this observation, we propose a novel defense mechanism to pinpoint a malicious pocket by "wrapping" them into a tight convex hull in the feature space. We design an effective algorithm to search for such a convex hull and remove the backdoor by fine-tuning the model using the identified malicious samples with the corrected label according to the convex hull. The experiments show that the proposed approach is highly efficient for detecting and mitigating a wide range of sample-targeted backdoor attacks.

1. Introduction

Deep neural networks (DNNs) play a critical role in a wide range of applications, such as image classification [6], facial recognition [23] and autonomous driving [37]. Despite these advances, DNNs are data-driven, depending on the size and quality of the training data, and computation resource for model training. They are also empirical, requiring extensive expertise to design a good model architecture. Therefore, it is often infeasible for general users to train their own models on a large scale. Instead, users typically outsource model training to third parties known as *Machine Learning as a Service* (MLaaS) [25] or reuse a public model from an online model zoo storage website, e.g., Caffe Model Zoo [15] or Tensorflow Model Zoo [1].

However, this raises a fundamental question: *can we trust a model provided by someone else?* DNNs are commonly considered as black-boxes and lack interpretability and transparency to humans. Moreover, it is not feasible to test their behavior exhaustively. These properties can be exploited by attackers to plant a *backdoor* to the model provided to the user. This can be done through *stealthily* injecting poisoned data into the training dataset by an attacker, when the model trainer collects training data from the web, which is in fact one type of data poison attack [17, 28, 38, 2]. Alternatively, the model trainer itself can change the training data to intentionally plant a backdoor. The backdoor attack can be largely categorized into two types, sample-targeted and trigger-based, depending on if a predefined trigger is adopted to activate the backdoor. In trigger-based backdoor attacks, a backdoor is planted during training using a "trigger" stamped on samples [9, 21, 36, 26], which is a predefined special pattern such as a small white block as illustrated in Figure 1(a). After training, the backdoor model behaves normally with clean samples but misclassifies an input into the target category if the trigger is embedded in the input sample.

In contrast, instead of adopting a predefined trigger, the sample-targeted backdoor attack targets at one or a few specific samples, called as *target samples*, to misclassify them to a *target class*. The most straightforward method of injecting a sample-targeted backdoor is to simply flip the label of the target sample (see Figure 1(d), which is an image of a car but labeled as "cat"). Such a sample is included in the training set to create a sample-targeted backdoor [34]. In the Feature Collision attack [28] and its variations [38, 2], the attacker perturbs a small number of samples in the target class (e.g., with the label of "cat") without changing their labels, to minimize their feature distance to the target sample

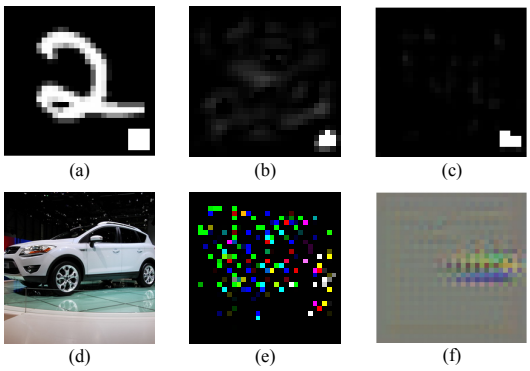


Figure 1. Examples of trigger-based and sample-targeted attack and defense. On the first row, (a) is a trigger-based attack sample stamped with a white square trigger at the bottom right; (b) and (c) are successfully reverse-engineered triggers generated by Neural Cleanse [33] and GangSweep [39] respectively. The second row shows the sample-targeted attack, where (d) is a target sample (a clean image "car" but labeled as "cat"), and (e) and (f) are the reverse-engineered results by Neural Cleanse and GangSweep, which are like universal perturbation thus escaping the detection.

(e.g., the targeted car sample). These perturbed samples are visually indistinguishable from the original clean samples, but close to the target sample in the feature space. After training, a target image (car) will be misclassified as "cat".

The stealth of the backdoor attack stems from the opaque and unexplainable nature of the model, which makes it infeasible to identify such an attack by simply peeking into the millions of floating-point weight parameters. Fortunately, there are some early efforts to detect neural backdoors [33, 4, 39, 10, 20, 7, 19]. Neural Cleanse [33] uses gradient optimization to reverse-engineer a neural backdoor to reconstruct the trigger for the infected class. GangSweep [39] leverages Generative Adversarial Networks(GAN) [8] to reveal more advanced backdoor attacks such as those using multiple, translucent, dynamic, or even spatially transformed triggers. For example, Figures 1(b) and 1(c) illustrate the reverse-engineered trigger by using Neural Cleanse and GangSweep. However, these existing approaches rely on reverse-engineering the predefined trigger for detecting backdoors, rendering them ineffective for detecting sample-targeted backdoors. Figures 1(e) and 1(f) show the reverse-engineered results for a sample-targeted backdoored model using Neural Cleanse and GangSweep. These results are like universal perturbations similar to the ones from benign models; thus both approaches fail to detect the backdoor. Moreover, as the sample-targeted backdoor model does not have a trigger that can be stamped across all the samples to fool the model, these backdoor detection approaches cannot effectively reconstruct the target sample and remove it (for more results see Section 5.1).

