Few-shot Backdoor Defense Using Shapley Estimation

Jiyang Guan
Institute of Automation, CAS
Beijing, China

Zhuozhuo Tu
The University of Sydney
Darlington, Australia

Ran He
Institute of Automation, CAS
Beijing, China

Dacheng Tao
JD Explore Academy
Beijing, China

Abstract

Deep neural networks have achieved impressive performance in a variety of tasks over the last decade, such as autonomous driving, face recognition, and medical diagnosis. However, prior works show that deep neural networks are easily manipulated into specific, attacker-decided behaviors in the inference stage by backdoor attacks which inject malicious small hidden triggers into model training, raising serious security threats. To determine the triggered neurons and protect against backdoor attacks, we exploit Shapley value and develop a new approach called Shapley Pruning (ShapPruning) that successfully mitigates backdoor attacks from models in a data-insufficient situation (1 image per class or even free of data). Considering the interaction between neurons, ShapPruning identifies the few infected neurons (under 1% of all neurons) and manages to protect the model's structure and accuracy after pruning as many infected neurons as possible. To accelerate ShapPruning, we further propose discarding threshold and ϵ-greedy strategy to accelerate Shapley estimation, making it possible to repair poisoned models with only several minutes. Experiments demonstrate the effectiveness and robustness of our method against various attacks and tasks compared to existing methods.

1. Introduction

Over the past years, Deep Neural Networks (DNNs) play a great role in machine learning and are applied in many critical domains such as face recognition [35], autonomous driving [6], and medical diagnosis [17]. However, because of a lack of transparency and interpretability [24], DNNs are easy to be manipulated by an adversary into attacker-decided behaviors and make serious mistakes in security-related areas, which causes serious threats and concerns. For example, it has been observed that adding deliberate and small distortion to the images in the inference stage(*i.e.*, adversarial examples) can cause misclassification in neural network classifiers [11].

Backdoor attacks, on the other hand, are a different type of attack, making use of opacity and overfitting of DNNs to create a maliciously trained network which achieves state-of-the-art performance on normal samples but behaves badly on specific attacker-chosen inputs. Gu *et al*. [12] demonstrates that, compared with adversarial examples, backdoor attacks can cause wrong predictions in models with much smaller distortion and even mislead facial recognition systems to identify any face as celebrities with almost invisible disturbance. Meanwhile, for black-box models like DNNs, it is difficult to identify the backdoor, and we can only use the test dataset to judge whether they are poisoned. Thus, the backdoor attack is more imperceptible and dangerous. Furthermore, as training on cloud or directly using the third-party trained models becomes more common today, backdoor attacks have more access to the models' training procedure. Thus, it is much easier for them to inject triggers into models in recent years.

The vulnerabilities to backdoor attacks raise concerns about the security of DNNs [20], and many defense methods have been proposed, trying to mitigate backdoor from the models *e.g*. Fine Pruning [21], Neural Cleanse [37], GangSweep [42] *etc*. However, these methods need a relatively large amount of clean data (*e.g*. 10% of training data required in Neural Cleanse), and can't locate poisoned neurons accurately (*e.g*. pruning 70% of all neurons in Fine Pruning). To determine the poisoned neurons and mitigate backdoor, we introduce Shapley value and propose a ShapPruning framework to guide detecting the attacked neurons, which successfully mitigates backdoor in the given mod-

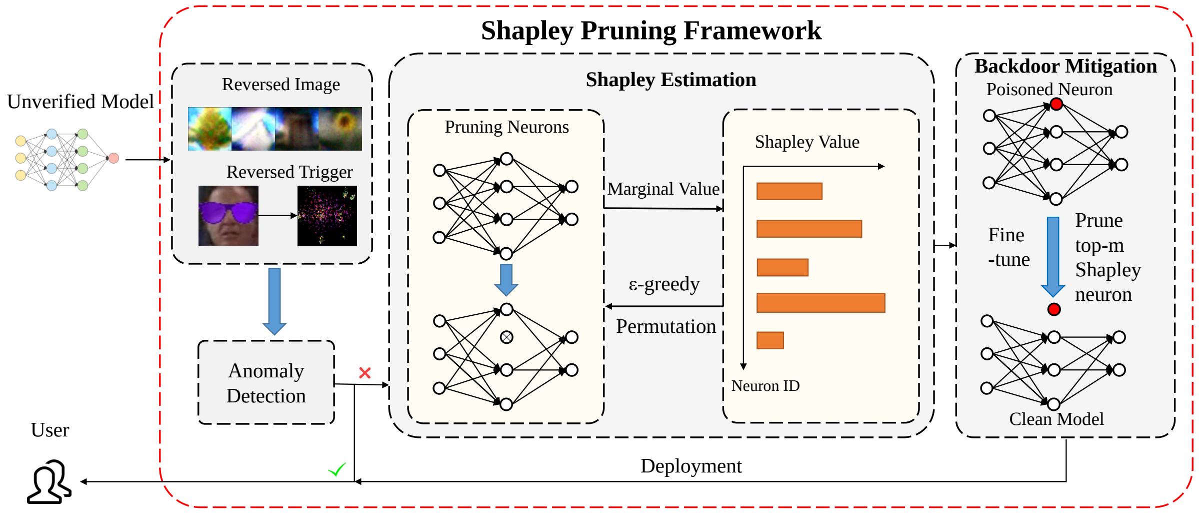


Figure 1. Shapley Pruning Framework. Our framework consists of four components, trigger and data reverse, detection, Shapley estimation, backdoor mitigation, and can effectively remove backdoor in models.

els. Shapley value is a concept from game theory and is used to allocate worth to cooperative players [1, 9, 32]. We use Shapley value to attribute the overall backdoor behavior to each neuron and find neurons with the largest Shapley value which are most responsible for backdoor behavior in models. Compared to prior work, our ShapPruning method can handle the data-insufficient situation and needs only a tiny amount of data (*e.g*. one image per class or even free of clean data with recovered data) and prunes a very small number of neurons (about 1% of all the neurons), to maintain a good classification accuracy (under 1% accuracy decline in most cases) and clean backdoor clearly.

Our contributions are summarized as follows:

- We introduce Shapley value into the backdoor area and propose a backdoor mitigation method called Shapley Pruning which can locate and prune poisoned neurons accurately with the reversed trigger.
- We also propose discarding threshold and ϵ-greedy based method to accelerate Shapley value's estimation, which yields a more accurate estimation with much less time.
- Our method considers the relationship between neurons and locates the attacked neurons accurately with few images. As a result, it can prune only 1% of all neurons to recover the model with a small accuracy decrease (accuracy declines 0.1% in the GTSRB dataset and the attack success rate drops to 0.4%). Moreover, our method is robust in different situations.
- We utilize information in model's batch normalization layer and propose a data-free backdoor cleanse method with mixture-mode ShapPruning.

