Complex Backdoor Detection by Symmetric Feature Differencing

Yingqi Liu[1]*, Guangyu Shen[1]*, Guanhong Tao[1], Zhenting Wang[2], Shiqing Ma[2], Xiangyu Zhang[1]
Purdue University[1], Rutgers University[2]
{liu1751, shen447, taog}@purdue.edu, {zhenting.wang, sm2283}@rutgers.edu, xyzhang@cs.purdue.edu

Abstract

Many existing backdoor scanners work by finding a small and fixed trigger. However, advanced attacks have large and pervasive triggers, rendering existing scanners less effective. We develop a new detection method. It first uses a trigger inversion technique to generate triggers, namely, universal input patterns flipping victim class samples to a target class. It then checks if any such trigger is composed of features that are not natural distinctive features between the victim and target classes. It is based on a novel symmetric feature differencing method that identifies features separating two sets of samples (e.g., from two respective classes). We evaluate the technique on a number of advanced attacks including composite attack, reflection attack, hidden attack, filter attack, and also on the traditional patch attack. The evaluation is on thousands of models, including both clean and trojaned models, with various architectures. We compare with three state-of-the-art scanners. Our technique can achieve 80-88% accuracy while the baselines can only achieve 50-70% on complex attacks. Our results on the TrojAI competition rounds 2-4, which have patch backdoors and filter backdoors, show that existing scanners may produce hundreds of false positives (i.e., clean models recognized as trojaned), while our technique removes 78-100% of them with a small increase of false negatives by 0-30%, leading to 17-41% overall accuracy improvement. This allows us to achieve top performance on the leaderboard.

1. Introduction

Backdoor attack (or trojan attack) on deep learning models injects malicious behaviors such that a compromised model behaves normally on clean inputs and misclassifies inputs stamped with a *trigger* to a *target label* [6, 20, 36, 40, 41]. It becomes a prominent threat due to the low complexity of launching such attacks, the devastating consequences especially in safety/security critical applications, and the difficulty of defense.

*Equal contribution

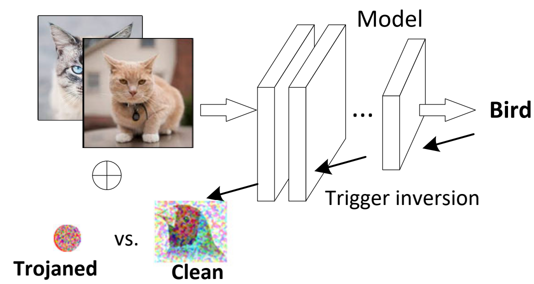


Figure 1. Backdoor detection by trigger inversion

There are a body of existing defense techniques such as trigger inversion [39, 66], attribution analysis [16, 25], trojaned input detection [14, 64], backdoor removal [34, 37], and model certification [65]. According to [3, 48], trigger inversion is an effective technique that can determine if a model contains backdoor without assuming the availability of any input with trigger. For example, Neural Cleanse (NC) [66], Artificial Brain Stimulation (ABS) [39], and K-Arm [55] make use of optimization to invert triggers and determine if a model is trojaned. They consider each label in the model as a potential target and use optimization to check if a small and fixed input pattern, i.e., a trigger, can be found to cause any input to be misclassified to the label. The intuition is that attackers tend to use small triggers for attack stealthiness. Figure 1 illustrates trigger inversion. An input pattern (the circular pattern or the rectangular one on the bottom) is generated by gradient back-propagation to flip cat samples to bird. If the subject model is clean, a *large* pattern that exhibits a lot of bird features is generated (e.g., the rectangular pattern with the "`clean`" tag). In contrast, when the model is trojaned (with a red circular patch), a *small* pattern containing the trigger features is inverted (e.g., the circular pattern with the "`trojaned`" tag). The size difference of the patterns is critical for these scanners, that is, a model is flagged as trojaned only when a small trigger can be found. Observe that inverted patterns are usually noisy and may not be human interpretable.

While existing techniques are effective for attacks with small and static triggers, advanced attacks proposed re-

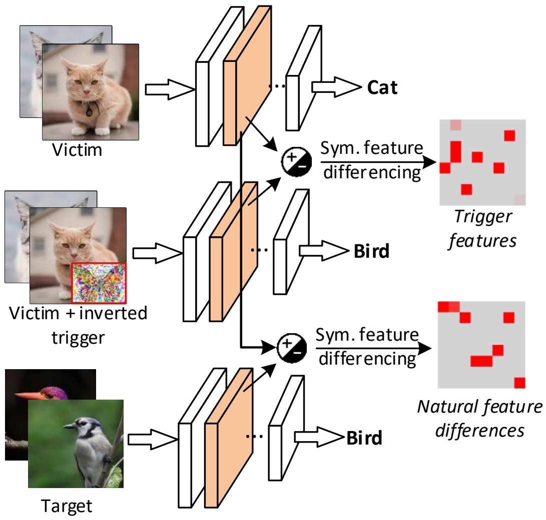


Figure 2. Ex-ray overview

cently [36, 41, 52] have large and dynamic triggers: the input level differences before and after injecting a trigger are substantial and the differences vary across different inputs. Composite attack [36] injects a backdoor by mixing benign features from two or more classes. For example, a butterfly appearing in a cat image causes the model to predict bird. Triggers may be large (e.g., a butterfly could be much larger than a typical patch trigger) and have different pixel level manifestations (e.g., various kinds of butterfly). Reflection backdoor [41] uses the reflection of an image as the trigger. Reflection occurs when pictures are taken behind a glass window. Reflection could be as large as a whole image. Hidden trigger backdoor [52] introduces perturbations on the training images of target label such that the perturbed images have similar inner activations to the trigger and forces the model to learn the correlations between the trigger and the target label. Since the trigger is not explicit in the training, the trojaning process is more difficult and requires larger triggers. Existing scanners such as NC, ABS, and K-arm have difficulties detecting these backdoors. They only achieve 0.5-0.7 accuracy (see the evaluation section).