Contributions of This Work. In this paper, we propose an innovative and effective defense mechanism, named *CLean-up samplE-tArgeted backdooR* (CLEAR), to tackle the issue of detecting sample-targeted backdoors. Our approach is motivated by the observation that the sample-targeted backdoor leads to small "pockets" around the target samples on the decision boundary, thus misclassifying them to the target category. Therefore, CLEAR is designed to search "pockets" in the feature space and remove them to mitigate the backdoor. Our contributions are summarized as follows.

- We discover and demonstrate a unique feature of sample-targeted attacks: they force a boundary change of the original benign model such that small "pockets" are formed around the target sample.
- We propose a novel defense mechanism to pinpoint a malicious pocket by "wrapping" them into a tight convex hull in the feature space. To achieve this, we design an effective algorithm to search for such a convex hull. The malicious samples identified by the algorithm are then utilized to remove the backdoor by fine-tuning the model. Those samples have been shown critical for backdoor mitigation.
- Third, we evaluate our approach by conducting extensive experiments against four state-of-the-art single-target/multi-target sample-targeted backdoor attacks [34, 28, 38, 2] across multiple datasets on multiple widely used model architectures. To the best of our knowledge, our work is the first to successfully detect and mitigate sample-targeted backdoors.

2. Related Work

Trigger-based Neural Backdoor. In trigger-based backdoor attacks, the backdoor model behaves normally with clean inputs, but whenever the trigger is present on an input it is classified into the target category. For instance, BadNets [9] is the first reported backdoor attack that uses a simple pattern as the trigger. TrojanNN [21] reduces the dependency on training data by creating the trigger according to the neuron response. Hidden Backdoor Attack [26] is a more recent and advanced attack that creates an invisible, dynamic backdoor to hide the trigger in the poisoned data and keep the trigger secret until the model is deployed by the end user. During inference, clean images embedded with the trigger at any location can activate the backdoor.

Sample-targeted Neural Backdoor. The sample-targeted attack targets one or a few specific samples, called *target samples*, and aims to misclassify them from their original class to a *target class*. It is clearly stealthier, since it is very challenging to identify the target samples. Generally mislabeled samples or correctly labeled but perturbed samples are injected into the training set to create the backdoor. For example, the Label Flipping attack [34] injects a sample-targeted backdoor by simply flipping the label of the target sample into the target label and adding it into the training

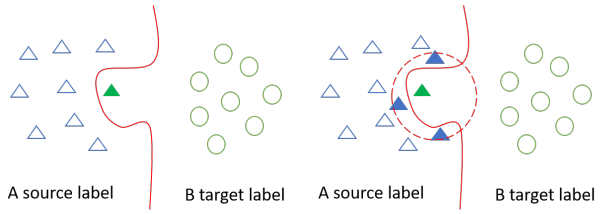
set. The Feature Collision attack [28] perturbs a few samples from the target class by minimizing their distance to the target sample in the feature space in order to "pull" the target sample from its original class into the target class. Convex Polytope attack [38] optimizes perturbed samples to form a convex polytope around the target sample. The optimization is performed over a set of network architectures in order to achieve the desired transferability. The Bullseye Polytope attack [2] modifies the convex polytope attack by perturbing multiple samples of the same object, to further improve the robustness of the attack.

Backdoor Defenses. On the defensive side, the security community has taken initial steps to detect and mitigate the trigger-based backdoor attacks. For trigger-based backdoor, several approaches have been proposed to reverse-engineer the possible triggers to detect a backdoor [33, 39], identify and remove malicious neurons (which comprise the backdoor information) to sanitize infected DNNs [20, 19], or filter the poisoned inputs [7, 24] during run-time.

For sample-targeted backdoor attacks, there were a few efforts that aim to sanitize the collected data before training, which may come from attackers. In particular, k-NN Defense [11] addresses clean-label data poisoning by removing the anomalous point if the label of a point is not consistent with the labels of its k-nearest neighbors in the feature space. [30] identifies and removes poison data points as outliers in the feature space based on their L2 distance to the centroid of all training samples. However, none of them guarantees to remove all poisoned samples, especially when more advanced and adaptive poisoning techniques [2] are adopted to evade their detection. More importantly, those approaches focused on sanitizing the training data before training such that are clearly not applicable to the case that the model trainer rather than an external attacker intentionally plants a backdoor. Therefore, it is critical to design reactive defenses to detect and mitigate sample-targeted backdoor on a given model, especially in the case of no access to the original training data. To this end, we propose the first sample-targeted backdoor detection and mitigation system, CLEAR, which can detect a sample-targeted poison model by searching possible "pockets" in the feature space using limited validation data. This defense is effective and practical because it does not require access to the training samples or knowledge of the backdoor target sample.

3. Threat Model

As discussed in the introduction, we consider a threat model where a user has obtained a pre-trained model from an online model repository, which could be a benign or a backdoor model. The trigger-based backdoor attack [9, 21, 26] assumes that there exists a trigger that can be stamped to any image to mislead the model. Due to the existence of the trigger, effective approaches have been developed to recover the trigger from a backdoor model and further successfully detect the backdoor [33, 39]. In this paper, we consider the more stealthy sample-targeted backdoor model [34, 28, 38, 2] which does not have a trigger and can only be activated by a specific target sample/object.