2. Related Work

Many defense methods have been proposed to deal with the security threats of backdoor attacks. From the perspective of the defender, there are two main settings to mitigate backdoor, *i.e*., model available defense and data available defense. Data available defenses usually use anomaly detection to detect and eliminate abnormal images in the poisoned training dataset [5, 36], or weaken the influence of backdoor dataset during model training [19, 29, 34]. However, in many cases, datasets are unavailable due to privacy concerns and what we can have access to is only trained models which might be injected malicious backdoor attacks. Thus, model available defense attract more attention. Our work considers this setting and focuses on clean data insufficient situations to recover poisoned models.

There is a broad body of literature trying to solve this problem. Fine Pruning [21] uses the activation of each neuron on clean data to determine which neurons to prune. But, because deep neural networks are complicated, using activation to guide neuron pruning ignores the correlation between neurons and can't locate the poisoned neurons accurately. Neural Cleanse [37] tries to reverse triggers and uses an unlearning way to patch the model. To improve Neural Cleanse, GangSweep [42], Tabor [13], and DeepInspect [2] were proposed to use GANs [10] and interpretable AI to generate better-reversed triggers. However, these methods can't accurately locate the neurons under attack, and their performance, to some extent, depends on fine-tuning. As a result, they usually need a relatively large amount of clean data and prune a large number of neurons. When clean data is insufficient, the performance of these methods declines. Besides, DeepInspect [2] is fragile, limited, and the data reverse used by this method is based on a single-layer network and a small face dataset situation [8]. Unlike the previous method, our method can mitigate backdoor from the poisoned models with only few images (even without clean data) and prune only a few neurons.

3. Method

We present our Shapley Pruning framework in this section. Firstly, we introduce Shapley value to DNNs and give its definition. Then, we give an algorithm for estimating Shapley value where we propose ϵ-greedy and discarding threshold to accelerate its estimation. Since Shapley value is evaluated on backdoor dataset, we involve trigger reverse synthesis to generate that dataset. Finally, we involve image recovery and propose a data-free backdoor mitigation method. An overview of our framework is given in Fig. 1.

3.1. Shapley Value

In DNNs, since there are a large number of neurons and complicated interactions between them, it is difficult to quantify each neurons' contribution to the overall output. To tackle this problem, we introduce Shapley value which, as one of the most important concepts in cooperative game theory, can allocate values to each player using the average of marginal values [1], and be used to determine the contribution of each neuron to the overall output.

We can treat a network as an n-player game with each neuron as a player. Let N be the set of all n neurons in the neural networks, denoted by $N = \{1, \dots, n\}$ and m be a metric function evaluating performance of players. In neural networks, m can be a score function such as accuracy or loss. The marginal contribution of neuron i can be defined as:

$$margin(i) = m(C \cup \{i\}) - m(C) \tag{1}$$

where C is a subset of players not containing i, i.e., expressed as $C \subset N \backslash i$. With the marginal contributions, Shapley value ϕ for neuron i can be defined using the average of them as follows [32]:

$$\phi_i(m) = \frac{1}{n} \sum_{C \subset N \backslash i} P_C \cdot (m(C \cup i) - m(C)) \tag{2}$$

where $P_C = \frac{(n-c-1)!c!}{(n-1)!}$ represents the relative importance of subset C , and c is the cardinality of C. In the next subsection, we will provide an algorithm for computing Shapley value for each neuron.

3.2. Estimation for Shapley Value

From Eq. (2), Shapley value can be expressed as the average of marginal contributions of the neuron in all possible orders. We define O as a permutation of neurons and $Af^i(O)$ means a subset of neurons after neuron i in the order O. $\pi(N)$ is all possible orders of neurons. Then, Shapley value of neuron i can be rewritten as follows [1]:

$$\phi_i(m) = \sum_{O \in \pi(N)} \frac{1}{n!} (m(Af^i(O) \cup i) - m(Af^i(O))) \tag{3}$$
$$i = 1, \dots, n$$

Eq. (3) shows that computing ϕ_i is equivalent to calculating the expectation of a random variable. Despite that estimating Shapley value exactly is time-consuming as it involves $n!$ permutations of all neurons in deep neural networks, we can approximate it by applying the Monte-Carlo estimation [7] which first samples permutations of neurons and then calculates the average of marginal contributions with those sampled permutations. Further, we propose discarding threshold and ϵ-greedy acceleration to estimate Shapley value more fast and precisely.

Discarding threshold. The main computational cost in estimating Shapley value is computing the marginal contribution of each neuron. For a small subset of neurons $Af^i(O)$, our experiments find that after removing a small portion of neurons, ASR (attack success rate) of the networks will reduce to a low rate sharply. Thus, the marginal contribution of neurons after that can be negligible, and we can avoid calculating it, which saves substantial computational cost. Moreover, we mainly focus on top-k neurons which are the most important in ASR when the network structure is complete and the performance is normal. Thus, we propose to discard neurons' marginal value after ASR is below a threshold, *e.g.* 0.2. Note that we do not set the marginal value of the neurons to be zero after the model's performance reduces to a low rate. This is because if the neurons with larger Shapley value are in the latter part of the permutation, the marginal value of those neurons will be set to zero, making their Shapley value underestimated, especially when the number of average iterations is small. Our experiments also demonstrate that setting to zero can cause fluctuation and randomness in Shapley estimation, and we need a large number of average iteration to offset that negative effect.

ϵ-greedy acceleration. Since we focus on neurons with top-k largest Shapley values, neurons with larger Shapley value should be calculated more times to be estimated more precisely and get a more accurate sorting of them. However, because of discarding threshold, the neurons after ASR is under a threshold will be discarded and lose the opportunity to be computed. To improve their calculation times, we should assign neurons with the top Shapley values a higher probability to be at the front of the permutations. To this end, we propose an ϵ-greedy based acceleration.

ϵ-greedy algorithm [38], as an optimization method, selects the best choice with probability $1-\epsilon$ and chooses from all the choices randomly with probability ϵ. ϵ-greedy algorithm is usually used in reinforcement algorithm and helps find the best choice in the action space [15]. Thus, to improve estimation efficiency, we follow this idea and propose an ϵ-greedy based algorithm, balancing exploration and utilization in finding the top-k Shapley value neurons. We divide neurons into two groups based on the current Shapley value, estimated by the current average of each neurons' marginal values, top-m and the other ($m \geq k$). We randomly permute neurons before the average iteration is smaller than l. Then after l, we utilize ϵ-greedy, choosing a random neuron from top-m with probability $1 - \epsilon$ and choosing from the others with probability ϵ iteratively, and get a permutation to prune the neurons.

3.3. Trigger Reverse Synthesis

We choose ASR as the metric function to estimate each neurons' Shapley value, and therefore, we need the backdoor dataset to calculate ASR. Furthermore, backdoor attacks manipulate DNNs to specific behaviors only on triggered images, indicating that the poisoned neurons are only activated on the backdoor images rather than normal images. Thus, reversing triggers from the backdoor networks, as an important part, can help the removal of backdoor neurons in poisoned models. Intuitively, backdoor attacks make use of DNNs' overfitting feature to create a shortcut for the trigger to cause DNNs' misclassification. We can use the trigger reverse synthesis to reverse backdoor trigger in the models and generate the reversed backdoor dataset.

