Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 1974822, 9 pages
https://doi.org/10.1155/2021/1974822

*Research Article*

# VarDefense: Variance-Based Defense against Poison Attack

**Mingyuan Fan, Xue Du, Ximeng Liu, and Wenzhong Guo**

*College of Computer and Data Science, Fuzhou University, 350108, China*

Correspondence should be addressed to Wenzhong Guo; guowenzhong@fzu.edu.cn

Received 4 August 2021; Revised 18 October 2021; Accepted 12 November 2021; Published 9 December 2021

Academic Editor: Lei Chen

The emergence of poison attack brings a serious risk to deep neural networks (DNNs). Specifically, an adversary can poison the training dataset to train a backdoor model, which behaves fine on clean data but induces targeted misclassification on arbitrary data with the crafted trigger. However, previous defense methods have to purify the backdoor model with the compromising degradation of performance. In this paper, to relieve the problem, a novel defense method VarDefense is proposed, which leverages an effective metric, i.e., variance, and purifying strategy. In detail, variance is adopted to distinguish the bad neurons that play a core role in poison attack and then purifying the bad neurons. Moreover, we find that the bad neurons are generally located in the later layers of the backdoor model because the earlier layers only extract general features. Based on it, we design a proper purifying strategy where only later layers of the backdoor model are purified and in this way, the degradation of performance is greatly reduced, compared to previous defense methods. Extensive experiments show that the performance of VarDefense significantly surpasses state-of-the-art defense methods.

## 1. Introduction

Recently, deep neural networks (DNNs) have obtained impressing achievements over a broad spectrum of domains, such as image recognition [1], natural language process [2], recommendation system [3], and others [4–6]. Unfortunately, when DNNs are used in security-critical or data-sensitive scenarios, potential benefits arouse the adversaries to attack the models.

Over the past decade, many attack methods have been developed. For example, model extraction attacks [7] expose the valuable information of the model, model inversion attacks [8] reconstruct the secret data, and evasion attack [9] makes the model output false predicts. Despite the fact that those attacks are dangerous, poison attack still is the most threatening attack due to its high attack feasibility compared to those attacks. Poison attack can be effortlessly conducted by mixing a small volume of poison data into the training dataset, and a backdoor will be planted in any model trained with the poison dataset. Subsequently, this backdoor is activated during model inference by a trigger crafted by the attacker which, whenever and wherever present in a given image, enforces the DNN to predict the poison label. The capacity of manipulating the output of the DNN can produce a severe consequence in many fields. For instance, it is extremely terrible that the autopilot system [10] mistakenly recognizes the pedestrian as something else and similar events happen in face recognition [11], etc. In short, poison attack makes DNNs unreliable and greatly drags the application of DNNs in the physical world.

For promoting applications of DNNs in scenarios with high-security requirements, several relatively effective defense approaches have been proposed to conquer poison attack over the past years. These defense methods generally are two-stage methods, including the detection stage and purifying stage. The detection stage is aimed at determining whether a DNN contains a backdoor. In detail, optimization is adopted to derive a trigger for each class and then inspecting if there is a trigger with exceptional distinctness than other triggers. The existence of a trigger with exceptional distinctness probably suggests that the model is poisoned. By adopting the idea, the detection stage obtains an impressive capacity of detecting the backdoor. In contrast, the performance of such methods focused on purifying stage is disappointing. Specifically, with such methods, despite the backdoor being fairly erased from the model, the induced loss in performance of the model is always compromising, extremely lowering the effectiveness of such approaches. This paper proposes a novel defense method, dubbed variance-based defense against poison attack (VarDefense), to reduce the impairment induced by removing the backdoor from the model.

VarDefense is motivated by the fact that the backdoor embedded in the model essentially is the neurons responding strongly to the trigger (named bad neurons), which contribute the most of the attack effectiveness. Hence, we desire to identify and then remove the bad neurons, and in this way, the performance of the model can be reserved as possible while the backdoor is quite removed. However, the information about the trigger pattern does not access in advance, and then, it is intractable to recognize the bad neurons directly. To circumvent the problem, we embrace the approach with a similar function that reserves the neurons that produce a great effect for clean data (named benign neurons). Moreover, in Approach, it is intuitively justified that the way can achieve the identical performance to the method of directly identifying the poison neurons. Extensive experiments show that the performance of VarDefense significantly surpasses state-of-the-art defense methods. Our contributions are summarized as follows:

(i) We suggest adopting variance as the purifying metric, which is more efficient compared to previous purifying metrics. Then, based on the variance, we propose a novel defense method VarDefense against the poison attack

(ii) We propose an insightful perspective to understand the mechanism of the poison attack and design a simple yet efficient purifying strategy that can greatly reduce the degradation of the model's performance

(iii) We conduct extensive experiments on four widely used benchmark datasets, i.e., MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100, to validate the effectiveness of VarDefense, where the experiment results demonstrate the superior performance of VarDefense than state-of-the-art defense methods

## 2. Related Works

*2.1. Backdoor Attack.* In the life cycle of DNNs, there are many potential threat factors used to attack, and poison attack is conducted during the training process. The seminal work of poison attack was proposed by Gu et al. [12], where authors just added a simple trigger pattern, e.g., a small white rect, into a small portion of training data, and then a backdoor is strongly injected into the model. Specifically, the model trained with the poison dataset classified poison input into a poison label over 99% success rate, without degradation of performance. Following Gu et al. [12], Chen et al. [13] argued that, for a practical attack, the images with the crafted trigger should be indistinguishable compared to its benign version. To do so, Chen et al. [13] proposed a blended strategy in which poison inputs are produced by blending the specific random noise with clean images. It greatly decreases the risk of being checked out by humans. Xie et al. [14] and Bhagoji et al. [15] extend the poison attack into the federated learning scenario, which further increases the threat of poison attack.

*2.2. Backdoor Defense.* With the emergence of poison attacks, several countermeasures were proposed, and such countermeasures can be roughly classified as detection methods and purifying methods.

Detection methods focus on recognizing whether a model is poisoned. Neural cleanse [16] has taken the first step for detecting whether a model contains a backdoor. In detail, the possible triggers for each class are reversed and then $L_1$ norm as the metric to inspect the trigger with exceptional features. The model is infected if there is a trigger with exceptional features. Despite the impressive performance of neural cleanse, it still requires a clean dataset. To further alleviate the constraint, Chen et al. [17] proposed DeepInspect to detect the backdoor without resort to a clean dataset, which obtains competitive performance compared to neural cleanse. There are some ambiguous approaches that are aimed at detecting the poison examples, and we also classify the approaches into the detection methods. For example, Gao et al. [18] proposed to filter poison examples through overlapping example patterns and inspect the output. If the randomness of the output is too small, the examples probably are poisoned.

In contrast to detection methods, purifying methods are aimed at removing the backdoor while maintaining the performance as possible. Liu et al. [19] proposed a simple way against poison attack, where the model is fine-tuning with benign examples. But the performance of the method is fairly low. Similar to it, Qiao et al. [20] proposed fine-tuning the model with the mixed dataset, which consisted of the clean data and the poison data (synthesized by the generative model) with the clean label other than the poison label. To remove the backdoor, Liu et al. [21] proposed fine-pruning, a model compression method. Fine-pruning is done by removing the neurons with low activation value w.r.t. clean data, but some bad neurons remain in the model. Besides, based on the idea that the model is not poisoned, definitely, if it is trained from scratch over clean data, Yoshida and Fujino [22] proposed knowledge distillation for removing backdoor, where clean labels can be predicted by the teacher (poison) model. Furthermore, Li et al. [23] proposed neural attention distillation to improve the effectiveness of vanilla knowledge distillation, becoming the state-of-the-art purifying method currently. However, the effectiveness of such methods heavily depends on the number of clean data and performs unstably over different datasets.

## 3. Scenario

This section gives an overview of the attack and defense scenario, beginning with attack scenario, followed by defense assumptions and goals.

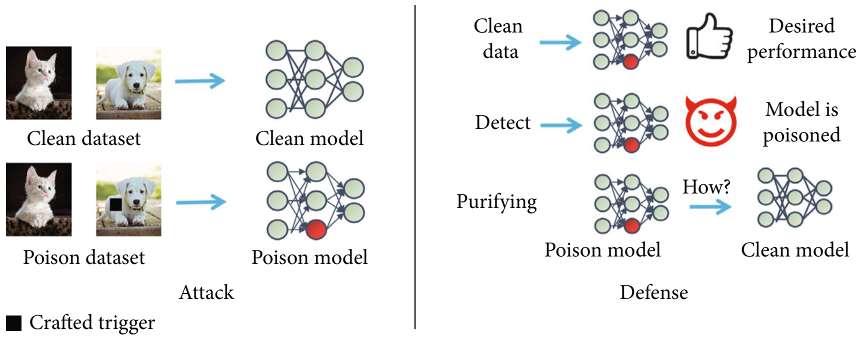


FIGURE 1: The illustration of the attack and defense scenario.