Essence of Backdoor Attack. We observe the essence of backdoor attack is that *model misclassification (to the target class) is induced by features not in the target class.* For example, in a composite attack, a cat image is misclassified to bird when a butterfly is also present in the image. The misclassification is essentially induced by the features of cat and butterfly (not bird's). Our overarching idea is hence to determine if the features of an inverted trigger are the natural features distinguishing the victim and target classes. If so, the model is clean. Otherwise, it is considered trojaned. Note that in our method small sizes and fixed triggers are no longer essential properties. One may argue that the attacker could craft a trigger with the target class's natural features. We will discuss such an adaptive attack in Section 4.3.

Our Method. Figure 2 illustrates our method. It takes a model and a small set of clean samples (e.g., 10 for each class). It first leverages an existing trigger inversion method to derive a trigger that can flip a set of clean samples of the victim class (cat) to the target class (bird). It feeds a set of clean victim samples to the model, and extracts the internal feature maps at a selected layer (the first row in the figure). It then injects the inverted trigger (the butterfly-like pattern) to the clean victim samples and extracts the corresponding feature maps (the second row). A novel feature comparison technique called *symmetric feature differencing* (SFD) is then applied to the two sets of feature maps, to determine the distinctive features between the two sets of samples (the first rectangular map on the right with red cells), which essentially denotes the trigger features. The map is also called a *feature difference mask* or just *mask*. A red cell in the mask indicates a feature map that is distinctive. It further feeds a set of clean target class (bird) samples to the model and extracts the feature maps (the third row). Applying SFD to the target class and the victim class feature maps yields the natural features distinguishing the two classes, that is, the second mask on the right. A model is considered trojaned when the two masks do not have similarity.

The key enabling technique of our method is SFD, which is a novel differential analysis. It is based on *counter-factual causality* [33]. Given two sets of samples, like the victim and victim+trigger samples mentioned above, it computes a smallest set of feature maps such that exchanging their activation values across the two sets entails exchanged classification results. They are considered the distinctive features. The formal definition and the computation algorithm can be found in Section 3.

Our contributions are summarized as follows.

- We develop a new scanning technique that can detect large and complex backdoors which are difficult for existing techniques.
- The technique is based on a novel symmetric feature differencing method that can identify the distinctive features of two sets of given examples.
- We implement a prototype EX-RAY("*DEtecting CompleX BackdooR in NeurAl Networks by SYmmetric Feature Differencing*"). It is general and can leverage different upstream trigger inversion methods. EX-RAY is publicly available at `https://github.com/PurduePAML/Exray`
- We evaluate EX-RAY on 4246 models (2081 benign and 2165 trojaned) with 23 structures and 7 datasets, and four attacks that have large/pervasive and dynamic triggers (reflection, composite, hidden, and filter attacks). We compare with three state-of-the-art trigger inversion based scanners, NC, ABS, and K-Arm. Our results show that EX-RAY can achieve 80-88% accuracy while the baselines can only achieve 50-70%.

We also use model interpretation techniques to show that EX-RAY indeed captures natural feature differences between classes. EX-RAY can also be used to remove false positives in backdoor scanning (i.e., clean models are considered trojaned), which are usually due to small triggers found between clean labels. EX-RAY can determine that such triggers essentially denote natural features of the target label and should be precluded. We test EX-RAY (with ABS as the upstream inversion technique) on TrojAI[1] rounds 2-4 and show that EX-RAY can reduce false warnings by 78-100%, with the cost of a small increase in false negatives (0-30%), i.e., trojaned models are considered clean. It can improve multiple upstream scanners' overall accuracy including ABS (by 17-41%), NC (by 25%), and the Bottom-up-Top-down backdoor scanner [1] (by 2-15%) in the competition. Our method also outperforms a number of other false positive removal methods that compare L2 distances and leverage attribution/interpretation techniques. EX-RAY will be released upon publication.

- On the TrojAI leaderboard, ABS+EX-RAY achieves top performance in 2 out of the 4 rounds for image classification, including the most challenging round 4, with an average cross-entropy (CE) loss around 0.32[2] and an average AUC-ROC[3] around 0.90. It is the only technique that successfully reached the round target (for both the training sets and the test sets remotely evaluated by IARPA), i.e., a CE loss lower than 0.3465, for all the 4 rounds. As far as we know, a large number of state-of-the-art scanning techniques have been evaluated in the competition, including NC [66], ABS [39], Meta neural analysis [73], ULP [29], DeepInspect [11], SCAn [60], K-Arm backdoor scanning [55], Noise analysis backdoor detection [16] and attribution based backdoor detection [25, 57].

Threat Model. Our threat model is consistent with that in existing works [3, 39]. Given a set of models, including both trojaned and clean models, and a small set of clean samples for each model (covering all labels), we aim to identify the models with injected backdoor(s) that can flip clean samples to the target class. These samples may belong to one or many victim class(es). The former is label-specific attack and the latter is universal attack. □

[1] TrojAI is a backdoor scanning competition organized by IARPA [3]. Rounds 1-4 are for image classification. Round 1 dataset is excluded due to simplicity.

[2] The smaller the better.

[3] An accuracy metric used by TrojAI, the larger the better.

2. Related Work

Backdoor Attack. Data poisoning [12, 20] injects backdoors by changing the label of inputs with trigger. Clean label attack [52, 54, 63, 75] injects backdoors without changing the data label. Dynamic backdoor [46, 53] focuses on crafting different triggers for different inputs and breaks the defense's assumption that trigger is static. [47] proposes to combine adversarial example generation and model poisoning. There are also attacks on NLP tasks [13, 31, 74], reinforcement learning [28, 68], and federated learning [7, 17, 61, 67, 72]. EX-RAY is a general primitive that may be of use in defending these attacks.

Defenses against Backdoor Attacks. ULP [29] and Meta neural analysis [73] train a few input patterns and a classifier from thousands of benign and trojaned models. The classifier predicts if a model has backdoor based on activations of the patterns. [51] proposes to reverse engineer the distribution of triggers. [23] finds that trojaned and clean models react differently to input perturbations. TABOR [21] and NeuronInspect [24] use an AI explanation technique to detect backdoor. There are techniques that defend backdoors by data sanitization [9, 50]. There are also techniques that detect if a given input is stamped with a trigger [10, 14, 15, 18, 19, 22, 35, 38, 42, 60, 62, 64]. They target a different problem as they require inputs with embedded triggers. EX-RAY is orthogonal to most of these techniques and can serve as a performance booster.