We first inject class-specific reversed trigger T_c into clean images and get the triggered image a_c as follows:

$$a_c = (1 - M_c) \odot a + M_c \odot T_c \tag{4}$$

where M_c represents the mask for class c, deciding the location and intensity of trigger being injected into original images, a represents the original image, T_c represents the trigger patter for class c and $\odot$ means Hadamard product. Similar to adversarial example generation, we optimize on networks' misclassification and trigger size to reverse backdoor. We use Cross-Entropy loss to optimize misclassification of triggered images to class c and $L1$ norm of the mask to optimize the trigger size. We sum the above objectives and get the equation as follows:

$$\min_{M_c, T_c} \quad CE(y_c, f(a_c)) + \lambda \cdot |M_c|_1 \quad for \quad a \in A \tag{5}$$

where y_c represents label for class c, A represents clean images available, $CE(\cdot)$ represents Cross-Entropy loss, $|M_c|_1$ represents $L1$ norm of the mask, and λ represents the trade-off parameter.

From the above method, we can get the reversed trigger for each target class. However, judging whether the network is poisoned and which is the target label is still a problem. Intuitively, since backdoor training produces a shortcut for the backdoor trigger in the poisoned models, the reversed trigger for the target label is the smallest among all the classes. Thus, we can get the reversed trigger and target label by finding the smallest trigger in trigger reverse synthesis.

Backdoor model detection. First, our experiment (given in Appendix) shows that the L1 norm of the reversed trigger for the target label is much smaller than the others. Thus, L1 norm for the target label can be seen as an outlier from the other triggers, and we can use anonamly detection method to find the target label. We employ MAD (Median Absolute Deviation) to judge whether the models are poisoned. By MAD, supposing that mask norms obey normal distribution [30], any anomaly index $I = d_i/MAD$ larger than a specific value will be treated as an outlier, where d_i is the absolute deviation between triggers' L1 norm and their median.

However, in experiments, we find that the reversed triggers for some classes can't converge to a small L1 norm, and their norms are abnormally larger than expected, causing a false positive in backdoor detection. Different from [37], since we only focus on whether the smallest reversed trigger is an outlier, we can just apply MAD to the set of the triggers whose norms are smaller than the median to avoid anomaly large norms. We define their deviations' set as $D_{small} = \{d_1, \ldots, d_l\}$. Furthermore, because normal distribution is symmetrical, the median of D_{small} can be used to replace the median of all labels' deviation as MAD. Then, we can use MAD to estimate the standard deviation σ of the distribution of norms and use it to detect backdoor model with confidence probability p, expressed as follows:

$$\sigma = \frac{MAD}{\Phi^{-1}(\frac{3}{4})} \approx 1.4826 \cdot MAD \tag{6}$$

$$\frac{d_i}{\sigma} \leq d = \Phi^{-1}(\frac{p+1}{2}), \quad d_i = d_1, d_2, \cdots, d_l \tag{7}$$

where $\Phi(\cdot)$ represents cumulative probability distribution of standard normal distribution and d represents max bound for d_i/σ with probability p. The norm with the deviation larger than $\sigma \cdot \Phi^{-1}(\frac{p+1}{2})$ is an outlier and the model has been poisoned. Proof and experiments on model detection are provided in Appendix.

3.4. Data-free Backdoor Mitigation

As we mentioned above, we can mitigate backdoor from models with few images using ShapPruning. Then we further research on a no clean data situation and propose a data-free ShapPruning method. To help data-free backdoor mitigation with Shapley estimation, we need to reverse training images from the poisoned models first. Recent works in transfer learning show that the batch normalization layer (BN layer) can be used to recover better images from the trained models and improve transfer efficiency [4, 14, 39]. Furthermore, because backdoor attacks only poison a very small portion of training datasets, they won't affect the information in BN layers, and thus, we can use BN layers to better reverse images. Then we can express the difference of the mean and variance between the recovered data and original training data in the model's each BN layer as follows:

$$L_{bn}(x) = \sum_i div(N(\mu_i(x), \sigma_i(x))), N(\mu_i, \sigma_i)) \qquad (8)$$

where $div(\cdot)$ represents divergence, $N(\mu_i(x), \sigma_i(x))$ represents mean and variance of recovered data x on BN layer i, and $N(\mu_i, \sigma_i)$ represents mean and variance recorded in BN layer i. Furthermore, considering the image prior information, we got the prior loss as follows:

$$L_{pr}(x) = \alpha_1 L_V(x) + \alpha_2 L_{norm}(x) \qquad (9)$$

where $L_V(x)$ represents the variation of images, $L_{norm}(x)$ represents images' norm, and α_1, α_2 are hyper-parameters. Then, with the analysis above, we use the total loss L_{total} to reconstruct the training images from poisoned models for estimating Shapley value:

$$L_{total}(x) = \alpha CE(f(x), y) + \beta L_{bn}(x) + \gamma L_{pr}(x) \qquad (10)$$

where $CE(\cdot)$ represents the Cross Entropy loss, $f(\cdot)$ represents the trained model, y represents the target label to reconstruct images and α, β, γ are hyper-parameters.

Mixture-mode. Furthermore, because there is still a difference between clean images and recovered images, our experiments find that there is a larger accuracy degradation during data-free ShapPruning. Therefore, we propose a mixture-mode and try to combine information of Acc (accuracy) and ASR. We calculate Shapley value of Acc and ASR separately and find neurons with top-k ASR Shapley value and bottom-l Acc Shapley value to prune. And our experiments demonstrate that this way can help us locate the neurons which are only important to backdoor but not accuracy, locating poisoned neurons more accurately.

3.5. Pruning Based on Estimated Shapley Value

Finally, we summarize our framework illustrated in Fig. 1. We first use trigger reverse synthesis to get the reversed trigger and the target label. Then we inject the trigger into clean data (in the data-free situation, clean data is recovered from poisoned models), and get the reversed backdoor dataset. Then, using ASR as a measurement, we implement the accelerated Shapley value estimation method to get top-k neurons with the largest Shapley value and the estimation algorithm is shown in Appendix. Finally, we prune the target network with the top neurons, fine-tune the network with the clean data available, and offer backdoor-free networks to the users.

4. Experiment

We evaluate our backdoor defense method with five mainstream tasks against four common attacks, BadNets Attack [12], Trojan Attack [22], Physical Key Attack [3] and Input-Aware Attack [27] in data-insufficient situations on VGG [33] and ResNet [16], and design a series of experiments to test its effectiveness and robustness.