*3.1. Attack Scenario.* We assume that attackers poison the training dataset in some certain ways, and then, the model is trained with the poison dataset, as shown in Figure 1(a). Then, the (poisoned) model performs well on clean data but exhibits targeted misclassification on the presence of poison data, i.e., a backdoor is injected into the model. This attack scenario is fairly common in practice because the training dataset is always not fully controlled by users. To illustrate, there are three typical cases. The first case is that the collected dataset may contain poison data crafted and spread by the attackers. Another case is that, in federated learning, the malicious participants are desired to poison their own dataset to manipulate the shared model. The third case is that, due to deficiency of computing resource, the users outsource the training process to an untrusted party and the party probably poisons the dataset to potential benefits. In short, the assumed attack scenario covers most of the attack scenario of the poison attack.

*3.2. Defense Scenario.* Now, the defenders (users) receive the (poisoned) model and there are two required tasks for defenders: validating the performance and detecting whether the model is poisoned. To validate the performance of the model, it is commonly supposed that the defenders have a small volume of clean data, i.e., test set. Moreover, we highlight that the test set is identically distributed with the test size, which is a widely used assumption in most backdoor defense papers [19–23] because it is impossible to adopt a test set outside the distribution of training set to measure the performance of the model. Then, as demonstrated in Figure 1(b), the defender attempt begins to validate the performance and the model produces desired performance on the dataset available to the defenders. Besides, we assume that the defenders has been ensured that a backdoor is embedded into the model (the step can be smoothly implemented with previous detection methods), and the remaining problem is how to purify the model. Specifically, the purifying method should fulfill two goals: (1) rendering the backdoor disable with slight performance loss and (2) without resort any extra resources except the holding dataset. In the next section, we will develop a method to meet the two goals.

*3.3. Approach.* In this section, we design the purifying approach VarDefense to address the poison attack, starting with the core idea of VarDefense; implementation details are revealed subsequently.

*3.4. Overview.* The backdoor embedded in the model essentially is the bad neurons that are implicitly coopted by the attack to recognize the crafted trigger. According to the idea, removing the backdoor from the model basically equals removing those bad neurons. Besides, the bad neurons are dormant in the presence of clean data, suggesting that recognizing the bad neurons can be realized by using clean data. Therefore, we assume that the defenders possess a small volume of clean data, which also is a widely used hypothesis in most defense scenarios as forementioned in Scenario. The leaving problem is how to implement the idea leveraging the clean data. Intuitively, the idea can be implemented as follows: (1) feeding a batch of clean data into the network, (2) adopting a certain predefined selection metric to measure the importance of neurons to the data, and (3) based on it, determining the neurons that should be purified, i.e., recognizing the bad neurons. In the next paragraphs, we first define the selection metric and then introduce the purifying strategy and implement details.

*3.5. Variance-Based Selection Metric.* In general, there are two types of selection metrics that are available to evaluate the importance of neurons: weight-based metrics and output-based metrics. Weight-based metrics utilize the weights of neurons for assessing the importance of neurons, while output-based metrics adopt the outputs of neurons to evaluate the importance of neurons. However, weight-based metrics are unsuitable to our method. Specifically, notice that the bad neurons must not be associated with small weights, which implies only removing the neurons with small weight being not enough to erase the backdoor.

In contrast to weight-based metrics, output-based metrics are favorable. In fact, the neurons of the outputs with the relatively small change to the deviation of the clean data generally contribute less in terms of clean data, since the outputs are approximately constant to the variation in the inputs, suggesting the neurons are probably redundant, i.e., the bad neurons. Hence, such metrics based on sensitivity can be exploited to distinguish the beneficial neurons from the bad neurons. There are two such representative metrics: variance and information entropy. Variance assesses how divergent a set of contiguous numbers is while information entropy measures the redundancy of a set of numbers. But, when vanilla information entropy is adopted to measure the divergence of contiguous data, discretization is the necessary data preprocessing procedure. Discretization involves how to bin that is not being well-handled till now. In sharp contrast, variance can inherently estimate the divergence of the contiguous variables without resort to any preprocessing trick. Moreover, the computation overhead required to variance is significantly lower than information entropy, considering the complicated process flow and log function being the high computation cost involved in information entropy. Therefore, variance is a better metric than information entropy.