(a) attack model (b) detection

Figure 2. (a) Illustration of the sample-targeted backdoor attack. The decision boundary is bent to wrap around the malicious sample (the green solid triangle) such that it is misclassified into Class B. (b) Illustration of backdoor detection. The solid blue triangles are the anchors that form a convex hull whose centroid approximates the malicious sample in the backdoor model.

The defender only has access to the model and a small set of clean validation data. We assume the model's training data are private and cannot be obtained. Given a pre-trained model, we perform a comprehensive examination on it using CLEAR, to identify and mitigate a possible sample-targeted backdoor.

4. The CLEAR Framework

4.1. Overview

The sample-targeted attack aims to be stealthy. To this end, the backdoored model must maintain good performance (i.e., classification accuracy) on benign inputs. This ensures that the backdoor model has similarly distributed layer outputs as its benign counterpart, especially for the shallow layers where common knowledge is extracted. Therefore, a well-trained backdoor model will still perform well on clustering data samples in the feature space. As a result, malicious samples are blended into the cluster of its original class and are surrounded by clean samples. To misclassify a malicious sample into another category, the backdoor model essentially reshapes the decision boundary to "wrap around" the malicious sample to include it into the target class, forming small pockets as illustrated in Figure 2(a).

To demonstrate this phenomenon, we conduct an experiment by training a clean model (a 5-layer fully connected neural network) and its corresponding malicious model separately on the Swiss roll dataset [32]. We compare the difference in their decision boundaries. As shown in Figure 3, the clean model is well generalized with a smooth decision boundary. However, the decision boundary of the backdoor model is warped, creating a small pocket that contains the target sample (the area in the yellow circle).

(a) Clean Model. (b) Backdoor Model.

Figure 3. Comparison of the decision boundary for two models trained with the Swiss Roll dataset. In (a) the decision boundary of the clean model is smooth. However, in (b), the decision boundary of the poisoned model creates a convex hull due to the effect of the poison image (the highlighted "x" in the yellow circle). The poison sample is misclassified as class 2 by the backdoor model.

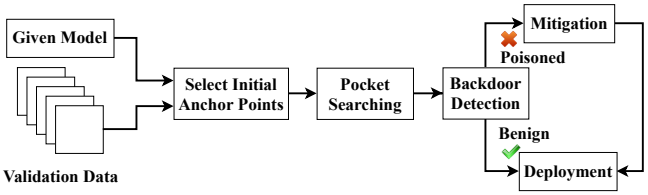


Figure 4. The framework for CLEAR. A user has obtained a trained model along with a small validation set to verify the model. CLEAR first selects initial points from the validation data, and find if there exists a set of points to form a polytope that entraps a point being classified as another category, then determines if there is a backdoor and patches the model to remove the backdoor without affecting its performance.

To this end, we speculate that if we can find and remove those pockets in the feature space, we should be able to remove the backdoor from the model. This observation motivates the proposed approach. More specifically, the overall architecture of CLEAR is illustrated in Figure 4. It consists of three phases as outlined below.

- **Anchor initialization.** To efficiently search for pockets, we first design an algorithm to select the initial anchor points from the validation dataset.
- **Pocket searching.** We take each set of the initial anchor points as a start point to examine if a set of perturbed anchors exists in the original class that can form a polytope entrapping a point being classified as another class. This is achieved by an iterative optimization algorithm.
- **Backdoor Detection and Mitigation.** Based on the probability of finding pockets we identify if there is a sample-targeted backdoor. Then we leverage the generated convex combination to remove the backdoor without affecting the performance on clean data.

4.2. Design and Optimization of Pocket Searching

It is extremely difficult to precisely measure the decision boundary in the feature space, especially when dealing with high dimensional data in complicated neural networks. Therefore, instead of directly looking for pockets on the decision boundary, we approximate their shape by forming small convex hulls. More specifically, we search for a small convex hull whose boundary nodes are from one class and that contains a feature point belonging to another class (see Figures 2(b) and 3(b)). To this end, we design an optimization algorithm to iteratively search for the boundary nodes

Algorithm 1: Pocket Searching Algorithm

Input: Validation data $\mathcal{X}$, number of classes N, number of selected samples n, number of anchor points in a convex set k;
Output: The anchor sets S_p, combined set S_c in the feature space
Let $S_p \leftarrow \{\}$, $S_c \leftarrow \{\}$
for *each source label* $l_s = 0 \rightarrow N$ **do**
  $\mathcal{X}_s \leftarrow$ correctly classified samples $\mathcal{X}$ from class l_s;
  $\{x_s^{(j)}\}_{j=1}^{n} \leftarrow$ select n samples in $\mathcal{X}_s$ with the highest confidence to be classified as class l_s;
  $\mathcal{F} = \{\phi(x_s^{(j)})\}_{j=1}^{n} \leftarrow$ Extract features from the intermediate layer;
  for *each target label* $l_t = 0 \rightarrow N$ *and* $l_t \neq l_s$ **do**
    Sample a set of k initial points from $\mathcal{F}$, denoted as $\{\phi(x_p^{(i)})\}_{i=1}^{k}$;
    For $1 \leq i \leq k$, initialize $c_i = \frac{1}{k}$;
    while $\{\phi(x_p^{(i)})\}_{i=1}^{k}$ *do not converge* **do**
      $\phi(x_c) \leftarrow \sum_{i=1}^{k} c_i \times \phi(x_p^{(i)})$;
      Compute L_p, L_c, and L by Eq. (2)–(4);
      Compute ∇L with regard to $\{\phi(x_p^{(i)})\}_{i=1}^{k}$ and update $\{\phi(x_p^{(i)})\}_{i=1}^{k}$;
      if $f(\phi(x_p^{(i)})) = l_s$ *for all* $1 \leq i \leq k$ *and* $f(\phi(x_c)) = l_t$ **then**
        $S_p \leftarrow S_p \bigcup \{\phi(x_p^{(i)})\}_{i=1}^{k}$;
        $S_c \leftarrow S_c \bigcup \{\phi(x_c)\}$;
      end
    end
  end
end

