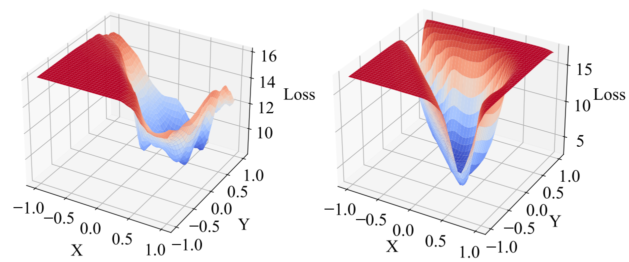
Better Trigger Inversion Optimization in Backdoor Scanning

Guanhong Tao, Guangyu Shen, Yingqi Liu, Shengwei An, Qiuling Xu
Shiqing Ma†, Pan Li, Xiangyu Zhang
Purdue University, †Rutgers University
{taog, shen447, liu1751, an93, xu1230, panli, xyzhang}@cs.purdue.edu
†sm2283@cs.rutgers.edu

Abstract

Backdoor attacks aim to cause misclassification of a subject model by stamping a trigger to inputs. Backdoors could be injected through malicious training and naturally exist. Deriving backdoor trigger for a subject model is critical to both attack and defense. A popular trigger inversion method is by optimization. Existing methods are based on finding a smallest trigger that can uniformly flip a set of input samples by minimizing a mask. The mask defines the set of pixels that ought to be perturbed. We develop a new optimization method that directly minimizes individual pixel changes, without using a mask. Our experiments show that compared to existing methods, the new one can generate triggers that require a smaller number of input pixels to be perturbed, have a higher attack success rate, and are more robust. They are hence more desirable when used in real-world attacks and more effective when used in defense. Our method is also more cost-effective.



(a) NC (b) Ours

Figure 1. Loss landscapes of NC and our method. The x-axis and y-axis show the coefficients on two random directions. The z-axis denotes the loss value.

1. Introduction

Backdoor attacks aim to induce model misclassification of arbitrary input samples to a target label by stamping a special input pattern called *trigger*. Backdoors could be *injected* by various methods, such as data poisoning [17, 35, 40] and neuron hijacking [39], and also naturally exist in normally trained models, called *natural backdoors* [41]. The latter is caused by distribution bias of low level features and can be exploited just like injected backdoors. For example, if a person always wears a unique pair of glasses in a clean face recognition dataset, the glasses may become a trigger to induce misclassification to the person.

Due to the prominent threat of backdoors, researchers have proposed a large body of defense solutions (see Section 2). Among them, backdoor scanning [21, 22, 68, 76] is an important type of defense. Many scanners [38, 41, 59, 65, 75] rely on *trigger inversion*, which leverages optimization to derive a small input pattern that can flip clean samples (of a *victim class*) to the target label. A model is considered having backdoor if an exceptionally small trigger can be found.

Most existing trigger inversion methods (e.g., ABS [38], K-arm [59], and Tabor [19]) are built on Neural Cleanse (NC) [65], which decouples a trigger into a perturbation vector and a *mask*. The perturbation vector denotes the perturbations applied to an input and the mask determines which part of the perturbation vector should be applied. NC minimizes the mask and the perturbation vector together to produce a small trigger (details in Section 3.1). Due to the multiplication correlation between the mask and the perturbation vector during optimization, NC can fall into local optima and fail to reach the optimal trigger, i.e., the smallest trigger with a high attack success rate. Figure 1a shows the loss landscape of NC using the contour plot with two random directions [16, 24, 31] with $(x = 0, y = 0)$ the optimum. Observe that there are multiple dips (local optima) on the loss surface, which prevent NC from reaching the optimum. In addition, triggers by NC are often not robust and may become ineffective when undertaking transformations (see Section 5.3).

Figure 2 shows the results of various techniques for generating a natural backdoor pattern for a normally trained

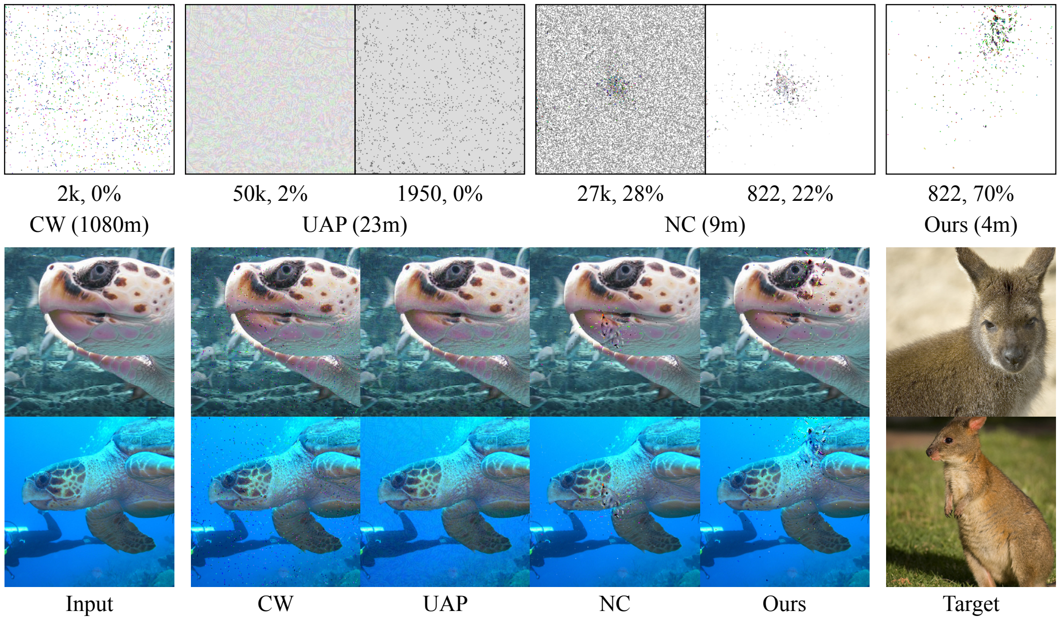


Figure 2. Comparison of generated backdoors. In the first row, the texts below backdoor images denote the number of perturbed pixels and the ASR on all the samples of loggerhead turtle from the validation set. The value shown together with the method name denotes the trigger generation time cost in minutes. The bottom two rows show example images stamped with backdoors by different methods, where the first column gives the victim class images and the last column the target class images.

model on ImageNet downloaded from [25]. Stamping each of these backdoors on sea turtle images can flip them to the kangaroo class. The first row shows the backdoor patterns by various inversion techniques. From left to right, the second and third rows show samples from the victim class (column 1), samples stamped with the backdoor patterns (columns 2-5), and the target class samples (last column). The fourth and fifth images in the first row denote the trigger generated by NC and its reduced version, respectively. Observe that the NC trigger requires perturbing 27k pixels and has only 28% ASR. When we reduce the NC trigger by removing the smallest perturbations to size 822 (which is the same as our trigger), the ASR degrades to 22%. This is because of the large number of local minima, as those on the loss surface of NC in Figure 1a. Our results in Section 5 show that on average, when NC triggers are reduced to the same size of ours, their ASRs on average degrade by 26%.