Interpretation/Attribution. EX-RAY is related to model interpretation and attribution, e.g., important features identification [5, 8, 56, 59]. [26] measures the importance of a concept (e.g., 'striped') for a class (e.g., zebra). The differences lie in that EX-RAY finds distinguishing features of two sets of examples.

3. Design

As illustrated by Figure 2, given a trigger t inverted by some upstream scanning technique (not our contribution) that flips victim class V samples to the target class T, EX-RAY first computes the distinctive features between V and V+t samples, then the distinctive features between V and T. Finally, it uses a similarity analysis to compare the two sets of distinctive features to determine if the trigger denotes natural differences between the two classes. If not, the model is considered trojaned. In this section, we explain the steps in details.

3.1. Symmetric Feature Differencing

The key enabling technique of EX-RAY is *symmetric feature differencing* (SFD) that determines the distinguishing features between two sets of examples (e.g., from classes V and T). SFD is based on *counter-factual causality* [33], which states that an effect event e causally depends on a cause event c if and only if, 1) if c were to occur e would occur; and 2) if c were not to occur e would not occur. In our context, we say a set of features are distinctive between two sets of examples if and only if 1) *exchanging these features across the two sets (event c) entails exchanged classification results (event e), and 2) the exchange of any such feature is necessary to the exchanged classification results*. For example, we say two sets of examples from two respective persons A and B in a face recognition model differ only at their nose if and only if 1) replacing the nose in the examples of A with B's nose causes the model to predict B, 2) replacing the nose is needed to cause misclassification, and vice versa. Note that although replacing both nose and mouth can also induce exchanged classifications, replacing mouth is not necessary. Hence mouth is not a distinctive feature. Automatically identifying such feature differences in the input space is challenging due to the difficulty of locating features, as a feature may manifest itself differently across input examples. Our differencing method hence identifies a set of neurons (i.e., feature maps) at some hidden layer that denote the distinctive features.

We formally define symmetric feature differencing in the following. To simplify our discussion, we assume the technique takes a subject model $F(x)$ and two inputs: x_v in V and x_t in T (instead of two sets of inputs). We will discuss the extension to two sets later in the section.

Definition 1 (Symmetric Feature Differencing) *Let $F(x)$ be a feed forward neural network. Given an inner layer l that provides good feature abstraction, let g be the submodel up to layer l and h the submodel after l, i.e., $F(x) = h(g(x))$. Let the number of features/neurons at l be n.* Symmetric feature differencing *(SFD) computes a mask M that is an n element vector with values 0 or 1. Let $\neg M$ be the negation of the mask such that $\neg M[i] = 1 - M[i]$ with $i \in [1, n]$.*

The mask M satisfies the following conditions.

$$h(g(x_v) \cdot M + g(x_t) \cdot \neg M) = V \tag{1}$$

$$h(g(x_v) \cdot \neg M + g(x_t) \cdot M) = T \tag{2}$$

$$\|M\|_0 \textit{ is minimal.} \tag{3}$$

Intuitively, Equation (1) denotes that copying x_v's features to x_t with the control of M causes the classification of V. Specifically, $g(x_v) \cdot M + g(x_t) \cdot \neg M$ means that when $M[i] = 0$ (i.e., $\neg M[i] = 1$), the original ith feature map of the T sample x_t is retained; when $M[i] = 1$, the ith feature map of x_t is replaced with that from the V sample x_v. The explanation for Equation (2) is similar. Equation (3) dictates the minimality of M, that is, any feature exchange indicated by the mask is necessary, faithfully following the counter-factual causality definition.

The SFD definition is graphically illustrated by an example in Figure 3. The dash box on the left shows the $g(x)$ function and that on the right the $h(\cdot)$ function. The top row in the left box shows that five feature maps (in yellow) are generated by $g()$ for a victim class sample x_v. The bottom row shows that the five feature maps (in blue) for a target class sample x_t. The dash box in the middle illustrates symmetric differencing. As suggested by the red entries in the mask M in the middle (i.e., $M[3] = M[5] = 1$), in the top row, the 3rd and 5th (yellow) feature maps are replaced with the corresponding (blue) feature maps from the bottom. Symmetric replacements happen in the bottom row as well. On the right, the exchanged feature maps cause the exchanged classification results.

Note that a minimal M must exist. In the worst case, M is filled with '1', indicating all feature maps shall be exchanged, which must yield the exchanged classification results. In general, the complexity of computing M is exponential. We hence propose a soft version of SFD.

Soft Symmetric Feature Differencing. In the soft version, we relax the meaning of mask. Instead of having an either 0 or 1 value, we allow the value to vary in $[0, 1]$, with 0 meaning no-exchange at all, 1 meaning complete exchange, and a value in between 0 and 1 partial exchange. For example, assume nose, eyes, and mouth are the three features in a face recognition model and assume $M[nose] = 1$, $M[eyes] = 0$, and $M[mouth] = 0.5$. The mask means that exchanging noses, retaining eyes, and mixing mouths half-half. The need of mixing mouths means that sometimes exchanging noses alone may not be sufficient to cause exchanged results and we need to partially consider their mouths which have a certain level of difference as well.

With the relaxation, the exchange operations, i.e., Equations (1) and (2), become continuous and differentiable. In addition, the minimality requirement in Equation (3) can be modeled by a differentiable arg min operation that minimizes the size of mask. Specifically, it can be reduced to the following constrained optimization problem.

$$\begin{aligned} &\arg\min_M \; sum(M), s.t. \\ &h(g(x_v) \cdot M + g(x_t) \cdot \neg M) = V \; and \\ &h(g(x_v) \cdot \neg M + g(x_t) \cdot M) = T \end{aligned} \tag{4}$$

To solve the problem, we devise a loss in (5). It has three parts. The first part $sum(M)$ is to minimize the mask size. The second part $w_1 \times ce_1$ is a barrier loss for Equation (1), with ce_1 the cross entropy loss when replacing x_t's features. Coefficients w_1 is dynamic. When the cross entropy loss is larger than a threshold α, w_1 is set to a large value w_{large}. This forces M to satisfy Equation (1). When the loss is small indicating the constraint is satisfied, w_1 is changed to