4.1. Experimental Setup

We compare ShapPruning with four existing methods Fine Pruning (FP) [21], Neural Cleanse (NC) [37], GangSweep (GS) [42] and DeepInspect (DI) [2] in the following five tasks (1).MNIST (2).CIFAR10 (3).CIFAR100 (4).GTSRB (5).YouTubeFace, which are commonly used in backdoor defense areas. Detailed information about the five datasets is shown in Appendix.

Attack configurations for BadNets. In our experiments, BadNets poisons images with a random colored square shown in Fig. 2. The trigger sizes are 5×5 or 10×10 to test our defense's robustness against different trigger sizes. We resize the images to 96×96, and thus, the trigger size is about 1% of the image size and injection ratio is 1%. Our experiments are based on a VGG-based model which is usually used in model compression tasks, whose structure is provided in Appendix. Also, we test ShapPruning on VGG11, VGG-Face (shown in Appendix) and ResNet to test its robustness on different architectures.

Attack configurations for Trojan Attack. We generate the trojan trigger shown in Fig. 5 with an initial square mask using gradient descent. Then, with the generated trigger, we fine-tune the trained model on the linear layers to inject backdoor into pre-trained models.

Attack configurations for Physical Key Attack. Physical Key Attack uses a pair of commodity glasses shown in Fig. 5 rather than a small square to inject backdoor into the model and can be more imperceptible.

Attack configurations for Input-Aware Attack. It uses a generator to generate sample-specific triggers. We set the Input-Aware Attack in all-to-one mode and attain a model with 99.41% ASR with the original setting in [27].

Available data. We suppose the defender can only get a small amount of clean data, specifically, one image per class in the few-shot setting *e.g.* only 10 images available for MNIST and CIFAR10. Furthermore, we propose a more strict condition in Sec. 4.5 where only the poisoned models are available but no clean data to help mitigate the backdoor.

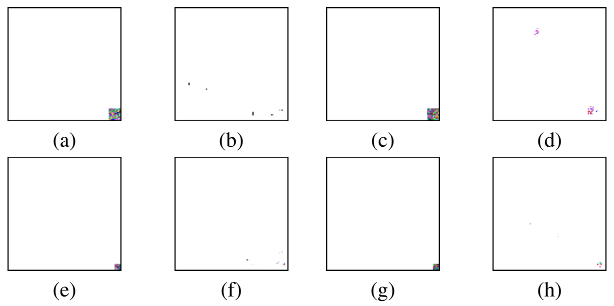


Figure 2. Original and reversed triggers in BadNets. (a), (c), (e), (g) are original triggers for CIFAR10, YouTubeFace, MNIST, GTSRB with the size of 10×10, 10×10, 5×5, 5×5 respectively. (b), (d), (f), (h) are reversed triggers for CIFAR10, YouTubeFace, MNIST, GTSRB using trigger reverse synthesis.

4.2. Shapley Pruning

In this subsection, we compare ShapPruning with other defense methods and prove its effectiveness.

Trigger reverse synthesis. We first use trigger reverse synthesis to get the reversed trigger in Figs. 2 and 5 where we find the reversed triggers and original triggers are in a similar location of the images, but have a relative difference in shape and color, caused by the insufficiency of data. Also, trigger reverse synthesis penalizes L1 norm, causing reversed triggers smaller than the original ones. These mismatches lead to some performance degradation in the backdoor mitigation compared with defense with the original trigger. However, despite the differences, ShapPruning can still locate the poisoned neurons precisely and mitigate backdoor with different trigger sizes.

Shapley value estimation. With the reversed poisoned data, we prune neurons in the order produced by ϵ-greedy and compute ASR decline in output iteratively, finding 50 average iterations are precise enough for estimating Shapley value and locating the poisoned neurons.

Pruning based on Shapley value. We compare our method with other three common methods including Fine Pruning (FP), Neural Cleanse (NC), and GangSweep (GS). From Tab. 1, ShapPruning mitigates backdoor best in poisoned models at the price of a tiny accuracy decline with only one image per class. On the contrary, other methods can't clean backdoor attacks with that few images. Especially, all the other defenses perform weaker in MNIST and CIFAR10 with only 10 clean images, which are fewer than the other two datasets, GTSRB and YouTubeFace.

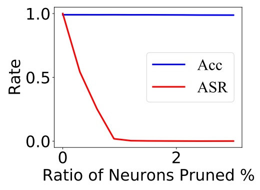
We suppose the poor performance of Neural Cleanse (NC) and GangSweep (GS) is caused by both the gap between the original and reversed trigger, and the weak generalization of training with a small amount of clean data. We explain the trigger gap's influence on mitigation degradation with the concepts from adversarial training. Neural Cleanse or GangSweep is similar to adversarial training [31, 40], which meets with performance degradation and poor generalization against different attacks. Thus, NC or GS, similar to adversarial training, behaves poorly with bigger trigger gaps, which is also found in [28]. Furthermore, we show Acc and ASR fluctuation during ShapPruning and Fine Pruning in Fig. 3. It demonstrates that ShapPruning can remove backdoor with only 1% of total neurons, compared with about 25% neurons removal in Fine Pruning. Fine Pruning, with that number of neurons pruned, may cause network structure changes and accuracy decline. Also, the insufficiency of clean data can weaken fine-tuning process and cause large accuracy fluctuation, especially in MNIST and CIFAR10 with only 10 images.

Defense against different attacks. We also defend against different attacks to test our method's robustness. We first show reversed triggers in Fig. 5, where an obvious gap between the original and reversed triggers is found. Based on the reversed triggers, we use different defense methods to mitigate backdoor, shown in Tab. 1. Also, we defend against Input-Aware Attack which injects sample-specific triggers into images to activate the backdoor. We find that with the trigger reverse, we can form a universal trigger pattern to activate backdoor and our experiments demonstrate that ShapPruning can mitigate Input-Aware Attack with only a few neurons pruned. And we also find that although there are different triggers for different samples, Input-Aware Attack still relies on a small number of sensitive neurons to activate backdoor and our method can find them precisely.

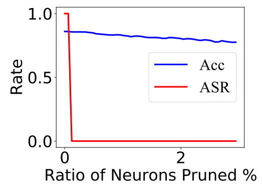
Time consumption. We conducted our experiments on Titan RTX GPU with 24GB memory and recorded time consumption for different methods in mitigating backdoor attacks. We compare our method with Neural Cleanse in GTSRB and find our method only consumes 585.95 seconds with 50 average iterations' Shapley estimation to get results in Tab. 1 after 671.13 seconds spent on trigger reverse. On the contrary, Neural Cleanse consumes 704.54 seconds which is 1.7x faster than our method. However, Neural Cleanse needs much more data and can't completely remove triggers in the few-shot setting. Furthermore, our method is time-saving and needs much less clean data compared with training a clean model from scratch.