Problem Statement. In the context of backdoor attack and defense, a good optimization method (for trigger generation) is critical. In this paper, we say a method is good if it produces triggers that are (1) small (i.e., having a small number of perturbed pixels), (2) having a high *attack success rate* (ASR) (the percentage of unseen clean samples that can be flipped by the trigger), (3) robust (against input transformations), and (4) has low computation overhead. A good trigger generation method serves both attack and defense. If it is used in attack, e.g., generating natural triggers for normally trained models to induce intended misclassification, a small and robust trigger makes the attack easy to launch and effective in the physical world. If it is used in defense, smaller triggers can help scanners more effectively determine if a model is trojaned, as an exceptionally small trigger is a good indicator of injected backdoor [38, 59, 65], and have better effectiveness in model hardening. □

We propose a novel optimization method. Instead of optimizing the product of the perturbation vector and the mask, our method only optimizes a perturbation vector. Specifically, we leverage the long-tail effects of $\tanh$ function to represent the binary nature of perturbations, with one end modeling the maximum perturbation and the other end no perturbation. We introduce two $\tanh$ functions for each pixel, one denoting positive perturbation and the other negative. Our optimization method has a much smoother loss surface than NC as shown in Figure 1b. Observe that the loss values all descend along the valley towards the optimal point at the bottom. On the ImageNet dataset, our generated triggers are two orders of magnitude smaller than those of NC, 2.73 times more robust, and have 20% higher ASR on average. Our method is 2.15 times faster than NC. The last image in the first row of Figure 2 shows our trigger. It has the smallest number of perturbed pixels (822) with the highest ASR (70%) on the unseen validation set. We also compare with UAP [58] and CW [4] (another two trigger generation methods adapted from adversarial attacks). Ours is one or more orders of magnitude faster. The implementation of our method is publicly available [1].

2. Related Work

Backdoor Attack. Existing backdoor attacks poison the training set using intentionally crafted samples with injected backdoor patterns together with the target label such as patch attacks [7, 17]. To achieve stealthiness, a different type of backdoor attacks applies imperceptible perturbations on poisoned data with the original label like clean label attacks [55, 57, 78]. Another type of backdoor attacks crafts different backdoors for different inputs [35,49,56]. Other than poisoning the training set, backdoors also naturally exist in clean models [41]. Backdoor attacks can be launched on models with various applications, such as natural language processing [28, 77], transfer learning [53, 66, 73], and federated learning [3, 67, 71].

Backdoor Defense. To detect poisoned models [19, 22, 26, 52, 72], existing works reverse-engineer backdoors [38, 65], and leverage the difference between poisoned and clean models when reacting to input perturbations [21, 68, 76]. Existing techniques also detect and reject inputs stamped with backdoors [5, 6, 8, 10, 12, 13, 34, 42, 43, 60, 62, 63]. Verification methods aim to provide guarantees that models are not vulnerable to certain types of backdoors [23, 30, 64, 69]. There are also works focusing on eliminating backdoors [33] by pruning out compromised neurons [37] or retraining leveraging data augmentation technique [74].

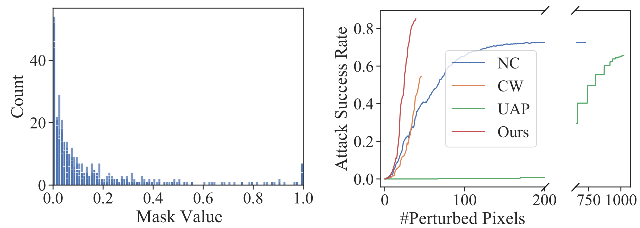
Optimization Methods for Trigger Generation. NC [65] is a state-of-the-art and we have detailed discussion and comparison throughout the paper. Existing adversarial attack methods were proposed to generate per-instance perturbations, such as fast gradient sign method (FGSM) [15], projected gradient descent (PGD) [44], JSMA [51], CW [4], and SLIDE [61], etc. Universal adversarial perturbation (UAP) [46] aims to generate a global perturbation that can cause a set of inputs to misclassify. We extend some of them to generate triggers (see the following section).

3. Existing Optimization Methods for Backdoor Trigger Generation and Their Limitations

In this section, we discuss in detail NC's optimization in trigger generation and two other optimizations that are popular in adversarial attack and hence adapted to trigger generation (i.e., CW and UAP). We focus on studying their limitations in trigger generation.

3.1. Optimization of Neural Cleanse (NC)

As mentioned earlier, the optimization method in NC is the most popular in trigger generation. Specifically, it solves the following optimization problem.