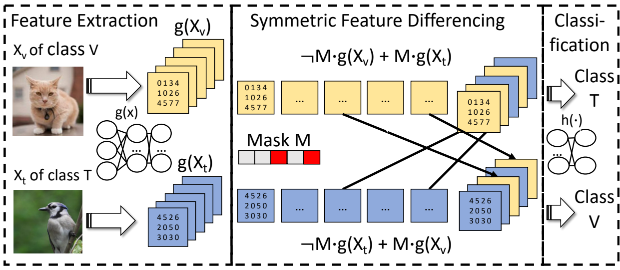


Figure 3. Illustrating symmetric feature differencing

a small value w_{small}. The optimization hence focuses on minimizing the mask. The third part $w_2 \times ce_2$ is similar.

$$\begin{aligned} \mathcal{L}_{pair}(x_v, x_t) &= sum(M) + w_1 \times ce_1 + w_2 \times ce_2, \\ with\ ce_1 &= CE(h(g(x_v) \cdot M + g(x_t) \cdot \neg M), V), \\ ce_2 &= CE(h(g(x_v) \cdot \neg M + g(x_t) \cdot M), T), \\ w_1 &= w_{large} \text{ if } ce_1 > \alpha \text{ else } w_{small}, \\ w_2 &= w_{large} \text{ if } ce2 > \alpha \text{ else } w_{small} \end{aligned} \tag{5}$$

Differencing Two Sets. The algorithm to identify the differential features of two sets can be built from the primitive of comparing two inputs. Given two sets X_V of class V and X_T of class T, ideally the mask M should satisfy Equations (1) and (2) for any $x_v \in X_V$ and $x_t \in X_T$. While such a mask must exist (with the worst case containing all the features), minimizing it becomes very costly. Assume $|X_V| = |X_T| = m$. The number of constraints that need to be satisfied during optimization is $O(m^2)$. Therefore, we develop a stochastic method that is $O(m)$. Specifically, let $\overrightarrow{X_V}$ and $\overrightarrow{X_T}$ be random orders of X_V and X_T, respectively. We minimize M such that it satisfies Equations (1) and (2) for all pairs $(\overrightarrow{X_V}[j], \overrightarrow{X_V}[j])$, with $j \in [1, m]$. Intuitively, we optimize on a set of random pairs from X_V and X_T that cover all the individual samples in X_V and X_T. The loss function is hence the following.

$$\mathcal{L} = \sum_{j=1}^{m} \mathcal{L}_{pair}(\overrightarrow{X_V}[j], \overrightarrow{X_T}[j])$$

When X_V and X_T have one-to-one mapping, such as the victim class samples and their compromised versions that have the trigger injected, we can directly use the mapping in optimization instead of a random mapping. We use Adam optimizer [27] with a learning rate 5e-2 and 400 epochs. Masks are initialized to all 1 to begin with. This denotes a conservative start since such masks suggest swapping all feature maps, which must induce the intended classification results swap.

Symmetry Is Necessary. Our technique is symmetric. Such symmetry is critical to effectiveness. One may wonder that a one-sided analysis that only enforces Equation (2) may be sufficient. That is, M is the minimal set of features that when copied from T (target) samples to V (victim) samples can flip the V samples to class T. Intuitively, it denotes the strong features of T. However, this is insufficient. In many cases, misclassification (of a V sample to T) in a clean model can be induced when strong features of class V are suppressed (instead of adding strong T features). As such, an inversion method may generate a trigger that neutralizes V features instead of injecting unique T features. The trigger features hence do not share much commonality with the features computed by the one-sided analysis. Consequently, the clean model is considered trojaned. Our experiments in Appendix A and H show the importance of symmetry.

3.2. Comparing Masks

After generating the masks, we compare them in the last step. Let the distinguishing features between the victim and target classes be M_1, and then those between the victim samples and their compromised versions be M_2. Next, we explain how to compare M_1 and M_2. Intuitively M_1 and M_2 should share a lot of commonality when the trigger denotes natural differences between classes, as reflected in the following condition.

$$sum(min(M_1, M_2)) > \beta \times min(sum(M_1), sum(M_2)) \tag{6}$$

Here, $min(M_1, M_2)$ yields a vector whose elements are the minimal between the corresponding elements in M_1 and M_2. It essentially represents the intersection of the two masks. The hyperparameter β is in $(0, 1)$. Intuitively, the condition asserts that the size of mask intersection is larger than β times the minimum size of the two masks, meaning the two have a large part in common. If all the inverted triggers for a model satisfy the condition, the model is considered clean, otherwise trojaned.

Additional Validation Checks. In practice, due to the uncertainty in the stochastic symmetric differencing algorithm, the presence of local minimums in optimization, and the small number of available clean samples, M_1 and M_2 may not have a lot in common. However, they should nonetheless satisfy the semantic constraint that both should denote natural feature differences of the victim and target classes if the trigger is not injected. Therefore, we propose an additional cross-validation check that tests if functionally M_1 and M_2 can take each other's place in satisfying Equations (1) and (2). In particular, while M_1 is derived by comparing the victim class and the target class clean samples, we copy the feature maps indicated by M_1 between the victim samples and their compromised versions with trigger and check if the intended class flipping can be induced; similarly, while M_2 is derived by comparing the victim class samples and their compromised versions, we copy the feature maps indicated by M_2 between the victim clean samples and the target clean samples to see if the intended class flipping can be induced. If so, the two are functionally similar and the trigger is natural. The check is formulated as follows.

$$\begin{aligned}
&Acc(h(g(X_V) \cdot M_2 + g(X_T) \cdot \neg M_2), V) > \gamma \wedge \\
&Acc(h(g(X_T) \cdot M_2 + g(X_V) \cdot \neg M_2), T) > \gamma \wedge \\
&Acc(h(g(X_V) \cdot M_1 + g(X_V + t) \cdot \neg M_1), V) > \gamma \wedge \\
&Acc(h(g(X_V + t) \cdot M_1 + g(X_V) \cdot \neg M_1), T) > \gamma
\end{aligned} \tag{7}$$

Here, $Acc()$ is a function to evaluate prediction accuracy on a set of samples and γ a threshold (0.8 in the paper). We use $g(X_V)$ to denote applying g on each sample in X_V for representation simplicity.