4.3. Defense with Different Data Amounts

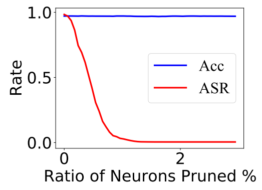
Previous methods need a large amount of clean data to mitigate backdoor, and thus, we want to explore the influence of the amount of clean data on backdoor mitigation. We compare mitigation results in Acc and ASR using Fine Pruning and ShapPruninig with different amounts of clean data in Fig. 4. Our experiment illustrates that backdoor mitigation performance improves with the amount of clean data rising and there is a significant fluctuation in ASR during



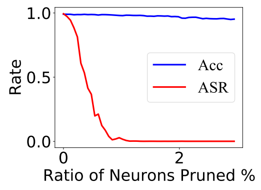
(a) ShapPruning on MNIST



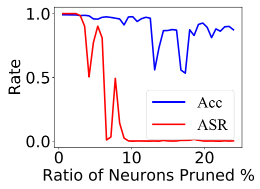
(b) ShapPruning on CIFAR10



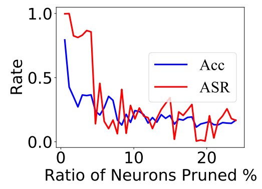
(c) ShapPruning on GTSRB



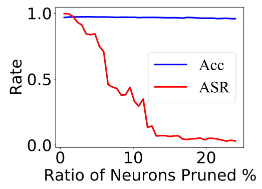
(d) ShapPruning on YouTubeFace



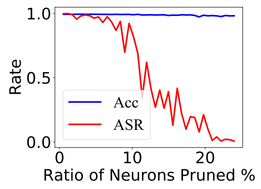
(e) Fine Pruning on MNIST



(f) Fine Pruning on CIFAR10



(g) Fine Pruning on GTSRB



(h) Fine Pruning on YouTubeFace

Figure 3. Acc and ASR fluctuation when pruning neurons guided by ShapPruning or Fine Pruning. (a)-(d) are for ShapPruning with max 3% of all the neurons pruned and (e)-(h) are for Fine Pruning with max 25% of all the neurons pruned.

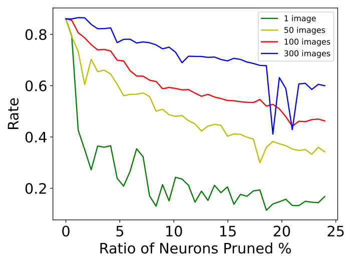
Benchmark	Before		FP [21]		NC [37]		GS [42]		ShapPruning		ShapPruning/o	
(%)	Acc	ASR	Acc↑	ASR↓	Acc↑	ASR↓	Acc↑	ASR↓	Acc↑	ASR↓	Acc↑	ASR↓
MNIST	99.02	100.00	97.00	3.02	98.64	29.87	95.30	80.33	98.99	0.34	99.06	0.56
CIFAR10	86.05	99.57	35.39	10.19	78.98	46.32	83.45	100.00	85.63	0.06	85.66	0.03
GTSRB	97.03	99.60	96.26	6.16	96.69	4.76	96.63	1.11	96.94	0.49	97.16	0.46
YouTubeFace	98.93	99.82	97.49	0.61	95.66	7.38	90.90	0.58	98.61	0.35	98.67	0.34
Input-Aware	99.41	99.37	98.12	2.66	99.32	43.55	88.85	32.05	99.29	0.15	99.35	0.24
Trojan Attack	97.08	92.06	16.32	2.56	95.01	2.01	96.33	10.91	96.03	0.98	96.44	0.64
Physical Key	98.39	100.00	90.70	0.05	98.49	64.34	97.21	54.21	95.94	0.60	97.26	0.08
ResNet-18	95.17	100.00	17.03	0.79	90.44	43.10	89.46	57.73	92.25	0.48	92.71	0.20
ResNet-34	98.37	99.98	97.93	0.19	98.65	0.25	56.84	6.55	98.49	0.07	98.51	0.05

Table 1. Different defenses methods against four common attacks and two common architectures (VGG and ResNet), where ShapPruning/o represents ShapPruning with original trigger. The first four lines show defenses against BadNets on four common datasets, the fifth to seventh lines show defenses against three different attacks (Input-Aware Attack on MNIST, Trojan Attack on GTSRB and Physical Key on YouTubeFace), and the eighth and ninth lines show defenses against ResNet (ResNet-18 on GTSRB and ResNet-34 on YouTubeFace). We record their Acc (higher is better) and ASR (lower is better) in the table.

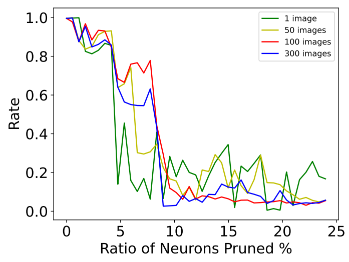
Fine Pruning with just 1 image per class. We attribute it to the lack of data to activate some normal neurons and there may be low activation values in many neurons, some of which are not poisoned. Furthermore, since data insufficiency weakens fine-tuning, Fine Pruning performance declines further. Similarly, ShapPruning with 300 images for each class performs best. But with the improvement of the data amount, backdoor mitigation is promoted to a relatively small extent. Furthermore our method performs the best in different experiments among all these defense methods with the same amount of data.

4.4. Acceleration Comparison

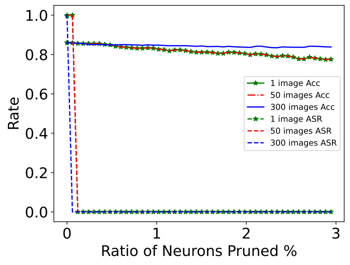
In this subsection, we compare our method of ϵ-greedy with T-MAB [9] which uses Bernstein error bounds [25,26] and find our method can more precisely and efficiently locate neurons with the largest top-k Shapley values under the same average iterations. We estimate Shapley value of neurons with 50 average iterations using these two methods. Meanwhile, we use the Monte-Carlo estimation of 5000 average iterations as the actual Shapley value for this task. We arranged neurons randomly before 30 average iterations and set ϵ to be 0.5 and 0.3 in 30-40 and after 40 iterations in ϵ-greedy. We then compare these two methods' top-50 neurons and find them whether in the top-70 neurons in the MC experiment. Our experiments find that there are 46 neurons in our methods' top-50 neurons in top-70 of actual value. On the contrary, there are only 27 neurons found in T-MAB. We attribute the inaccuracy of T-MAB to that Bernstein error bound is too conservative and consumes too much time to determine which neurons' Shapley values are too small to calculate. On the contrary, our method combines exploration and utilization, getting more accurate estimations of neurons to prune in just 50 average iterations.



(a) Acc for Fine Pruning



(b) ASR for Fine Pruning



(c) Acc and ASR for Shapley Pruning

Figure 4. Fine Pruning and ShapPruning with different sizes of datasets on CIFAR10. We test on 4 datasets with different amounts of clean data with 1 image, 50 images, 100 images, 300 images per class respectively for Fine Pruning and 1 image, 50 images, 300 images per class respectively for ShapPruning.