(a) The distribution of mask values for 10 runs of NC

(b) The relation between ASR and perturbed pixels

Figure 3. Characteristics of generated backdoors on all the test samples of a victim class from the CIFAR-10 dataset

$$\min_{\boldsymbol{m},\boldsymbol{p}} \mathcal{L}_{\text{NC}} = \mathcal{L}(\mathcal{M}(\boldsymbol{x}'), y_t) + \lambda \cdot \|\boldsymbol{m}\|_1, \forall \boldsymbol{x} \in \boldsymbol{X}, \quad (1)$$

$$\text{where } \boldsymbol{x}' = (1-\boldsymbol{m}) \circ \boldsymbol{x} + \boldsymbol{m} \circ \boldsymbol{p}. \quad (2)$$

Variables $\boldsymbol{m}$ and $\boldsymbol{p}$ denote the mask and the perturbation vector, respectively; $\mathcal{L}(\cdot,\cdot)$ denotes the cross entropy loss function of the subject model $\mathcal{M}$; y_t is the target label. Intuitively, the optimization aims to flip the classification result (the first term in the loss function) and reduce the trigger size (the second term). The introduction of the mask enables using optimization to reduce the trigger size. However, it also has some undesirable effects. NC has to optimize both $\boldsymbol{m}$ and $\boldsymbol{p}$ that are correlated by the $\circ$ operation in Equation 2, which is difficult and leads to low ASRs and large sizes. NC tends to produce many small values in the mask, indicating the corresponding input pixels need to be slightly perturbed. Although these values are small, many of them cannot be set to zero. Otherwise, the ASR degrades. These small and pervasive perturbations make attack in the physical world difficult and the backdoor not robust against input transformations (see results in Section 5.3).

Figure 3a shows the distribution of mask values for 10 random runs of NC for generating a natural trigger that flips plane to dog in a ResNet20 model on CIFAR-10. Observe that a large portion of mask values fall in the range from 0 to 0.1, which is equivalent to keeping 90% of the original pixel values. In Figure 3b, we start from the generated triggers, gradually set the smallest mask values to 0, which is equivalent to gradually reducing the number of perturbed pixels, and show the changes of ASR with the number of perturbed pixels. The number of perturbed pixels of the NC trigger is 725 with the ASR of 0.66. According to Figure 3a, most of them have small values. However, when the number of perturbed pixels is gradually reduced to 150, the ASR starts to degrade quickly, indicating the perturbations on these pixels need to be retained, even though they are still small. In contrast, our trigger has 0.83 ASR with only 39 perturbed pixels. Besides making physical attack difficult and not robust, the larger triggers by NC are less effective in exposing injected pervasive backdoors and model hardening (see Section 5.4 and Section 5.5).

3.2. Optimization of CW

There are existing optimization methods in adversarial attack that can be adapted for trigger generation, such as JSMA [51] and CW [4], with the later the state-of-the-art. The CW L^0 attack first searches for perturbations on all pixels that can cause misclassification using the L^2 norm. It then uses a processing step external to the optimization to remove the perturbations that are the least important after each optimization epoch. The algorithm can be easily adapted to generate backdoor: instead of optimizing one input, we optimize a set of inputs. Details of the optimization can be found in Appendix A.

The adapted CW optimization has a few limitations in trigger generation. First, it is very expensive. To determine the unimportant perturbations, it has to perform gradient back-propagation to each pixel of each input and sort the importance values for all pixels. As a result, it is often two to three orders of magnitude slower than our technique and NC (see Section 5). Second, its trigger size reduction is by an external step instead of optimization, and the reduction is monotonic. As such, if some step of reduction is not towards a global minimal, it cannot be reverted. As a result, CW's optimization yields 28.28% lower ASRs and 17.25% larger trigger sizes on average compared to ours on CIFAR-10 (see Section 5). The first image in the first row of Figure 2 shows a CW backdoor. Its number of perturbed pixels (2k) is smaller than NC (27k) but larger than ours (822). However, its ASR is close to 0. It also takes 1,080 minutes to generate, compared to 4 minutes in our method.

3.3. Optimization of Universal Adversarial Perturbation (UAP)

UAP [46, 58] generates a global perturbation that can cause a set of inputs to misclassify. It has a similar goal as ours and can be adapted for trigger generation. Details can be found in Appendix B.

4. Our Method

According to our problem statement in the introduction section, having a small number of perturbed pixels is critical for backdoor trigger generation. NC uses a mask vector to denote which parts of an input are subject to perturbation. However, it requires optimizing the product of the mask and the perturbation vector, which is difficult. We propose to directly optimize a perturbation vector, without using a mask like that in NC and CW. We use $\tanh$ functions to denote perturbations of individual pixels and use optimization to minimize the sum of all these functions. The long-tail effects of the $\tanh$ function allow us to nicely model the two ends for a pixel's value change, namely, a pixel is either not changed at all or has change of arbitrarily large magnitude (within bound). Figure 4a illustrates the concept. The y axis denotes pixel value and the x axis perturbation. The former is normalized to $[-1, 1]$ and the latter is in $(-\infty, +\infty)$. The red horizontal line denotes an original pixel value. The blue curve denotes how the pixel value changes with x. The pixel is changed by adding $\frac{1}{2}(\tanh(x) + 1)$. Note that although x is unbounded, the $\tanh$ function bounds the pixel value change in $(0, 1)$. Observe that the long left tail of the blue curve means that a large number of x values on the left correspond to close-to-0 changes to the pixel, whereas the long right tail means that those x values on the right correspond to the maximum change. The shape and the continuity of the curve on one hand encourage achieving tail values (in order to have a small loss value), and on the other hand, allow perturbations to recover from tail values if needed.

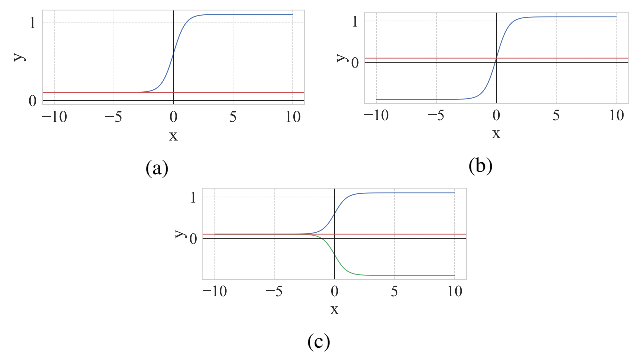


Figure 4. Illustration of using different $\tanh$ functions for the perturbation of a pixel. In (a), we denote positive change by adding $\frac{1}{2}(\tanh(x) + 1)$ to the original pixel value (red line). In (b), we denote both positive and negative changes by adding $\tanh(x)$ to the pixel value. In (c), we use two $\tanh$ functions to denote positive and negative changes, respectively.