4. Evaluation

We evaluate EX-RAY on a range of backdoor attacks including four complex backdoors (i.e., composite attack, reflection attack, hidden attack, and filter attack), and on the traditional patch attack. We also study the applicability of EX-RAY to various upstream scanners by boosting their backdoor detection performance. We demonstrate that EX-RAY can be leveraged to fix models with injected and natural backdoors. In addition, we validate that the generated masks by EX-RAY is capable of capturing feature differences. We devise two adaptive attacks targeting the basis of EX-RAY. EX-RAY is implemented in PyTorch [49] and will be released upon publication.

Experiment Setup. The experiments are conducted on 4,246 models in total, with 200 models for composite attack, 148 models for reflection attack, 34 models for hidden attack, 1920 models for filter attack, and 1944 models for patch attack. For composite attack, we generate 100 trojaned models on five datasets (i.e., MNIST [32], Fashion MNIST [71], SVHN [44], CIFAR10 [30], and the Youtube Face dataset [69]) using the official implementation [36]. We follow [39] to create 100 clean models (20 models for each dataset). Network in Network and VGG16 are used for these models. For reflection attack, there are three different reflection settings, i.e., same depth of field, out of focus, and ghost effect. For each setting, we generate 20 trojaned models on CIFAR10 and 17 trojaned models on ImageNet using the official repository [41]. We also obtain 20 clean models on CIFAR10 from [39] and 17 clean models on ImageNet from torchvision [2]. We employ a few model structures such as Network in Network, VGG, ResNet, SqueezeNet, and DenseNet in this experiment. For hidden attack, we leverage 34 models on ImageNet with half clean from [2] and half trojaned by [52]. The model structures used include VGG, ResNet, SqueezeNet, and DenseNet. For filter and patch attacks, we make use of the TrojAI datasets from rounds 2 to 4, consisting of 3,840 models with half clean and half trojaned. We also evaluate on 24 models on ImageNet. Details can be found in Appendix F. In addition, we study the hyper-parameters used in EX-RAY in Appendix G and Appendix I.

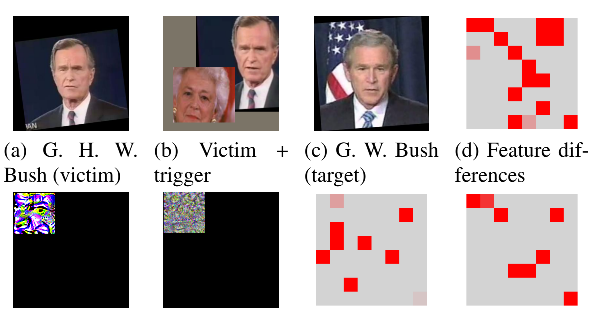
Table 1. EX-RAY on composite attack

	ABS					ABS+EX-RAY				
	TP	FP	FN	TN	Acc/ROC	TP	FP	FN	TN	Acc/ROC
MNIST	16	12	4	8	0.6	18	3	2	17	0.88
FMNIST	12	9	8	11	0.58	19	6	1	14	0.83
SVHN	15	7	5	13	0.7	19	4	1	16	0.88
CIFAR10	16	13	4	7	0.58	17	3	3	17	0.85
Youtube face	12	4	8	16	0.7	19	5	1	15	0.85

4.1. Detecting Complex Backdoor Attacks

In this section, we study the performance of EX-RAY in detecting three advanced backdoor attacks, namely, composite attack [36], reflection attack [41], and hidden attack [52], in comparison with a state-of-the-art technique ABS [39].

Detecting Composite Attack. Table 1 shows the detection results on composite attack. The first column denotes the datasets. Columns 2-6 show the detection results by ABS. Columns 7-11 present the detection results using ABS with EX-RAY. Columns TP, FP, FN, TN, and Acc/ROC denote the number of true positives, false positives, false negatives, true negatives, detection accuracy and ROC-AUC, respectively. The upstream methods output a binary result denoting whether a model is trojaned or not, and the clean and trojaned models are evenly distributed. Thus the ROC-AUC is the same as the accuracy. For ABS, we use the best possible bound for the inverted trigger size during detection. For ABS+EX-RAY, we set the bound for the trigger size to be half of the input. Observe that ABS+EX-RAY can achieve 83%-88% detection accuracy, substantially outperforming



(a) G. H. W. Bush (victim) (b) Victim + trigger (c) G. W. Bush (target) (d) Feature differences (e) Inverted trigger (trojaned model) (f) Inverted trigger (clean model) (g) Trigger features (trojaned model) (h) Trigger features (clean model)

Figure 4. Composite attack: George H. W. Bush + Barbara Bush → George W. Bush, and natural feature differences versus trigger differences. The maps in (d), (g), and (h) denote the neurons in a hidden layer with a red square denoting a distinctive neuron and its color the distinctive capability.

the original ABS, which has only 58%-70% accuracy. Note that EX-RAY reduces not only FPs, but also FNs. The reason for the latter is that ABS and other scanners like NC and K-arm are based on trigger size, while there is not a good separation by size. In contrast, EX-RAY is based on feature differencing.

Figure 4 (a-c) present a composite attack to a face recognition model, in which the presence of Barbara Bush in George H. W. Bush's images flips the classification results to George W. Bush. EX-RAY identifies the trigger features in Figure 4 (g) by comparing George H. W. Bush + trigger and George H. W. Bush. Observe that they are quite different from the natural feature differences between George H. W. Bush and George W. Bush in Figure 4 (d) that distinguish the clean examples from the classes. In contrast, when a clean model is scanned, a trigger is inverted (Figure 4 (f)) to flip George H. W. Bush + trigger to George W. Bush. Observe that it is equally uninterpretable as that in Figure 4 (e), the inverted trigger for the trojaned model. This is due to the inherent limitation that trigger inversion can hardly generate natural-looking input features but rather noise-like patterns. Therefore, it is difficult to perform feature differencing at the input level. EX-RAY, however, produces a set of trigger features (Figure 4 (h)) that have substantial commonality with the natural feature differences (Figure 4 (d)), that is, (h) is a subset of (d). In other words, the noise-looking trigger in Figure 4 (f) indeed denotes natural differences. This indicates the model is benign.