(anchors) of the convex hull that satisfy the above condition. Algorithm 1 summarizes the overall pocket searching process of CLEAR, which is further elaborated below.

Anchor Initialization. To efficiently find a convex hull, we introduce a simple yet effective algorithm to select the initial anchor points from the validation dataset.

Given a pre-trained model, the distribution of the target sample is unknown since we do not know which label the attacker targets. We enumerate every label to be a hypothetical target label and select the corresponding samples from the validation set to search for the convex hull. For each label l_s, we first feed validation samples into the pre-trained model and record samples being correctly classified. As discussed in Section 4.1, the malicious sample wrapped by a pocket is likely located in the cluster of its original class and surrounded by clean samples. Instead of randomly selecting samples from a given class, we select n samples with the highest confidence to be classified to class l_s from $\mathcal{X}_s$ (i.e., the validation samples labeled as l_s), and extract the output of the n samples at the intermediate layer as their feature $\mathcal{F} = \{\phi(x_s^{(j)})\}_{j=1}^{n}$, where $\phi(\cdot)$ is the feature extractor. If the model has only one fully connected dense layer, we extract the feature before the last convolution block; otherwise, we take the feature from the penultimate layer. In our experiments, we choose $n = 50$.

Pocket Searching. As discussed earlier, we enumerate every class as a hypothetical target class since we do not know which class is the target class. For a given hypothetical target class, the pocket searching is repeated for a number of times with randomly selected initial anchor points. More specifically, we randomly sample k initial anchor points in the feature space from $\mathcal{F}$, denoted as $\{\phi(x_p^{(i)})\}_{i=1}^{k}$, as the starting point to find if there is a set of points in the original class that can form a polytope entrapping a point classified in another class. The value of k is a design parameter, which should be no less than 5 based on our experiments. Otherwise, it may fail to search the correct pocket. In our implementation, we set $k = 5$ by default.

After we select k samples as the initial anchor points $\{\phi(x_p^{(i)})\}_{i=1}^{k}$, their convex combination is represented by $\phi(x_c)$, i.e.,

$$\phi(x_c) = \sum_{i=1}^{k} c_i \times \phi(x_p^{(i)}), \tag{1}$$

where c_i is the convex coefficients, with $c_i \geqslant 0$ and $\sum_{i=1}^{k} c_i = 1$. We try to perturb the anchor points $\{\phi(x_p^{(i)})\}_{i=1}^{k}$ and optimize them towards forming a convex polytope in the feature space, such that a convex combination is created that will lie within the convex polytope and be misclassified as the target class (denoted as l_t). Note that although we can optimize c_i and $\{\phi(x_p^{(i)})\}_{i=1}^{k}$ simultaneously, it is neither efficient nor effective. Instead, we set the coefficients c_i $(1 \leq i \leq k)$ as $\frac{1}{k}$ to enforce the combination

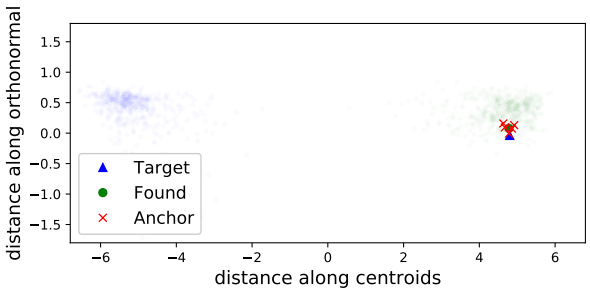


Figure 5. Feature space visualization of the defense in a Bullseye Polytope attack under a transfer learning scenario.

lies in the center of the polytope formed by anchor points in the feature space.

In a nutshell, we formulate and solve the optimization problem as follows. We define

$$L = L_p + L_c + \frac{\alpha}{k} \sum_{i=1}^{k} \left(\left\| \phi(x_p^{(i)}) - \phi(x_c) \right\|^2 \right), \tag{2}$$

where

$$L_p = \frac{1}{k} \sum_{i=1}^{k} (CrossEntropy(f(\phi(x_p^{(i)})), l_s)), \tag{3}$$

and

$$L_c = CrossEntropy(f(\phi(x_c)), l_t). \tag{4}$$

Here, $f(\cdot)$ is the output of the model, and α balances the importance between classification loss and the size of the convex polytope. In Eq. (2), the first term enforces the perturbed anchor points to be still correctly classified; the second term ensures their convex combination to be classified into the target class; and the third term is a constraint that ensures the feature representation of the perturbed anchor points are close to their combination.