However, using one $\tanh$ for each pixel only allows denoting changes along one direction, positive or negative. A naïve design is to use one tail to denote maximum positive change and the other tail to denote maximum negative change. That is, the pixel is changed by $\tanh(x)$. However, it loses the key benefit of encouraging as many pixels to have 0 value change as possible. Figure 4b illustrates the concept. Observe that the blue curve tends to go to either the maximum positive or the maximum negative. The part denotes 0 change (i.e., the interaction of the blue curve and the red line) has a steep slope such that it is unlikely for the optimization to stabilize at this point. Our solution is hence to use two $\tanh$ functions for a pixel, one denoting positive change and the other negative. Figure 4c illustrates the concept. In addition to the blue curve going upward, there is also the green curve that goes downward, denoting the negative changes. The key difference from the above naïve method is that both curves have a long tail on zero change, which enables the optimization to stabilize. If the optimization desires positive change, it just needs to go up along the blue curve and stay on the left tail along the green curve, and vice versa. Formally, we have the following optimization objectives.

$$\min_{\boldsymbol{b}_p,\boldsymbol{b}_n} \mathcal{L}_{ours} = \mathcal{L}\big(\mathcal{M}(\boldsymbol{x}'), y_t\big) + \alpha \cdot \mathcal{L}_{\text{pixel}}, \tag{3}$$

$$\text{where } \boldsymbol{x}' = \text{clip}\Big(\boldsymbol{x} + \frac{1}{2}\big(\tanh(\boldsymbol{b}_p)+1\big)\cdot maxp - \frac{1}{2}\big(\tanh(\boldsymbol{b}_n)+1\big)\cdot maxp\Big), \tag{4}$$

$$\text{and } \mathcal{L}_{\text{pixel}} = \sum_{h,w}\Bigg(\max{}_c\Big(\frac{1}{2}\big(\tanh(\frac{\boldsymbol{b}_p}{\gamma})+1\big)\Big)\Bigg) + \sum_{h,w}\Bigg(\max{}_c\Big(\frac{1}{2}\big(\tanh(\frac{\boldsymbol{b}_n}{\gamma})+1\big)\Big)\Bigg). \tag{5}$$

Variables $\boldsymbol{b}_p,\ \boldsymbol{b}_n \in (-\infty, +\infty)$ denote positive and negative perturbations, respectively; $\mathcal{L}(\cdot,\cdot)$ denotes the cross entropy loss function of the subject model $\mathcal{M}$; y_t is the target label; α controls the weight of the second objective. We dynamically adjust α according to the attack success rate during optimization to better balance the two objectives. Operation clip$(\cdot)$ constrains the values to the valid pixel value range. In Equation 4, $\frac{1}{2}(\tanh(\boldsymbol{b}_p)+1)\cdot maxp$ denotes the positive value change and $\frac{1}{2}(\tanh(\boldsymbol{b}_n)+1)\cdot maxp$ the negative change, with $maxp$ the upper bound of pixel values (i.e., 255). The function $\sum_{h,w}$ sums perturbations at all pixels with $\max_c$ the maximum among the three R, G, B channels. Parameter γ is used to alter the slope of $\tanh$ such that the optimization is smoother. We empirically set $\gamma = 10$.

A Simplified Version. Empirically we find that when using $\tanh$ in perturbing pixel values (in Equation 4), the optimizer continues to have gradient descents from the cross-entropy loss term in Equation 3, which is much more complex than the $\mathcal{L}_{\text{pixel}}$ term, to variables $\boldsymbol{b}_p$ and $\boldsymbol{b}_n$, even when the pixel value changes (e.g., $\frac{1}{2}(\tanh(\boldsymbol{b}_p)+1)\cdot maxp$) are already close to 0. This unnecessarily slows down the optimization. We hence replace Equation 4 with the following.

$$\boldsymbol{x}' = \text{clip}\big(\boldsymbol{x} + \text{clip}(\boldsymbol{b}_p \cdot maxp) - \text{clip}(\boldsymbol{b}_n \cdot maxp)\big), \tag{6}$$

Specifically, we remove the $\tanh$ functions on $\boldsymbol{b}_p$ and $\boldsymbol{b}_n$. Instead, we directly scale them with $maxp$ and then clip them to the valid range. This is equivalent to using a linear function in the cross-entropy loss term in Equation 3 instead of $\tanh$, while keeping the $\tanh$ functions in the $\mathcal{L}_{\text{pixel}}$ loss term. Intuitively, the shape of clip$(\boldsymbol{b}_p \cdot maxp)$ is similar to that of a $\tanh$ function. That is, the values on the two sides are zero and maximum, and there is a slope within a small range in the center. As such, Equation 6 approximates Equation 4. Empirically, we find that it makes our method faster and does not degrade the quality of generated triggers when it is used to generate natural triggers. It is faster because the clip operations prevent unnecessary gradient descents. However, we also find that Equation 4 is necessary in generating injected triggers for trojaned models during backdoor scanning (see Section 5.5). We speculate trojaned models have more non-linear behaviors than clean models due to data poisoning, which requires a smoother loss function. Specifically, trojaned models need to learn not only the relations between normal features and correct labels, but also the relations between poisoned data and the target label. This requires them to have more complex decision boundaries than benign models, and hence more non-linear behaviors. Smoother functions help escaping local optima with the increased non-linearity of trojaned models. Moreover, our ablation study in Appendix K shows that the $\tanh$ in Equation 5 is always beneficial.

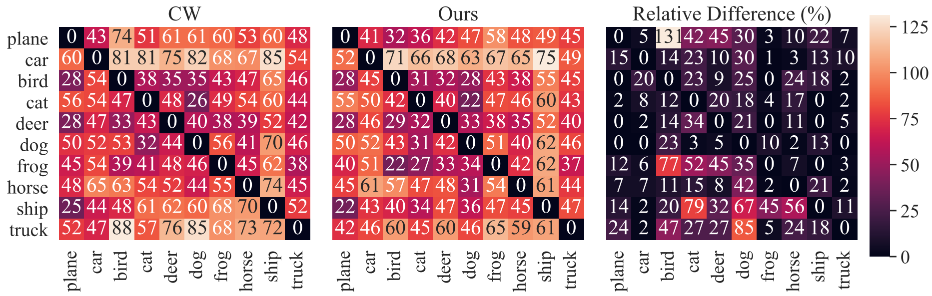
5. Evaluation

The evaluation is conducted on four datasets including ImageNet. For backdoor scanning, we leverage pre-trained models from the TrojAI competition [50] with a variety of classification tasks and model types. We also conduct an ablation study to understand the effects of different design choices (see Appendix K). Most experiments are conducted on a server equipped with two Intel Xeon Silver 4214 2.20GHz 12-core processors, 256 GB of RAM, and eight NVIDIA Quadro RTX 6000 GPUs.