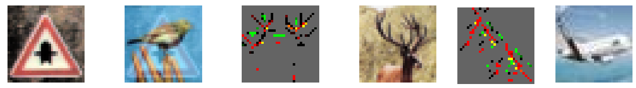
Detecting Reflection Attack. Table 2 presents the results for reflection attack. Column 1 denotes the datasets. Column 2 shows the three attack settings. Columns 3-7 present the results of ABS and the remaining columns ABS+EX-RAY. For ABS, we use the best possible bound for the inverted trigger size during detection. For ABS+EX-RAY, we set the bound for the trigger size to be 25% of the input. The stability of EX-RAY regarding trigger size bound can be found in Appendix I. Observe that our technique can achieve 80%-85% accuracy, whereas ABS only has 55%-68% accuracy.

Figure 5 (a) shows a triangle sign used as a trigger to flip images (Figure 5 (b)) to airplane (Figure 5 (f)) in a trojaned model. Figure 5 (e) shows a trigger generated by ABS for the airplane label in the trojaned model. Observe that the generated trigger has (triangle) features of the real trigger as in Figure 5 (a-b). EX-RAY determines the model is a true positive as the inverted trigger shares very few features with airplane. In contrast, Figure 5 (c) presents a trigger generated by ABS for a clean model with deer as the target label (Figure 5 (d)). Observe that the trigger resembles deer antlers. The triggers inverted for other labels also have a similar nature. EX-RAY hence recognizes the model as a true negative.

Detecting Hidden-trigger Attack. EX-RAY has 85% accuracy on hidden-trigger attack whereas ABS has 68%. Details and case studies can be found in Appendix B.

Table 2. EX-RAY on reflection attack

		ABS					ABS+EX-RAY				
		TP	FP	FN	TN	Acc/ROC	TP	FP	FN	TN	Acc/ROC
CIFAR10	Same DOF	13	7	7	13	0.65	18	4	2	16	0.85
	Out of focus	12	7	8	13	0.63	16	4	4	16	0.80
	Ghost effect	9	7	11	13	0.55	17	3	3	17	0.85
ImageNet	Same DOF	12	6	5	11	0.68	15	3	2	14	0.85
	Out of focus	9	6	8	11	0.59	13	3	4	14	0.80
	Ghost effect	10	6	7	11	0.62	15	3	2	14	0.85



(a) Injected trigger (b) Reflection (c) Inverted trigger (d) T: Deer (e) Inverted (f) T: Plane

Figure 5. A case for reflection attack

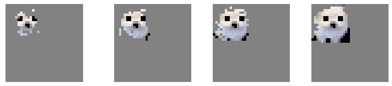
Table 3. TrojAI leaderboard results; CE L denotes cross entropy loss and R-A denotes ROC-AUC

	Round 2		Round 3		Round 4	
	CE L	R-A	CE L	R-A	CE L	R-A
ABS only	0.685	0.736	0.541	0.822	0.894	0.549
ABS+EX-RAY	0.324	0.892	0.323	0.900	0.322	0.902
Deficit from top	0	0	0.023	-0.012	0	0

4.2. Experiments on TrojAI and ImageNet Models

We evaluate EX-RAY on TrojAI rounds 2-4 training sets and ImageNet models. We use ABS as the upstream scanner and a relatively large trigger size bound to have a small number of false negatives. The experimental results show that the vanilla ABS encounters a large number of FPs, whereas EX-RAY substantially reduces the FPs by 78-100% with the cost of a slightly increased number of FNs by 0-30%. EX-RAY improves the overall detection accuracy by 17-41% across all the evaluated datasets. We also compare EX-RAY with eight baseline methods that make use of simple L2 distance, attribution/interpretation techniques, and one-sided (instead of symmetric) analysis. The results demonstrate that EX-RAY consistently outperforms baseline methods. In addition, we evaluate the runtime cost of EX-RAY, which takes 95 seconds to scan a model in TrojAI datasets on average, whereas the upstream scanner ABS takes 337 seconds. This delineates a reasonable overhead introduced by EX-RAY. We also study the effects of hyper-parameters of EX-RAY on a TrojAI dataset. The results show that EX-RAY is reasonably stable under various settings. Please see detailed results and discussions in Appendix H.

Results on TrojAI Leaderboard (Test Sets). Table 3



(a) 80 (b) 120 (c) 160 (d) 200

Figure 6. Adaptive attack triggers with different sizes

shows the results on TrojAI test sets. The column CE L shows the cross entropy loss of each method and the column R-A shows the ROC-AUC. In two of the three rounds, our solution achieves the top performance[4]. In round 3, it ranks number 2 and the results are comparable to the top performer. In addition, it beats the IARPA round goal (i.e., cross-entry loss lower than 0.3465) for all the three rounds. Our performance on the leaderboard, especially for round 2 that has a large number of natural backdoors and hence causes substantial difficulties for most performers[5], suggests the contributions of EX-RAY. As far as we know, many existing solutions such as [11, 16, 25, 29, 39, 57, 58, 60, 66] have been tested in the competition by different performers.

4.3. Adaptive Attacks

We conduct three adaptive attacks. In the first attack, we use features of the target class as the trigger. Since EX-RAY distinguishes trojaned and clean models by comparing the similarity between inverted triggers' features and distinctive features between the victim and target classes. Knowing our method, the attacker may choose to use the target class's features as the trigger to evade EX-RAY. We use parts of a target class image as the trigger. We use four triggers with different sizes. For each trigger, we trojan 10 Network in Network models on CIFAR10 with dog being the target class. Figure 6 (a-d) show the triggers with the size of 80, 120, 160 and 200, respectively. Observe that they are all part of some dog image. In addition, we also train 20 clean models on CIFAR10 to see if ABS+EX-RAY can distinguish the trojaned and clean models. The results are shown in Table 4. The first row shows the trigger size. The second row shows the average attack success rate when using the triggers on *clean models*. Note that since these triggers are composed of the target class's features, they might already be able to flip other images to the target even in clean models. The third row shows the FP rate. The fourth row shows the TP rate. Observe while ABS+EX-RAY consistently has a low FP rate, its TP rate decreases when the trigger become larger. When the trigger size is 200, the TP rate degrades to 0.5, meaning that it only recognizes half of the trojaned models. However, since the trigger (of size 200) is already quite large and contains strong target class features such that it can flip 70% of all the images to the target on all the clean models. This hence may not constitute a meaningful attack as it is almost equivalent to stamping a target class image.