SGD [3] is employed to perform the optimization over $\{\phi(x_p^{(i)})\}_{i=1}^{k}$, with the objective to minimize L. If the optimization converges to a convex hull such that when the vertices are in the original class while their combination is classified as the target class, a backdoor is identified. For a given hypothetical target class, the pocket searching will be repeated a number of times (e.g., less than 10 times in our implementation) with randomly selected initial anchor points. Our results show that the optimization can be sensitive to the initiation. However, if there is a backdoor, the optimization will have an extremely high probability to report the target class in less than 10 iterations in all searches.

We enumerate every class to repeat the pocket searching process to examine if it is a target class.

Visualization and Insights. To gain insights into the pocket searching algorithm, we conduct an experiment to visualize the approximate locations of the target sample injected by the attacker and the convex combination point found by the proposed algorithm, all in the feature space. We follow the projection scheme used in [28], where the

x-axis is the direction along the line connecting the centroids of the target and original class features and the y-axis is the component of the parameter vector orthogonal to the vector between the centroids. Figure 5 shows an example of a poisoned DPN92 [5] network under a Bullseye attack in transfer learning (the detailed exp eriment setting can be found in Section 5). The $\triangle$ represents the injected target sample which is within the cluster of the original class samples (green points) in the feature space but being classified as the target class, while $\bullet$ is an example convex combination point found by Algorithm 1, which is close to the target sample. The 'x' marks are the correctly classified anchor points used to generate the combined sample. Also, the distance between the generated sample and the target is much small (within the smallest distance between the target and nearby clean samples in the same class). Thus we suppose the polytope can well approximate the target.

4.3. Backdoor Detection and Mitigation

Based on the pocket searching results, we can perform effective backdoor detection and mitigation. To this end, we define the probability of finding pockets as $P = \frac{N_{found}}{N_{total}}$, where N_{found} is the number of found convex polytopes and N_{total} is the total searches. If P is higher than a threshold, we consider it as a backdoored model. The threshold is set as 50% in our implementation.

To remove the backdoor, we use model patching, i.e., fine-tuning the model with a new dataset, which includes the small validation set (less than 50 samples from each class) and the discovered convex combination points with the original (correct) label (i.e., the label of its corresponding anchors). The fine-tuning process effectively removes the planted backdoor.

5. Experiment Results

In this section, we evaluate the effectiveness of CLEAR against the Label Flipping, Feature Collision, Convex Polytope Attacks, Bullseye Polytope single and multi-target attacks in both transfer learning and end-to-end training scenarios, respectively. For each attack, we adopt all the experimental setups including model architectures and hyperparameters from [28, 38, 2] and conduct the experiments using three benchmarks: CIFAR-10 [18], Multi-View Car Dataset [22] and Mini ImageNet [6] (that randomly selects a subset of 10 classes from ImageNet). We consider sample-targeted backdoor attack models which not only successfully misclassify the target sample(s) to the target category, but also maintain a high classification accuracy on clean training and testing data. Details of each attack are described below.

Dataset and Architecture. We test the Label Flipping and Feature Collision attacks on the CIFAR-10 dataset with two model architectures: ResNet18 [12] and GoogLeNet [31]. For Convex Polytope and Bullseye Polytope single and multi-target attacks, we test them on the CIFAR-10 dataset with 8 model architectures, SENet18 [13], DPN92 [5], GoogLeNet, MobileNetV2 [27], ResNet50, ResNeXt29_2x64d [35], ResNet18, DenseNet121 [14]. For the Bullseye multi-target attack, we test on the Multi-View Car Dataset, which contains images from 20 different cars with 360-degree rotations at increments of 3-4 degrees with all 8 architectures. We also test Label Flipping and Bullseye Polytope single target attack on Mini ImageNet with ResNet18 and VGG19 [29] architectures. In addition to testing in transfer learning, we also test with all these architectures except GoogLeNet[1] in end-to-end training.

Attack configuration. For each attack, we first download a clean model with each network architecture from the official repository or train a benign model with clean training data. We then train backdoored models by poisoning the training dataset using the open-source implementations of each sample-targeted attack method [28, 38, 2]. Specifically, for each clean model, we randomly select 10 different samples (which are now target samples) and different target class for each sample[2]. We plant the backdoor into the model with two settings: transfer learning (finetune the last dense layer) and end-to-end training (finetune all layers). Thus for each attack, we generate 10 backdoored models for each model architecture. In our experiments, the accuracy of the backdoor models drops less than 5% on clean data.

5.1. Backdoor Detection

We test CLEAR on all backdoor models by searching the possible "pockets" between each pair of classes with the SGD solver [3] and an adapted learning rate. Since the scale of the range of the feature for different models is usually different due to the different model architectures, to speed up the pocket searching, we use an adaptive learning rate, i.e., select $lr = 0.001 \times (max\{\phi(\mathcal{X})\} - min\{\phi(\mathcal{X})\})$, where $\mathcal{X}$ is the validation data. We use the detection success rate as the main performance metric, which is the percentage of the malicious models that have been detected with pockets at a certain target class.