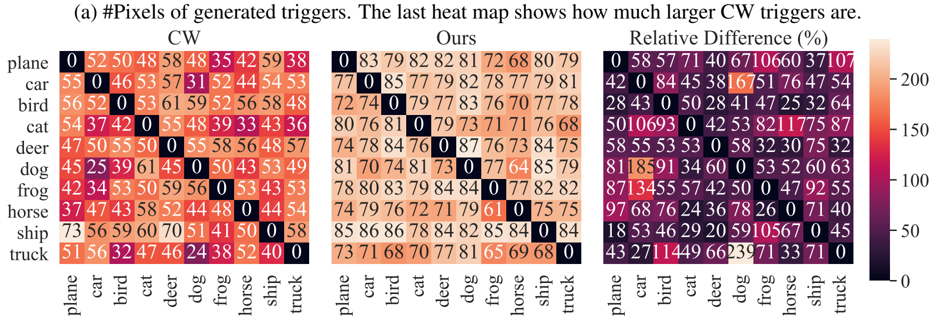
5.1. Experiment Setup

Datasets and Models. We use four datasets: CIFAR-10 [27], SVHN [47], LISA [45] and ImageNet [54]. We also conduct experiments on 300 pre-trained models (including clean and poisoned models) from rounds 2-4 of TrojAI competition [50]. Details are in Appendix C.

Baselines. Three existing optimization methods discussed in Section 3 are employed as the baselines: NC [65], CW [4], and UAP [58]. We randomly select 100 images from the validation set as the generation set for CIFAR-10 and SVHN, that is, the set of clean images used for trigger generation. For ImageNet, CW can only be performed on 50 images given the GPU memory limit. We hence randomly select 50 images from the training set as the generation set for all the methods. We use 90% ASR as the threshold on the generation set for CW, NC and ours. Since UAP may not produce any trigger with a high ASR, we do not use the threshold for UAP. As UAP is an L^∞ attack, we use an L^∞ bound of $8/255$ for CIFAR-10 and ImageNet, and 0.03 for SVHN. Due to the different natures of these methods, it is hard to define a uniform criterion (threshold) of convergence. For fair comparison, we use a conservative (i.e., fairly large) number of optimization epochs (1000



(a) #Pixels of generated triggers. The last heat map shows how much larger CW triggers are.



(b) ASRs of generated backdoors. The last map shows how much higher our ASRs are

Figure 5. Comparison of CW and ours for all class pairs on CIFAR-10

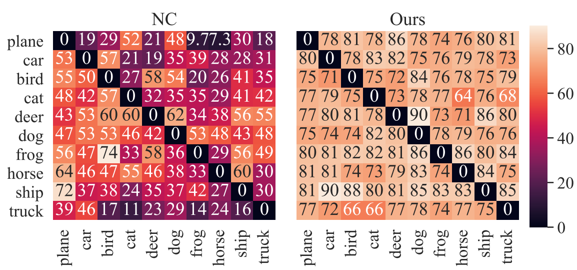


Figure 6. Comparison of NC and ours on the ASR for all class pairs on the CIFAR-10 dataset

epochs) for all the methods. Note that both CW and NC converge slower than ours. Please see the results on SVHN in Appendix F and the comparison with UAP in Appendix E due to the page limit.

Metrics. We consider the following criteria. The number of perturbed pixels (#pixels) measures the size of generated triggers. The attack success rate (ASR) gauges the percentage of unseen clean samples that can be flipped by a trigger. For evaluating ASR, we use the whole test set for CIFAR-10 and SVHN, and the whole validation set for ImageNet. We also measure the time cost.

5.2. Evaluation on CIFAR-10

Comparison with CW Optimization. In this experiment, we use CW and our method to generate natural triggers for all the class pairs for a clean ResNet20 model on CIFAR-10. Figure 5 shows the comparison. Each cell in heat map denotes the result for a natural backdoor flipping all the test samples from a victim class (row) to a target class (column). Figure 5a and Figure 5b show the number of perturbed pixels and the ASRs for CW (the left heat map) and ours (the middle heat map), respectively. The right heat map in Figure 5a shows how much larger the CW triggers are compared to ours. Observe that there are a few class pairs where CW and ours have the same trigger size, such as bird→plane and deer→plane. However, for other pairs, CW has a significantly larger trigger size than ours. For instance, for pair plane→bird, the trigger by CW is 131% larger than ours. Even with a much larger trigger, CW however still has lower ASR (50% vs 79% for plane→bird). This is because CW uses an external procedure to reduce the number of perturbed pixels (removing unimportant pixels based on $\boldsymbol{g}_i \cdot \boldsymbol{\delta}_i$ as discussed in Appendix A). Our method converges 10.88 times faster than CW on average (see Appendix D).

Comparison with NC. NC tends to generate triggers with a large number of small perturbations. The generated triggers hence cannot be easily applied in physical attacks. We conduct two experiments: (1) align the number of perturbed pixels of the NC triggers and our triggers and then compare the corresponding ASRs; (2) align the ASRs and compare the trigger sizes. For the first experiment, we use the sizes of our triggers as the reference, and align the NC triggers by gradually removing their smallest perturbations until they have the same sizes as ours. We then compare the ASRs of our triggers and the reduced NC triggers. Figure 6 presents the results. Observe that for most class pairs, the reduced NC triggers have less than 50% ASR. In the worst case, NC has only 7.3% ASR (plane→horse). On average, NC

Table 1. Comparison of different methods on a victim class loggerhead turtle (left table) and a victim class Persian cat (right table) from ImageNet. The first column shows the target classes. The second column shows the methods. The third/sixth column is the time cost in minutes and the fourth/seventh column the number of perturbed pixels (#Pixels). The fifth/eighth column shows ASR on the samples from validation set.