Benchmark	Before		FP [21]		NC [37]		DI [2]		ShapPruning	
(%)	Acc	ASR	Acc↑	ASR↓	Acc↑	ASR↓	Acc↑	ASR↓	Acc↑	ASR↓
CIFAR10	86.51	100.00	27.27	0.00	82.48	56.71	48.35	30.26	83.04	0.90
CIFAR100	62.08	99.98	35.39	10.19	47.12	27.82	53.76	81.44	59.51	0.89
Trojan Attack	97.08	92.06	22.49	71.32	95.17	2.82	84.43	33.24	96.21	0.14

Table 2. Different defense methods against different attacks and architectures in the data-free situation.

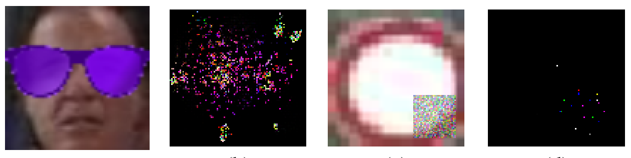


Figure 5. Trigger synthesis for Physical Key Attack and Trojan Attack. (a), (c) are examples of poisoned data for Physical Key Attack and Trojan Attack, and (b), (d) are the reversed trigger generated by trigger synthesis for Physical Key and Trojan Attack.

4.5. Data-free Backdoor Mitigation

The experiment results with the few-shot setting mentioned above demonstrates that ShapPruning is robust against different attacks and architectures. Then, we further introduce our ShapPruning framework into a data-free situation. Firstly, we try to reverse the training images from the poisoned model and show them in Fig. 6, where we compare our reversed images with model inversion attack [8] used in DeepInspect. Because DeepInspect's model inversion method is usually used in shallower networks *e.g*. multilayer perceptron, the recovery results degrade sharply in VGG or ResNet. Furthermore, the similarity between the recovered images and real images influences trigger reverse and neuron activation, thus deciding backdoor defense performance. With the help of information in batch normalization layers, our method reconstructs better images which human eyes can identity.

Then we utilized the backdoor mitigation methods on the

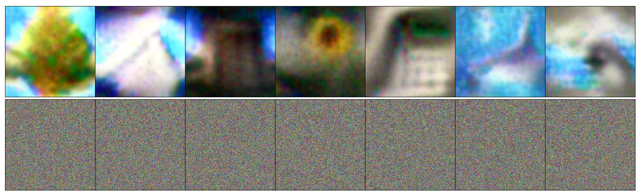


Figure 6. Images recovered for ShapPruning (Ours) and DeepInspect. The first line is ours and the second line is DeepInspect.

recovered images. Specifically, Fine Pruning (FP), Neural Cleanse (NC), and our ShapPruning method are conducted on our recovered images and DeepInspect (DI) is conducted with its original method. To test our methods' robustness, we evaluate on different attacks and architectures. We defend against BadNets on CIFAR10 and CIFAR100 with VGG16 and ResNet34 separately. Furthermore, we also defend on the different attack method *e.g*. Trojan Attack. The results are shown in Tab. 2. With better-recovered images and the robustness of mixture-mode ShapPruning, our method can mitigate backdoor clearly with only a small accuracy decline.

5. Conclusions

In this work, we propose Shapley Pruning framework to detect and mitigate backdoor attacks from poisoned models. Our method considers the interaction between neurons, locates the few infected neurons precisely, and protects models' structure and accuracy while pruning as many infected neurons as possible. Compared to prior work, our method mitigates backdoor successfully, using much fewer images (or even no clean data) and pruning much fewer neurons (about 1% of total neurons) than previous methods. Furthermore, we mitigate backdoor with only less than 1% accuracy decline in most situations. Also, our acceleration method, discarding threshold and ϵ-greedy, can effectively reduce time consumption and help complete most tasks in just several minutes. Our method needs to reverse backdoor triggers for computing ASR in estimating Shaley value, which may be time-consuming. A more efficient and direct way is to use clean data for computing Shapley value to find the neurons in the model which contribute most to backdoor attacks, and we leave it to future work.

References

[1] Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009. 2, 3

[2] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In *IJCAI*, 2019. 2, 5, 8

[3] 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. 5

[4] Yoojin Choi, Jihwan Choi, Mostafa El-Khamy, and Jungwon Lee. Data-free network quantization with adversarial knowledge distillation. In *CVPR*, 2020. 5

[5] Min Du, Ruoxi Jia, and Dawn Song. Robust anomaly detection and backdoor attack detection via differential privacy. In *ICLR*, 2019. 2

[6] Zine el abidine Kherroubi, Samir Aknine, and Rebiha Bacha. Leveraging on deep reinforcement learning for autonomous safe decision-making in highway on-ramp merging (student abstract). In *AAAI*, 2021. 1

[7] Alan M Ferrenberg and Robert H Swendsen. Optimized monte carlo data analysis. *Computers in Physics*, 3(5):101–104, 1989. 3

[8] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *SIGSAC*, 2015. 3, 8

[9] Amirata Ghorbani and James Zou. Neuron shapley: Discovering the responsible neurons. *arXiv preprint arXiv:2002.09815*, 2020. 2, 7

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 2

[11] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 1

[12] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017. 1, 5

[13] 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. 2

[14] Matan Haroush, Itay Hubara, Elad Hoffer, and Daniel Soudry. The knowledge within: Methods for data-free model compression. In *CVPR*, 2020. 5

[15] Matthew Hausknecht and Peter Stone. The impact of determinism on learning atari 2600 games. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. 4

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5

[17] Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina CS Valentim, Huiying Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131, 2018. 1

[18] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009.

[19] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Anti-backdoor learning: Training clean models on poisoned data. *arXiv preprint arXiv:2110.11571*, 2021. 2

[20] Yiming Li, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *arXiv preprint arXiv:2007.08745*, 2020. 1

[21] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 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, 2, 5, 7, 8

[22] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. *NDSS*, 2018. 5

[23] Yuchen Liu, Jiajun Zhang, Hao Xiong, Long Zhou, Zhongjun He, Hua Wu, Haifeng Wang, and Chengqing Zong. Synchronous speech recognition and speech-to-text translation with interactive decoding. In *AAAI*, 2020.