Tables 1 and 2 compare the backdoor detection performance of CLEAR against four different attacks on three benchmarks with four state-of-the-art backdoor detection algorithms, including Neural Cleanse [33], GangSweep [39], ABS [20], and STRIP [7]. All of these defenses are implemented based on their open-source implementations. These attacks largely fall into two categories: mislabeled and clean-labeled backdoor attacks. For ImageNet, we select one attack from each category, i.e., the

[1] By [38, 2], it is hard to attack GoogLeNet in end-to-end training.

[2] By [2], for bullseye multi-target attack, we choose target cars with over 90% accuracy on the clean model as target samples.

Table 1. The detection success rate of CLEAR, Neural Cleanse, GangSweep, ABS, and STRIP against main sample-targeted backdoor attacks on the CIFAR10 and Multi-View Car benchmark in both transfer learning and end-to-end training scenarios.

	Label Flipping		Feature Collison		Convex Polytope		Bullseye Polytope Single-target		Bullseye Polytope Multi-target	
	Transfer	End-to-end	Transfer	End-to-end	Transfer	End-to-end	Transfer	End-to-end	Transfer	End-to-end
CLEAR	95.0%	90.0%	100%	90.0%	96.3%	95.7%	97.5%	95.7%	93.8%	87.1%
Neural Cleanse	✖	✖	✖	✖	✖	✖	✖	✖	✖	✖
GangSweep	✖	✖	✖	✖	✖	✖	✖	✖	✖	✖
ABS	✖	✖	✖	✖	✖	✖	✖	✖	✖	✖
STRIP	✖	✖	✖	✖	✖	✖	✖	✖	✖	✖

Table 2. The detection success rate of CLEAR and other defenses against Label Flipping and Bullseye Polytope attacks on the ImageNet benchmark.

Defense Strategy	Label Flipping		Bullseye Polytope Single-target	
	Transfer	End-to-end	Transfer	End-to-end
CLEAR	95%	60%	95%	80%
Neural Cleanse	✖	✖	✖	✖
GangSweep	✖	✖	✖	✖
ABS	✖	✖	✖	✖
STRIP	✖	✖	✖	✖

Label Flipping and Bullseye Polytope single-target attacks. We use "✖" to represent that none of the malicious models have been detected by the detection algorithm. As shown in Tables 1 and 2, Neural Cleanse and GangSweep fail to detect any sample-targeted backdoored model, as there does not exist a trigger for them to reverse-engineer for backdoor detection. Similarly, ABS cannot detect any sample-targeted backdoored model due to the negligible increase of the maximum activation value in the hidden layers. In addition to that, online detection schemes, e.g., STRIP, also failed on identifing malicious samples as these samples are drawn from the same distribution of the clean images.

In contrast, CLEAR can successfully detect most of the sample-targeted backdoor models with over 93% detection success rate in the transfer learning setting. For the end-to-end training scenario, since the attacker finetunes the entire model including the feature extractor, the feature of the target sample may move out of the cluster of the original class, thus resulting in slightly degraded detection rate.

Computational Efficiency. To evaluate the efficiency of CLEAR in pocket searching, we run it on an Nvidia RTX2080 Mobile Max-Q GPU with 8GB memory. CLEAR takes less than 1 second to search for a backdoor pocket from a set of initial points in the feature space. Besides, we set the bound of the searching space as $[min\{\phi(\mathcal{X})\}, max\{\phi(\mathcal{X})\}]$. Once the combined points are out of range, the search will be terminated.

5.2. Backdoor Mitigation

For an identified backdoor model, we patch it by finetuning the model for 5 epochs with a new training set that includes both a small set of clean validation data (50 samples) and the discovered pocket samples. The pocket samples are generated based on the clean validation data in the pocket searching phase and their labels are corrected to the class of the anchor points. We train the last linear layer with the Adam optimizer [16] at a learning rate of 0.1.

We evaluate the performance of CLEAR for backdoor mitigation with two metrics: the Attack Success Rate (ASR), which is the percentage of backdoor models that still misclassify the target sample to the target label; and the testing accuracy (Acc) which represents the model's accuracy on clean images (these test samples are not in the training set or validation set). For the single target attack, if the target sample is correctly classified to its original class label, we consider that the backdoor has been removed. For the multi-target attack, since there are more than one target sample, we consider the backdoor has been moved if over 90% of the target object images are classified to their original class label. Clearly, the ASR of all poisoned models is 100% since they are all backdoored. A good mitigation approach should significantly lower ASR.

Tables 3 and 4 show the classification accuracy of the testing set and the ASR of the backdoored models under different attacks in transfer learning and end-to-end training settings before and after patching on the CIFAR-10 benchmark. After patching with generated pocket samples, over 90% of backdoored models are protected against all the attack approaches (described earlier), without significantly sacrificing the classification accuracy of the testing samples. This also shows that even if we mis-detect a benign model as a possible malicious model, finetuning with the generated pocket samples would not have much side impact on the model. An important observation is that the identified malicious samples are critical for removing the backdoor. This can be seen that in Tables 3 and 4, patching with clean samples cannot clean most multi-target backdoored models in both transfer learning and end-to-end training. As illustrated in Table 5, for the large-scale ImageNet benchmark, our approach of using generated pocket samples can also remove almost all the poisoned models under the Label Flipping and Bullseye Polytopes Single-target attacks, while patching with clean samples fails.