T	Method	Time	#Pixels	ASR	Time	#Pixels	ASR
Snowbird	CW	845.57	1849	0.00%	850.49	1097	4.00%
	UAP	21.19	50171	0.00%	22.44	50175	10.00%
	NC	9.19	26032	60.00%	9.35	25887	58.00%
	Ours	4.35	432	72.00%	4.43	519	66.00%
Robin	CW	1039.72	1674	0.00%	983.07	1063	2.00%
	UAP	21.77	50172	0.00%	22.85	50176	14.00%
	NC	9.19	26094	34.00%	9.44	26358	46.00%
	Ours	4.10	467	60.00%	4.52	433	54.00%
Grouse	CW	1035.85	2150	0.00%	882.65	1340	2.00%
	UAP	22.94	50174	0.00%	22.10	50174	12.00%
	NC	9.52	25977	14.00%	9.10	25688	44.00%
	Ours	4.02	675	60.00%	4.35	656	54.00%
Kangaroo	CW	1079.54	2165	0.00%	1028.50	1503	0.00%
	UAP	22.69	50173	2.00%	22.52	50176	8.00%
	NC	9.02	26583	28.00%	9.10	29165	54.00%
	Ours	4.27	822	70.00%	4.35	621	62.00%

has 39.83% ASR for all class pairs, degraded from 65.52% without reduction. This demonstrates that the large number of perturbations in NC triggers are important for a good ASR although they may have small values. In contrast, our triggers have higher ASRs than NC's for all class pairs. On average, ours have 78.22% ASR, even higher than the original NC triggers without size reduction. In the second experiment, we use NC's ASR as the reference and then gradually remove the smallest perturbations in our triggers until their ASRs drop to the same level as NC's and then compare the sizes. Figure 10a in Appendix presents the results. Observe that NC has one order of magnitude larger trigger sizes than ours for all the class pairs, indicating that our generated triggers indeed perturb much fewer pixels. We also study an NC variant, ABS [38], for trigger generation and have similar observations in Appendix G.

5.3. Evaluation on ImageNet

ImageNet has 1,000 classes. It is hence infeasible to test on all class pairs, especially for CW, which takes more than 14 hours to generate just one trigger. We hence randomly select 8 class pairs for experiments (see results on more class pairs in Table 9 in Appendix). Table 1 presents the quality of generated triggers. Observe that CW takes more than 800 minutes to generate a trigger for all the evaluated class pairs, and the highest ASR it can achieve is 4% for pair cat→snowbird. The size of generated triggers by CW is smaller than UAP and NC, but one order of magnitude larger than ours. UAP is much faster than CW, but is still one order of magnitude slower than ours. Its ASRs are also very low, with the highest 14%. Compared to the other two baselines, NC is faster and has a better ASR (42.25% on average). However, the triggers by NC have more than 25k perturbed pixels, which are almost half of the whole image ($224 \times 224 \approx 50k$). Our method has the lowest time cost, requiring less than 5 minutes to generate a valid trigger with a higher ASR (62.26% on average). Compared to NC, our triggers are two orders of magnitude smaller and have 20% higher ASR. We also conduct an experiment similar to the above on a desktop to demonstrate that our method can be easily deployed on machines with limited resources (see Appendix H). We further study the robustness of generated triggers under various image transformations. Results show that most of NC triggers become ineffective after 96% rescaling or 2° rotation (nearly 0% ASR). Our method has a consistently higher ASR than NC (see details in Appendix I). We also study the robustness of triggers by applying transformations during trigger generation. The observations are similar (see Appendix J).

Table 2. Comparison of different methods on model hardening. First two columns denote different training methods and model accuracy. The third and the fifth columns show the average trigger size measured by NC and ours, respectively. The fourth and the sixth columns denote the improvement.

Method	Accuracy	Adv$_{\text{NC}}$	Increase$_{\text{NC}}$	Adv$_{\text{Ours}}$	Increase$_{\text{Ours}}$
Natural	95.15%	55.11	-	32.83	-
UAP	93.16%	49.40	-8.86%	23.69	-27.08%
NC	93.45%	75.77	39.10%	45.57	39.69%
Ours	94.18%	122.79	121.07%	83.24	152.02%

5.4. Model Hardening

As natural backdoors widely exist in clean models. It is important to harden models against such attacks. We use the generated triggers by different methods to harden models and then apply NC and our method to generate triggers for all class pairs to measure improvement. Table 2 shows the results on a ResNet32 model for SVHN. Observe that the improvement on average trigger size by our method is 3x larger than those by existing methods (i.e., UAP and NC). We evaluate on two more datasets and five more models, and the observation is similar. Please see details in Appendix L.1.

5.5. Backdoor Scanning

We study the performance of existing backdoor scanners by replacing their trigger inversion method with ours on the polygon attack and three advanced backdoor attacks.

For polygon backdoors, we evaluate on 300 pre-trained models from the TrojAI competition. The results show our method can improve a state-of-the-art scanner K-arm [59]'s accuracy by 2% via replacing its optimization component

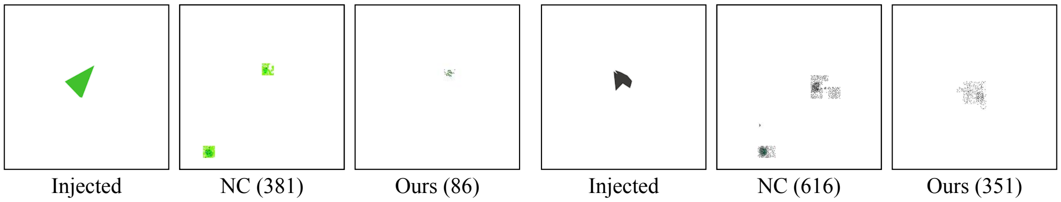


Figure 7. Comparison between injected backdoor and reverse engineered backdoor for poisoned models from the TrojAI dataset. Columns Injected show the original injected backdoors. Columns NC and Ours present backdoors generated by NC and ours, respectively. The numbers in the brackets denote the number of perturbed pixels of corresponding backdoors.