[24] David Mascharka, Philip Tran, Ryan Soklaski, and Arjun Majumdar. Transparency by design: Closing the gap between performance and interpretability in visual reasoning. In *CVPR*, 2018. 1

[25] Andreas Maurer and Massimiliano Pontil. Empirical bernstein bounds and sample variance penalization. *arXiv preprint arXiv:0907.3740*, 2009. 7

[26] Volodymyr Mnih, Csaba Szepesvári, and Jean-Yves Audibert. Empirical bernstein stopping. In *ICML*, 2008. 7

[27] Tuan Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. In *NeurIPS*, 2020. 5

[28] 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. 6

[29] Elan Rosenfeld, Ezra Winston, Pradeep Ravikumar, and Zico Kolter. Certified robustness to label-flipping attacks via randomized smoothing. In *ICML*, 2020. 2

[30] Peter J Rousseeuw and Christophe Croux. Alternatives to the median absolute deviation. *Journal of the American Statistical association*, 88(424):1273–1283, 1993. 4

[31] Ali Shafahi, Mahyar Najibi, Zheng Xu, John Dickerson, Larry S Davis, and Tom Goldstein. Universal adversarial training. In *AAAI*, 2020. 6

[32] LS Shapely. A value for n-person games. contributions to the theory of games, 1953. 2, 3

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5

[34] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks. In *NeurIPS*, 2017. 2

[35] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014. 1

[36] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In *NeurIPS*, 2018. 2

[37] 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)*, pages 707–723. IEEE, 2019. 1, 2, 4, 5, 7, 8

[38] Michael Wunder, Michael L Littman, and Monica Babes. Classes of multiagent q-learning dynamics with epsilon-greedy exploration. In *ICML*, 2010. 3

[39] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deep-inversion. In *CVPR*, 2020. 5

[40] Haichao Zhang and Jianyu Wang. Defense against adversarial attacks using feature scattering-based adversarial training. In *NeurIPS*, 2019. 6

[41] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *CVPR*, 2020.

[42] Liuwan Zhu, Rui Ning, Cong Wang, Chunsheng Xin, and Hongyi Wu. Gangsweep: Sweep out neural backdoors by gan. In *ACM MM*, 2020. 1, 2, 5, 7

6. Appendix

6.1. Overview

We mainly show detailed experimental settings, supplementary experiments, and the proof in Appendix, which demonstrates our method's effectiveness and robustness against different models and attacks.

6.2. Experimental Settings

Model structure. Our VGG-based model is illustrated in Tab. 3 with input shape 96×96 and this network can attain state-of-the-art performance on mainstream tasks.

Layer	Channels	Filter size	Stride	Activation
Conv	96	3	1	ReLU
MaxPool	96	3	2	-
Conv	128	3	1	ReLU
MaxPool	128	3	2	-
Conv	160	3	1	ReLU
MaxPool	160	3	2	-
Conv	256	3	1	ReLU
Conv	256	3	1	ReLU
MaxPool	256	3	2	-
Conv	384	3	1	ReLU
Conv	384	3	1	ReLU
MaxPool	384	3	2	-
FC	1024	-	-	ReLU
FC	class	-	-	-

Table 3. Network structure

Dataset	labels	Image size	Training Images
MNIST	10	28x28x1	60,000
CIFAR10	10	32x32x3	60,000
CIFAR100	100	32x32x3	60,000
GTSRB	43	32x32x3	35,288
YouTubeFace	1,283	55x47x3	375,645

Table 4. Detailed information about datasets

Datasets. Furthermore, we show the five datasets used in our experiments in Tab. 4.

Computing infrastructures. The experiments are conducted on Linux servers equipped with an Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz, 256GB RAM and 8 NVIDIA TITAN RTX GPUs (with 24GB memory each). All models are implemented in PyTorch version 1.8.0 with CUDA version 11.4, Python 3.7. Our code and datasets are provided in the code.zip. The experiments are conducted five times for Neural Cleanse, Fine Pruning, GangSweep, DeepInspect, and ShapPruning on each setting and we choose the best result as the methods' final performance.

6.3. Shapley Pruning

Shapley estimation algorithm. We summarize our neurons' Shapley value estimation method in Algorithm 1. $avg_{itr}(\cdot)$ represents the function calculating the average iteration for all neurons in iteration t, O^t represents the permutation order at iteration t and $O^t(j)$ represents the j-th neuron in that permutation.

Physical Key Attack. We demonstrate Acc and ASR fluctuation of Fine Pruning and ShapPruning during neuron pruning against Physical Key Attack in Fig. 7.

Testing on different models. We also defend against BadNets on different models, including VGG11 and VGG-Face, to test our method's robustness on different model structures. We test on VGG11 in GTSRB and VGG-Face in YouTubeFace, and inject triggers with 1% trigger injection ratio. Acc and ASR fluctuation of ShapPruning and Fine Pruning is illustrated in Fig. 8 and Fig. 9, where Acc and ASR are given.

Algorithm 1 Accelerated Backdoor Shapley Estimation

Input: Network's neurons $N = \{1, \ldots, n\}$, discarding threshold $t_{discard}$, total average iteration I, average iteration threshold for randomly choosing l, top neurons in ϵ-greedy m

Output: top-k neurons with largest Shapley value

```
1:  Let $t = 1$.
2:  while $avg_{itr}(t) \leq I$ do
3:    if $avg_{itr}(t) \leq l$ then
4:      Randomly permutate $N$ and get pruning order $O^t$
5:    else
6:      Divide $N$ into two sets $N_1$ (top-$m$) and $N_2$
7:      for $num$ in $1 \ldots n$ do
8:        $O^t(num) = \epsilon - greedy(N_1, N_2)$
9:      end for
10:   end if
11:   for $j$ in $1 \ldots n$ do
12:     if $ASR(O^t(j-1) \ldots O^t(n)) \geq t_{discard}$ then
13:       $margin^t_{O^t(j)} = ASR(O^t(j) \ldots O^t(n)) - ASR(O^t(j+1) \ldots O^t(n))$
14:     else
15:       Break
16:     end if
17:     $\phi = Average(margin)$
18:   end for
19:   $t \leftarrow t + 1$
20: end while
21: Choose top-$k$ neurons with largest Shapley value
```

Mixture-mode ShapPruning. Compared with normal ShapPruning, mixture-mode ShapPruning can combine information of Acc and ASR, and maintain the model's accuracy as much as possible. To test our mixture-mode ShapPruning's effectiveness in a data-free situation, we defend against BadNets on CIFAR10 and CIFAR100 with VGG16 and ResNet34 separately. Furthermore, we also defend on a different attack method, Trojan Attack. The results are shown in Tab. 5. Compared with normal ShapPruning, our mixture-mode ShapPruning can mitigate backdoor with a smaller Acc decline without clean data.

6.4. Backdoor Model Detection

We conduct our backdoor model detection experiments on MNIST, CIFAR10, GTSRB and YouTubeFace, and train clean and backdoor models on each of them. Firstly, we show the L1 norm of the reversed triggers for each class in

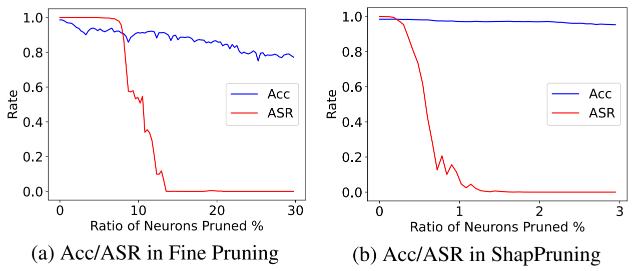


Figure 7. Acc and ASR fluctuation against Physical Key Attack during Fine Pruning and ShapPruning.