Table 4. Adaptive Attack using target class features

Trigger Size	80	120	160	200
ASR on clean models	0.39	0.56	0.63	0.7
FP/ # of clean models	0.1	0.1	0.1	0.1
TP/ # of trojaned models	1	1	0.8	0.5

In the second attack, we force the activations of victim class samples with the trigger to resemble those of the target class inputs such that EX-RAY cannot distinguish the two. While the strongest attack can increase the FP rate of EX-RAY, it causes substantial model accuracy degradation so that the model becomes unusable. In the third adaptive attack, we generate a trigger similar to a third class while having similar feature-level representations to the target class. Experiments show that EX-RAY has 75% true positive rate and 10% false positive rate on this adaptive attack. Please see Appendix L.

4.4. Other Experiments

Appendix C shows that EX-RAY can detect another two state-of-the-art backdoor attacks. Appendix D shows that EX-RAY outperforms three other state-of-the-art backdoor defenses. Appendix E and J demonstrate that EX-RAY can boost the detection performance of other upstream scanners on composite attack, reflection attack, and TrojAI dataset. Appendix K shows that EX-RAY's masks indeed capture feature differences using a model interpretation technique. Appendix M shows how EX-RAY is used to help fixing injected backdoors.

5. Conclusion

We develop a method to detect complex backdoors that have large and dynamic triggers. It is built on a novel symmetric feature differencing technique that identifies a smallest set of features separating two sets of samples. Our results show that the technique is highly effective and outperforms the baselines. It also enables us to achieve top results on the rounds 2 and 4 leaderboard of the TrojAI competition, and rank the 2nd in round 3.

Acknowledgement

We thank the anonymous reviewers for their constructive comments. This research was supported, in part by IARPA TrojAI W911NF-19-S-0012, NSF 1901242 and 1910300, ONR N000141712045, N000141410468 and N000141712947. Any opinions, findings, and conclusions in this paper are those of the authors only and do not necessarily reflect the views of our sponsors.

[4]TrojAI ranks solutions based on the cross-entropy loss of scanning results. Intuitively, the loss increases when the model classification diverges from the ground truth. Smaller loss suggests better performance [3]. Past leaderboard results can be found at [4].

[5]Most performers had lower than 0.80 ROC-AUC in round 2.

References

[1] Sri-csl/trinity-trojai: This repository contains code developed by the sri team for the iarpa/trojai program. https://github.com/SRI-CSL/Trinity-TrojAI, 2021. 3, 13, 15

[2] torchvision.models — pytorch 1.7.0 documentation. https://pytorch.org/docs/stable/torchvision/models.html, 2021. 6, 13

[3] Trojai leaderboard. https://pages.nist.gov/trojai/, 2021. 1, 3, 8, 13

[4] Trojai past leaderboards. https://pages.nist.gov/trojai/docs/results.html#previous-leaderboards, 2021. 8

[5] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*, 2018. 3, 14

[6] Eugene Bagdasaryan et al. Blind backdoors in deep learning models. *arXiv preprint arXiv:2005.03823*, 2020. 1

[7] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020. 3

[8] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017. 3, 14, 17

[9] Yinzhi Cao, Alexander Fangxiao Yu, Andrew Aday, Eric Stahl, Jon Merwine, and Junfeng Yang. Efficient repair of polluted machine learning systems via causal unlearning. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018. 3

[10] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018. 3

[11] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In *IJCAI*, pages 4658–4664, 2019. 3, 8, 12, 13

[12] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017. 3

[13] Xiaoyi Chen, Ahmed Salem, Michael Backes, Shiqing Ma, and Yang Zhang. Badnl: Backdoor attacks against nlp models. *arXiv preprint arXiv:2006.01043*, 2020. 3

[14] Edward Chou et al. Sentinet: Detecting localized universal attack against deep learning systems. *IEEE SPW 2020*, 2020. 1, 3

[15] Min Du et al. Robust anomaly detection and backdoor attack detection via differential privacy. In *International Conference on Learning Representations*, 2019. 3

[16] N Benjamin Erichson, Dane Taylor, Qixuan Wu, and Michael W Mahoney. Noise-response analysis for rapid detection of backdoors in deep neural networks. *arXiv preprint arXiv:2008.00123*, 2020. 1, 3, 8, 13

[17] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to byzantine-robust federated learning. In *29th {USENIX} Security Symposium*, 2020. 3

[18] Hao Fu, Akshaj Kumar Veldanda, Prashanth Krishnamurthy, Siddharth Garg, and Farshad Khorrami. Detecting backdoors in neural networks using novel feature-based anomaly detection. *arXiv preprint arXiv:2011.02526*, 2020. 3

[19] 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 35th Annual Computer Security Applications Conference*, pages 113–125, 2019. 3

[20] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019. 1, 3

[21] 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. 3, 12

[22] Jonathan Hayase, Weihao Kong, Raghav Somani, and Sewoong Oh. Spectre: Defending against backdoor attacks using robust statistics. *arXiv preprint arXiv:2104.11315*, 2021. 3

[23] Shanjiaoyang Huang, Weiqi Peng, Zhiwei Jia, and Zhuowen Tu. One-pixel signature: Characterizing cnn models for backdoor detection. *arXiv preprint arXiv:2008.07711*, 2020. 3

[24] Xijie Huang et al. Neuroninspect: Detecting backdoors in neural networks via output explanations. *arXiv preprint arXiv:1911.07399*, 2019. 3, 12

[25] Susmit Jha, Sunny Raj, Steven Fernandes, Sumit K Jha, Somesh Jha, Brian Jalaian, Gunjan Verma, and Ananthram Swami. Attribution-based confidence metric for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 11826–11837, 2019. 1, 3, 8, 13

[26] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, 2018. 3

[27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[28] Panagiota Kiourti, Kacper Wardega, Susmit Jha, and Wenchao Li. Trojdrl: evaluation of backdoor attacks on deep reinforcement learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020. 3

[29] Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. Universal litmus patterns: Revealing backdoor attacks in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 301–310, 2020. 3, 8, 13