Table 3. Detecting a new pervasive backdoor attack [48]

Method	Dataset			
	MNIST	CIFAR-10	GTSRB	CelebA
NC	1.51	1.74	1.61	1.03
Ours	3.15	2.25	2.59	3.04

(based on NC) with ours. Note that the original scanner already had a state-of-the-art detection accuracy close to 90% such that 2% improvement is non-trivial. We also demonstrate example backdoors generated by NC and our method in Figure 7. The three images on the left show the injected trigger, the triggers inverted by NC and by ours, respectively. The three images on the right show another example. The number beside the method name denotes the trigger size. Observe that our generated backdoors are significantly smaller than NC's. Especially for the left case, ours is one order of magnitude smaller than that of NC. It is important to have small inverted triggers as scanners rely on the size of those triggers to distinguish poisoned models from benign ones. Note that in the TrojAI competition, the location of injected triggers is randomized to make the triggers more robust. The location shown in Figure 7 is only one of such cases. The generated triggers may be at any location.

We also evaluate our method on detecting three advanced backdoor attacks, namely, WaNet [48], invisible backdoor [32], and blind backdoor [2]. Compared to simple patch backdoors, WaNet and invisible backdoor have triggers that are not fixed. Their triggers are content based distortions. Blind backdoor uses inverted backdoors by existing scanners to adversarially train backdoored models, making the attack robust. We use the same anomaly index to detect backdoored models as that in the original NC paper, namely, a model with an anomaly index larger than 2 is considered backdoored. We download all the publicly available pre-trained models from WaNet [48]. Table 3 shows the anomaly indices for different models using NC and ours. We can see that NC cannot detect any of the evaluated models (consistent with the results reported in [48]), whereas our method can detect all the backdoored models (as we can generate a much smaller trigger for the target). The observation on the other two attacks are the same. Our inspection shows that although the injected triggers are pervasive, the models pick up low level features such as curly lines during poisoning. NC generates large triggers for the target class that are not distinguishable from those of benign classes, whereas our triggers are much smaller. Please see Appendix L.2 for more details. Our method is also consistently superior in detecting invisible backdoor and blind backdoor. Please see Appendix L.2.

6. Conclusion

We propose a new optimization method for backdoor trigger generation that minimizes the number of perturbed pixels. Compared to the state-of-the-art methods, our method is more cost-effective and can generate triggers with a smaller size, higher attack success rate, and better robustness. It also improves performance of model hardening and backdoor scanning.

Limitations of Our Method. Similar to NC and other existing scanners [19, 38, 59], our technique requires using a (small) set of clean samples in optimization, trying to flip their classification results. There are situations in which clean samples may not be available. It is unclear how our method can be extended to handle those cases. We will leave it to our future work.

Potential Negative Societal Impacts. The proposed method is general, aiming to generate better backdoor triggers. It could serve both attack and defense. Malicious users could use our method to generate triggers for pre-trained models and use them in attack. However, just like adversarial attack techniques are critical to improving model robustness, the triggers generated by our technique can be used to scan and mitigate backdoor vulnerabilities.

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.

References

[1] PixelBackdoor. https://github.com/Gwinhen/PixelBackdoor, 2022. 2

[2] Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. In *USENIX Security 21*, 2021. 8, 19

[3] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *AISTATS 2020*, pages 2938–2948. PMLR, 2020. 3

[4] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *SP 2017*, pages 39–57. IEEE, 2017. 2, 3, 4, 5

[5] Alvin Chan and Yew-Soon Ong. Poison as a cure: Detecting & neutralizing variable-sized backdoor attacks in deep neural networks. *arXiv preprint arXiv:1911.08040*, 2019. 3

[6] 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

[7] 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

[8] Edward Chou, Florian Tramer, and Giancarlo Pellegrino. Sentinet: Detecting localized universal attack against deep learning systems. *SPW 2020*, 2020. 3

[9] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR 20*, pages 3213–3223, 2016. 13

[10] Min Du, Ruoxi Jia, and Dawn Song. Robust anomaly detection and backdoor attack detection via differential privacy. In *ICLR 19*, 2019. 3

[11] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *CVPR 18*, pages 1625–1634, 2018. 12, 13

[12] 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

[13] 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 *ACSAC 19*, pages 113–125, 2019. 3

[14] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *IJRR*, 2013. 13

[15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572*, 2014. 3

[16] Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. In *ICLR 2015*, 2015. 1

[17] 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

[18] 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. 18

[19] Wenbo Guo, Lun Wang, Yan Xu, Xinyu Xing, Min Du, and Dawn Song. Towards inspecting and eliminating trojan backdoors in deep neural networks. In *ICDM*, 2020. 1, 3, 8, 18

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 12, 13

[21] Shanjiaoyang Huang, Weiqi Peng, Zhiwei Jia, and Zhuowen Tu. One-pixel signature: Characterizing cnn models for backdoor detection. In *ECCV*, 2020. 1, 3, 18

[22] Xijie Huang, Moustafa Alzantot, and Mani Srivastava. Neuroninspect: Detecting backdoors in neural networks via output explanations. *arXiv preprint arXiv:1911.07399*, 2019. 1, 3, 18

[23] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Intrinsic certified robustness of bagging against data poisoning attacks. *arXiv preprint arXiv:2008.04495*, 2020. 3

[24] Daniel Jiwoong Im, Michael Tao, and Kristin Branson. An empirical analysis of the optimization of deep network loss surfaces. *arXiv e-prints*, pages arXiv–1612, 2016. 1

[25] Keras. Applications. https://keras.io/api/applications/, 2021. 2, 13

[26] Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. Universal litmus patterns: Revealing backdoor attacks in cnns. In *CVPR*, pages 301–310, 2020. 3, 18

[27] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5, 12

[28] Keita Kurita, Paul Michel, and Graham Neubig. Weight poisoning attacks on pre-trained models. In *ACL 20*, 2020. 3

[29] Fredrik Larsson, Michael Felsberg, and P-E Forssen. Correlating fourier descriptors of local patches for road sign recognition. *IET Computer Vision*, 5(4):244–254, 2011. 13

[30] Alexander Levine and Soheil Feizi. Deep partition aggregation: Provable defense against general poisoning attacks. *arXiv preprint arXiv:2006.14768*, 2020. 3

[31] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 6391–6401, 2018. 1

[32] Shaofeng Li, Minhui Xue, Benjamin Zhao, Haojin Zhu, and Xinpeng Zhang. Invisible backdoor attacks on deep neural networks via steganography and regularization. *TDSC 20*, 2020. 8, 19