In addition, we evaluate the effectiveness of Fine-pruning [19] on mitigating the backdoored models by re-

Table 3. Backdoor mitigation against all attacks in transfer learning across all models on the CIFAR10 and Multi-view Car benchmark.

Defense Strategy	Label Flipping		Feature Collision		Convex Polytope		Bullseye Polytope Single-target		Bullseye Polytope Multi-target	
	Test Acc	ASR	Test Acc	ASR	Test Acc	ASR	Test Acc	ASR	Test Acc	ASR
Poisoned Model	$94.7\% \pm 1\%$	100%	$94.5\% \pm 1\%$	100%	$94.2\% \pm 1\%$	100%	$91.4\% \pm 2\%$	100%	$90.3\% \pm 2\%$	100%
CLEAR patch with generated samples	$94.9\% \pm 1\%$	**10.0%**	$94.0\% \pm 1\%$	**0%**	$95.1\% \pm 1\%$	**3.8%**	$91.3\% \pm 2\%$	**5.0%**	$90.4\% \pm 2\%$	**8.8%**
Patch with clean samples	$94.8\% \pm 1\%$	45.0%	$94.7\% \pm 1\%$	5.0%	$95.2\% \pm 1\%$	28.8%	$92.1\% \pm 1\%$	23.8%	$91.5\% \pm 1\%$	90.0%
Fine-pruning	$94.9\% \pm 1\%$	70.0%	$93.9\% \pm 1\%$	100%	$94.3\% \pm 2\%$	51.25%	$91.7\% \pm 1\%$	61.25%	$91.1\% \pm 1\%$	100%

Table 4. Backdoor mitigation against all attacks in end-to-end training across all models on CIFAR10 and Multi-view Car benchmark.

Defense Strategy	Label Flipping		Feature Collision		Convex Polytope		Bullseye Polytope Single-target		Bullseye Polytope Multi-target	
	Test Acc	ASR	Test Acc	ASR	Test Acc	ASR	Test Acc	ASR	Test Acc	ASR
Poisoned Model	$94.9\% \pm 1\%$	100%	$93.4\% \pm 1\%$	100%	$92.1\% \pm 1\%$	100%	$92.1\% \pm 1\%$	100%	$88.2\% \pm 1\%$	100%
CLEAR patch with generated samples	$93.9\% \pm 1\%$	**40.0%**	$93.8\% \pm 2\%$	**5.0%**	$91.7\% \pm 1\%$	**7.1%**	$91.4\% \pm 1\%$	**8.6%**	$89.1\% \pm 2\%$	**15.7%**
Patch with clean samples	$95.0\% \pm 1\%$	100%	$93.9\% \pm 1\%$	20.0%	$92.1\% \pm 1\%$	45.7%	$92.4\% \pm 1\%$	48.6%	$88.5\% \pm 1\%$	92.8%
Fine-pruning	$94.7\% \pm 1\%$	100%	$92.8\% \pm 1\%$	100%	$91.8\% \pm 1\%$	100%	$91.3\% \pm 1\%$	100%	$88.2\% \pm 1\%$	100%

Table 5. Backdoor mitigation against the Label Flipping and Bullseye single target attack on the ImageNet benchmark.

Defense Strategy	Label Flipping				Bullseye Polytope Single-target			
	Transfer		End-to-End		Transfer		End-to-End	
	Test Acc	ASR	Test Acc	ASR	Test Acc	ASR	Test Acc	ASR
Poisoned Model	$97.3\% \pm 1\%$	100%	$97.4\% \pm 1\%$	100%	$96.9\% \pm 1\%$	100%	$97.2\% \pm 1\%$	100%
CLEAR	$97.1\% \pm 1\%$	**0%**	$96.9\% \pm 1\%$	**20%**	$97.0\% \pm 1\%$	**5%**	$96.1\% \pm 2\%$	**20%**
Patch with clean samples	$97.4\% \pm 1\%$	95%	$97.0\% \pm 1\%$	100%	$96.9\% \pm 1\%$	75%	$96.8\% \pm 1\%$	95%
Fine-pruning	$96.4\% \pm 1\%$	90%	$96.8\% \pm 1\%$	100%	$95.0\% \pm 2\%$	85%	$95.1\% \pm 1\%$	100%

moving redundant neurons of the last convolution layer. As shown in Tables 3-5, it fails to remove any sample-targeted backdoor in end-to-end scenarios, and can remove up to 48.75% of the target samples in the transfer learning setting. It is observed that, if the logits of the target sample (i.e., the output of model before the softmax layer) at the target category is only slightly higher than that of the source category, the misclassification of the target sample has a high probability of being corrected after finetuning with clean samples. However, if the target sample is misclassified with a high confidence, the backdoor is difficult to be removed by finetuning with limited clean samples.

5.3. Adaptive Attack

An adaptive attack assumes that the attacker is aware of CLEAR and tries to deliberately evade it. To succeed in the attack formulated in Section 3, it is inevitable to reshape the decision boundary to "wrap around" the target sample, forming a small pocket, in order to achieve the desired misclassification into the target class. The only detour is to directly map the target feature into the target class. This however would result in unacceptable classification errors on normal samples, which is very suspicious. As a result, the attack would have been detected. Therefore, even if the attacker understands the defense mechanism, it is fundamentally challenging to construct effective adaptive attacks.