[30] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 6

[31] Keita Kurita et al. Weight poisoning attacks on pre-trained models. *arXiv preprint arXiv:2004.06660*, 2020. 3

[32] Yann LeCun et al. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2010. 6

[33] David Lewis. Counterfactuals and comparative possibility. In *Ifs*, pages 57–85. Springer, 1973. 2, 3

[34] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. *arXiv preprint arXiv:2101.05930*, 2021. 1

[35] Yiming Li, Tongqing Zhai, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shutao Xia. Rethinking the trigger of backdoor attack. *arXiv preprint arXiv:2004.04692*, 2020. 3

[36] Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. Composite backdoor attack for deep neural network by mixing existing benign features. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020. 1, 2, 6

[37] Kang Liu et al. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 273–294. Springer, 2018. 1

[38] Yuntao Liu et al. Neural trojans. In *2017 IEEE International Conference on Computer Design*, 2017. 3

[39] 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 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1265–1282, 2019. 1, 3, 6, 8, 13, 14

[40] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *25nd Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-221, 2018*. The Internet Society, 2018. 1

[41] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In *European Conference on Computer Vision*, 2020. 1, 2, 6

[42] Shiqing Ma, Yingqi Liu, Guanhong Tao, Wen-Chuan Lee, and Xiangyu Zhang. Nic: Detecting adversarial samples with neural network invariant checking. In *NDSS*, 2019. 3

[43] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 13

[44] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 6

[45] Anh Nguyen and Anh Tran. Wanet–imperceptible warping-based backdoor attack. *arXiv preprint arXiv:2102.10369*, 2021. 12

[46] Tuan Anh Nguyen et al. Input-aware dynamic backdoor attack. *Advances in Neural Information Processing Systems*, 33, 2020. 3, 12

[47] Ren Pang, Hua Shen, Xinyang Zhang, Shouling Ji, Yevgeniy Vorobeychik, Xiapu Luo, Alex Liu, and Ting Wang. A tale of evil twins: Adversarial inputs versus poisoned models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 85–99, 2020. 3

[48] Ren Pang, Zheng Zhang, Xiangshan Gao, Zhaohan Xi, Shouling Ji, Peng Cheng, and Ting Wang. Trojanzoo: Everything you ever wanted to know about neural backdoors (but were afraid to ask). *arXiv preprint arXiv:2012.09302*, 2020. 1

[49] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 6

[50] Andrea Paudice et al. Label sanitization against label flipping poisoning attacks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 5–15. Springer, 2018. 3

[51] Ximing Qiao et al. Defending neural backdoors via generative distribution modeling. In *Advances in Neural Information Processing Systems*, pages 14004–14013, 2019. 3

[52] Aniruddha Saha et al. Hidden trigger backdoor attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. 2, 3, 6

[53] Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang. Dynamic backdoor attacks against machine learning models. *arXiv preprint arXiv:2003.03675*, 2020. 3

[54] 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 *Advances in Neural Information Processing Systems*, 2018. 3

[55] Guangyu Shen, Yingqi Liu, Guanhong Tao, Shengwei An, Qiuling Xu, Siyuan Cheng, Shiqing Ma, and Xiangyu Zhang. Backdoor scanning for deep neural networks through k-arm optimization. 2021. 1, 3, 13

[56] Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*, 2016. 3, 14

[57] Karan Sikka, Indranil Sur, Susmit Jha, Anirban Roy, and Ajay Divakaran. Detecting trojaned dnns using counterfactual attributions. *arXiv preprint arXiv:2012.02275*, 2020. 3, 8, 13

[58] Octavian Suciu, Radu Marginean, Yigitcan Kaya, Hal Daume III, and Tudor Dumitras. When does machine learning {FAIL}? generalized transferability for evasion and poisoning attacks. In *27th {USENIX} Security Symposium*, 2018. 8, 13

[59] Mukund Sundararajan et al. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017. 3, 14

[60] Di Tang, XiaoFeng Wang, Haixu Tang, and Kehuan Zhang. Demon in the variant: Statistical analysis of dnns for robust backdoor contamination detection. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021. 3, 8, 13

[61] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems. In *European Symposium on Research in Computer Security*, pages 480–501. Springer, 2020. 3

[62] Brandon Tran et al. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems*, pages 8000–8010, 2018. 3

[63] Alexander Turner et al. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019. 3

[64] Akshaj Kumar Veldanda, Kang Liu, Benjamin Tan, Prashanth Krishnamurthy, Farshad Khorrami, Ramesh Karri, Brendan Dolan-Gavitt, and Siddharth Garg. Nnoculation: broad spectrum and targeted treatment of backdoored dnns. *arXiv preprint arXiv:2002.08313*, 2020. 1, 3

[65] Binghui Wang, Xiaoyu Cao, Neil Zhenqiang Gong, et al. On certifying robustness against backdoor attacks via randomized smoothing. *arXiv preprint arXiv:2002.11750*, 2020. 1

[66] 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 *2019 IEEE Symposium on Security and Privacy (SP)*, 2019. 1, 3, 8, 13, 15, 18

[67] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *Advances in Neural Information Processing Systems*, 33, 2020. 3

[68] Lun Wang, Zaynah Javed, Xian Wu, Wenbo Guo, Xinyu Xing, and Dawn Song. Backdoorl: Backdoor attack against competitive reinforcement learning. *arXiv preprint arXiv:2105.00579*, 2021. 3

[69] Lior Wolf et al. Face recognition in unconstrained videos with matched background similarity. In *CVPR 2011*, pages 529–534. IEEE, 2011. 6

[70] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020. 13

[71] Han Xiao et al. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 6

[72] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2019. 3

[73] Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A Gunter, and Bo Li. Detecting ai trojans using meta neural analysis. *arXiv preprint arXiv:1910.03137*, 2019. 3, 12, 13

[74] Xinyang Zhang et al. Trojaning language models for fun and profit. *arXiv preprint arXiv:2008.00312*, 2020. 3

[75] Shihao Zhao, Xingjun Ma, Xiang Zheng, James Bailey, Jingjing Chen, and Yu-Gang Jiang. Clean-label backdoor attacks on video recognition models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14443–14452, 2020. 3