[33] Yige Li, Nodens Koren, Lingjuan Lyu, Xixiang Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. In *ICLR 21*, 2021. 3

[34] 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

[35] Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. Composite backdoor attack for deep neural network by mixing existing benign features. In *CCS 20*, pages 113–131, 2020. 1, 3

[36] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013. 12

[37] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *RAID 18*, pages 273–294. Springer, 2018. 3

[38] 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 *CCS 19*, pages 1265–1282, 2019. 1, 2, 3, 7, 8, 15, 18

[39] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *NDSS 18*, 2018. 1

[40] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In *ECCV 20*, pages 182–199. Springer, Cham, 2020. 1

[41] Yingqi Liu, Guangyu Shen, Guanhong Tao, Zhenting Wang, Shiqing Ma, and Xiangyu Zhang. Ex-ray: Distinguishing injected backdoor from natural features in neural networks by examining differential feature symmetry. *arXiv preprint arXiv:2103.08820*, 2021. 1, 3

[42] Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural trojans. In *ICCD 17*, pages 45–48. IEEE, 2017. 3

[43] Shiqing Ma, Yingqi Liu, Guanhong Tao, Wen-Chuan Lee, and Xiangyu Zhang. Nic: Detecting adversarial samples with neural network invariant checking. In *NDSS 19*, 2019. 3

[44] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR 18*, 2018. 3, 13

[45] Andreas Mogelmose, Mohan Manubhai Trivedi, and Thomas B Moeslund. Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. *T-ITS*, 13(4):1484–1497, 2012. 5, 12

[46] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *CVPR 17*, pages 1765–1773, 2017. 3, 4

[47] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 5, 12

[48] Anh Nguyen and Anh Tran. Wanet–imperceptible warping-based backdoor attack. In *ICLR 2021*, 2021. 8, 19

[49] Tuan Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. *NeurIPS 20*, 2020. 3

[50] NIST. TrojAI. https://pages.nist.gov/trojai/, 2020. 5, 13, 18

[51] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *EuroS&P 16*, pages 372–387. IEEE, 2016. 3, 4

[52] Ximing Qiao, Yukun Yang, and Hai Li. Defending neural backdoors via generative distribution modeling. In *NeurIPS 19*, pages 14004–14013, 2019. 3, 18

[53] Shahbaz Rezaei and Xin Liu. A target-agnostic attack on deep models: Exploiting security vulnerabilities of transfer learning. In *ICLR*, 2020. 3

[54] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 5, 13

[55] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden trigger backdoor attacks. In *AAAI 20*, number 07, pages 11957–11965, 2020. 3

[56] 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

[57] 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 *NeurIPS 18*, pages 6103–6113, 2018. 3

[58] Ali Shafahi, Mahyar Najibi, Zheng Xu, John Dickerson, Larry S Davis, and Tom Goldstein. Universal adversarial training. In *AAAI 20*, volume 34, pages 5636–5643, 2020. 2, 4, 5, 12

[59] 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. In *ICML*, 2021. 1, 2, 7, 8, 18

[60] Di Tang, XiaoFeng Wang, Haixu Tang, and Kehuan Zhang. Demon in the variant: Statistical analysis of dnns for robust backdoor contamination detection. In *USENIX Security*, 2021. 3

[61] Florian Tramèr and Dan Boneh. Adversarial training and robustness for multiple perturbations. In *NeurIPS 19*, 2019. 3

[62] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In *NeurIPS*, pages 8000–8010, 2018. 3

[63] 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. 3, 18

[64] Binghui Wang, Xiaoyu Cao, Neil Zhenqiang Gong, et al. On certifying robustness against backdoor attacks via randomized smoothing. *arXiv preprint arXiv:2002.11750*, 2020. 3

[65] 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 *S&P 19*, pages 707–723. IEEE, 2019. 1, 2, 3, 5, 17, 18, 19

[66] Bolun Wang, Yuanshun Yao, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. With great training comes great

vulnerability: Practical attacks against transfer learning. In *USENIX Security 18*, pages 1281–1297, 2018. 3

[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. *NeurIPS*, 33, 2020. 3

[68] Ren Wang, Gaoyuan Zhang, Sijia Liu, Pin-Yu Chen, Jinjun Xiong, and Meng Wang. Practical detection of trojan neural networks: Data-limited and data-free cases. In *ECCV*, 2020. 1, 3, 18

[69] Maurice Weber, Xiaojun Xu, Bojan Karlas, Ce Zhang, and Bo Li. Rab: Provable robustness against backdoor attacks. *arXiv preprint arXiv:2003.08904*, 2020. 3

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

[71] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *ICLR*, 2019. 3

[72] 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, 18

[73] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. Latent backdoor attacks on deep neural networks. In *CCS*, pages 2041–2055, 2019. 3

[74] Yi Zeng, Han Qiu, Shangwei Guo, Tianwei Zhang, Meikang Qiu, and Bhavani Thuraisingham. Deepsweep: An evaluation framework for mitigating dnn backdoor attacks using data augmentation. *arXiv preprint arXiv:2012.07006*, 2020. 3

[75] Xinqiao Zhang, Huili Chen, and Farinaz Koushanfar. Tad: Trigger approximation based black-box trojan detection for ai. *arXiv preprint arXiv:2102.01815*, 2021. 1

[76] Xiaoyu Zhang, Ajmal Mian, Rohit Gupta, Nazanin Rahnavard, and Mubarak Shah. Cassandra: Detecting trojaned networks from adversarial perturbations. *arXiv preprint arXiv:2007.14433*, 2020. 1, 3, 18

[77] Xinyang Zhang, Zheng Zhang, and Ting Wang. Trojaning language models for fun and profit. In *EuroS&P*, 2021. 3

[78] Shihao Zhao, Xingjun Ma, Xiang Zheng, James Bailey, Jingjing Chen, and Yu-Gang Jiang. Clean-label backdoor attacks on video recognition models. In *CVPR 20*, pages 14443–14452, 2020. 3

[79] Liuwan Zhu, Rui Ning, Cong Wang, Chunsheng Xin, and Hongyi Wu. Gangsweep: Sweep out neural backdoors by gan. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 3173–3181, 2020. 18