6. Conclusion

In this work, we have proposed the first detection and mitigation scheme to address sample-targeted backdoor attacks. We have revealed and demonstrated that the boundary change caused by the sample-targeted backdoor forms small "pockets" around the target sample. Based on this observation, we have proposed a novel defense mechanism to pinpoint malicious pockets by "wrapping" them into a tight convex hull in the feature space. We have designed an algorithm to search for such a convex hull. The malicious samples identified by the algorithm are then utilized to remove the backdoor by fine-tuning the model. Compared to the previous backdoor detection solutions, the proposed approach is highly effective for detecting and mitigating a wide range of sample-targeted backdoor models under different benchmark datasets.

7. Acknowledgments

This work was supported in part by NSF under Grants CNS-2120279, CNS-1950704, CNS-1828593, and OAC-1829771, ONR under Grant N00014-20-1-2065, NSA under Grant H98230-21-1-0278, DoD Center of Excellence in AI and Machine Learning (CoE-AIML) under Contract Number W911NF-20-2-0277, the Commonwealth Cyber Initiative, and InterDigital Communications, Inc.

References

[1] Jing Yu Koh. ModelZoo: Discover open source deep learning code and pretrained models. `http://www.modelzoo.co`.

[2] H. Aghakhani, Dongyu Meng, Yu-Xiang Wang, C. Krügel, and G. Vigna. Bullseye polytope: A scalable clean-label poisoning attack with improved transferability. *ArXiv*, abs/2005.00191, 2020.

[3] Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

[4] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *The Thirty-Third AAAI Conference on Artificial Intelligence Safety Workshop*, 2019.

[5] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. In *Proceedings of Advances in Neural Information Processing Systems(NeurIPS)*, pages 4467–4475, 2017.

[6] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of 2009 IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 248–255, 2009.

[7] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C. Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the Annual Computer Security Applications Conference(ACSAC)*, page 113–125, 2019.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of Advances in Neural Information Processing Systems(NeurIPS)*, pages 2672–2680, 2014.

[9] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, pages 47230–47244, 2019.

[10] Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. *arXiv preprint arXiv:1908.01763*, 2019.

[11] Neal Gupta, W. Ronny Huang, Liam Fowl, Chen Zhu, Soheil Feizi, Tom Goldstein, and John P. Dickerson. Strong baseline defenses against clean-label poisoning attacks. *CoRR*, abs/1909.13374, 2019.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, pages 770–778, 2016.

[13] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, pages 7132–7141, 2018.

[14] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, pages 4700–4708, 2017.

[15] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM international conference on Multimedia(ACM-MM)*, pages 675–678, 2014.

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations(ICLR)*, 2015.

[17] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1885–1894, 2017.

[18] Yann LeCun, LD Jackel, Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, Urs A Muller, Eduard Sackinger, Patrice Simard, et al. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, page 276, 1995.

[19] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *Research in Attacks, Intrusions, and Defenses*, pages 273–294. Springer International Publishing, 2018.

[20] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security(CCS)*, page 1265–1282, 2019.

[21] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *Proceedings of the Annual Network and Distributed System Security Symposium (NDSS)*, 2018.

[22] M. Ozuysal, V. Lepetit, and P. Fua. Pose estimation for category specific multiview object localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 778–785, 2009.

[23] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 41.1–41.12, September 2015.

[24] Ximing Qiao, Yukun Yang, and Hai Li. Defending neural backdoors via generative distribution modeling. In *Proceedings of Advances in Neural Information Processing Systems(NeurIPS)*, pages 14004–14013, 2019.

[25] Mauro Ribeiro, Katarina Grolinger, and Miriam AM Capretz. Mlaas: Machine learning as a service. In *Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 896–902, 2015.

[26] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden trigger backdoor attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 11957–11965, 2020.

[27] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, pages 4510–4520, 2018.

[28] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Proceedings of the Advances in Neural Information Processing Systems(NeurIPS)*, pages 6103–6113, 2018.

[29] K. Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[30] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, page 3520–3532, 2017.

[31] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, pages 1–9, 2015.

[32] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[33] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, pages 707–723, 2019.

[34] Huang Xiao, Battista Biggio, Blaine Nelson, Han Xiao, Claudia Eckert, and Fabio Roli. Support vector machines under adversarial label contamination. *Neurocomputing*, 160:53 – 62, 2015.

[35] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, pages 1492–1500, 2017.

[36] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. Latent backdoor attacks on deep neural networks. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security(CCS)*, pages 2041–2055, 2019.

[37] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020.

[38] Chen Zhu, W. Ronny Huang, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. Transferable clean-label poisoning attacks on deep neural nets. In *Proceedings of Machine Learning Research(PMLR)*, volume 97, pages 7614–7623, 2019.

[39] Liuwan Zhu, Rui Ning, Cong Wang, Chunsheng Xin, and Hongyi Wu. Gangsweep: Sweep out neural backdoors by gan. In *Proceedings of the ACM International Conference on Multimedia(ACM-MM)*, page 3173–3181, 2020.