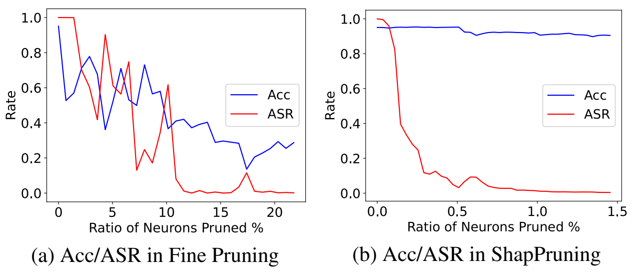


Figure 8. Fine Pruning and ShapPruning in GTSRB based on VGG11. Our poisoned model is trained with 95.05% Acc and 100.00% ASR initially. Then we use Fine Pruning and ShapPruning to mitigate backdoor attacks. Fine Pruning finally gets 42.02% Acc and 1.18% ASR, while ShapPruning gets 95.24% Acc and 1.67% ASR.

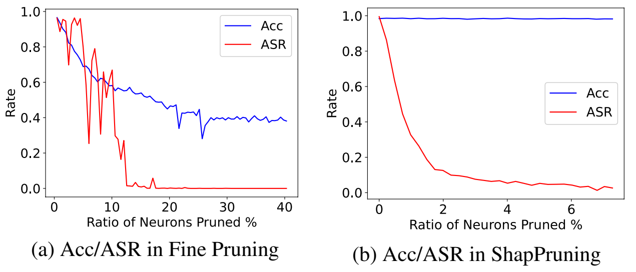


Figure 9. Fine Pruning and ShapPruning in YouTubeFace based on VGG-Face. Our poisoned model is trained with 98.40% Acc and 99.50% ASR initially. Then we use Fine Pruning and ShapPruning to mitigate backdoor attacks. Fine Pruning finally gets 55.35% Acc and 1.50% ASR, while ShapPruning gets 98.20% Acc and 1.63% ASR.

Fig. 10, which demonstrates that the L1 norm for the target label is much smaller than the other reversed triggers. Then, we use this finding to detect the backdoor target label.

Furthermore, we also propose MAD-small and compare it with the traditional MAD detection method, illustrated in Fig. 11. For traditional MAD, some triggers don't converge to a small L1 norm, which causes a large outlier and false positive for backdoor model detection. In our experiments, we use detection index, defined as $I_{detect} =$

Benchmark	Before		Mix-ShapPruning		ShapPruning	
(%)	Acc	ASR	Acc↑	ASR↓	Acc↑	ASR↓
CIFAR10	86.51	100.00	83.04	0.90	81.42	1.74
CIFAR100	62.08	99.98	59.51	0.89	56.34	1.21
Trojan Attack	97.08	92.06	96.21	0.14	94.82	0.67

Table 5. Mixture-mode ShapPruning compared with normal ShapPruning on data-free backdoor mitigation, where Mix-ShapPruning represents mixture-mode ShapPruning

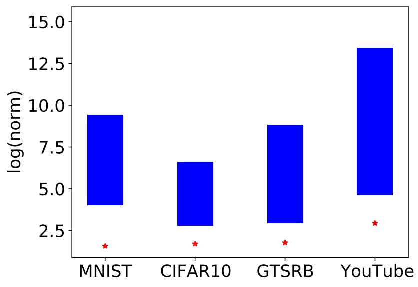
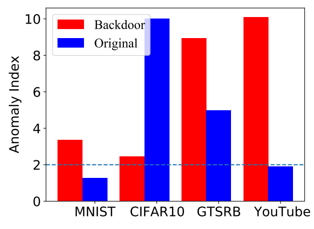
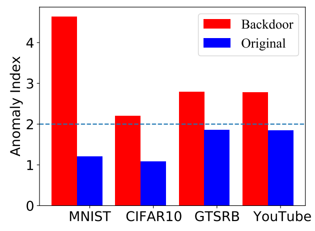


Figure 10. The L1 norm of the reversed triggers in the backdoor models. $*$ represents the L1 norm for the target label and the blue bars represent the range of the L1 norm of the reversed triggers for the other labels. We find that the L1 norm for the target label is much smaller than the other triggers and we can get the target label with this finding.



(a) MAD



(b) MAD-small

Figure 11. Backdoor detection with MAD and MAD-small. MAD-small represents using deviation set D_{small} to estimate σ and detect outlier. The horizontal dotted line represents max bound for outlier detection and models with MAD larger than that are classified as the backdoor models.

$max(d_i/\sigma), \quad d_i = d_1 \cdots d_n$ to judge whether the models are poisoned, where d_i is the absolute deviation between triggers' L1 norm and their median. If I_{detect} is larger than the max bound, we will judge that this model is poisoned. Obviously, we need a fixed max bound for all experiments and datasets to judge whether the models are poisoned. Then, we analyze the results of these two detection methods. In CIFAR10, I_{detect} of the clean model, calculated with traditional MAD, is larger than I_{detect} of the poisoned model. Furthermore, there isn't a fixed max bound for anomaly detection, which varies in different models in traditional MAD. Since we don't know the specific max bound for the datasets in advance, we can't detect backdoor successfully. On the contrary, our method, detection with D_{small}, uses the max bound setting to 2 to distinguish the clean and backdoor models successfully in all four datasets.

6.5. Proof for MAD

MAD is the median of absolute deviation and has a strong relationship with the distribution's standard deviation σ in the normal distribution. From the definition, we have the following equation:

$$P(|x - median| \leq MAD) = \frac{1}{2} \tag{11}$$

Then, when the number of samples is large enough, $median = \mu$, and μ is the average of the distribution. We have the equation as follows:

$$P(\frac{|x-\mu|}{\sigma} \leq \frac{MAD}{\sigma}) = \frac{1}{2} \tag{12}$$

Furthermore, we can use the standard normal distribution's cumulative probability distribution Φ to present the above equation:

$$\Phi(0 \leq x \leq \frac{MAD}{\sigma}) = \frac{1}{4} \tag{13}$$

Then, we can calculate σ from MAD as follows:

$$\sigma = \frac{MAD}{\Phi^{-1}(\frac{3}{4})} = 1.4826 \cdot MAD \tag{14}$$

Moreover, because we suppose that the L1 norm of the reversed triggers for the non-target label obeys normal distribution, their norm should be in a range around their average μ with a probability. From the above analysis, for a poisoned model, d_i/σ of the non-target label can be treated as the absolute value of the standard normal distribution. Under probability p, the absolute value of the standard normal distribution should be smaller than the max bound $\Phi^{-1}(\frac{p+1}{2})$. Then, d_i/σ can be larger than the max bound only with probability $1-p$. Thus, d_i/σ larger than the max bound can be treated as an outlier and does not obey the normal distribution with the confidence probability p.