### *3.6. Purifying Strategy and Implementation Details*

*3.6.1. Purifying Strategy.* Recall that our goal is removing the backdoor while maintaining the performance as possible, and to fulfill the goal as possible, we have to bespeak a proper purifying strategy. The purifying strategy can be divided based on four factors: (1) the number of purifying layers, (2) purifying earlier layers or later layers, and (3) layer-wise purifying or network-wise purifying. It should be clarified that layer-wise purifying and network-wise purifying differ in purifying the neurons with the metric below the predetermined threshold within one certain layer or the network. For VarDefense, purifying multiple and later layers, and layer-wise purifying are shown in Figure 2. There are reasons for it. A piece of shared knowledge is that shallow and deep layers of DNNs are responsible for recognizing general and specific features, respectively. Besides, the trigger is a specific feature, which implies the backdoor is the neurons located in later layers. Therefore, purifying the later layers is enough for fairly removing the backdoor. Another benefit is to avoid the huge damage to the model induced by pruning earlier layers (all of the general features matter). For the third question, layer-wise purifying is advisable because the weights and outputs of neurons in different layers generally have different orders of magnitude. In other words, network-wise purifying is prone to purify the neurons in the layers with higher output magnitude.

*3.6.2. Implementation Details.* Now, the remaining problem is to determine the threshold for classifying neurons and how to purify the neurons below the threshold. We begin with solving the first problem. Intuitively, the fixed threshold strategy is not feasible since it is difficult to ensure a specified threshold where the backdoor is completely removed with desired loss of performance of the model. To conquer the dilemma, the dynamic incremental threshold strategy is introduced in VarDefense: initializing the threshold from a small value and iteratively raising it, until the threshold meets the predetermined maximum value. To overcome the second problem, we design an appropriate loss function involving crossentropy loss and $L_1$ regularization term-associated weights of neurons for purifying the bad neurons. On the one hand, the regularization term is adopted to remove the backdoor fairly. On the other hand, the crossentropy loss function is aimed at fine-tuning for decreasing the damage induced by purifying, i.e., avoiding compromising performance loss. Specifically, the loss function is defined as follows:

$$L = \frac{1}{n}\sum_{i=1}^{n}\left(\alpha \bullet \mathrm{Cross}(y_i, F(x_i)) + \sum_{j\in\mathscr{M}} \left\| w^j \right\|_1\right), \qquad (1)$$

where $\alpha$ is the hyperparameter that balances two loss terms, $\mathscr{M}$ is purifying layers set, and $w^j$ denotes all parameters of the $j$th layer.

*3.6.3. Momentum Trick.* The effectiveness of VarDefense heavily depends on the quality of clean data. In fact, to accurately evaluate the effect of a neuron for clean data, the importance of the neuron should be measured by all clean data simultaneously. But the solution has to consume a huge cost and requires advanced equipment, which is a harsh condition to normal users. To overcome the issue, we utilize the widely used momentum trick. In detail, by accumulating the previous values, the momentum trick can effectively reduce the bias of minibatch data, greatly improving the effectiveness of the purifying process.

## 4. Experiments

### *4.1. Experiment Setting*

*4.1.1. Attacks.* Poison attacks mainly differ in the characteristic of the crafted trigger, e.g., size and position. To comprehensively evaluate the effectiveness of VarDefense to various poison attacks, both the size and position of the crafted trigger are covered in the experiments. For trigger size, we consider the four classic forms, including $3 \times 3$, $5 \times 5$, $7 \times 7$, and global noise. Meanwhile, for trigger position, the fixed position and random position are involved. Finally, the poison ratio is set to widely used 0.05, which is the fraction of poison data added per minibatch.

*4.1.2. Baselines.* To validate the performance of VarDefense, three mainstream types of state-of-the-art defense method is considered, including fine-pruning [21], GAN-based defense (the authors did not name their defense method, and thus, we use GAN-based defense to denote it) [20], and neural attention distillation (NAD) [23].

*4.1.3. Datasets and Model Architecture.* The performance of defense methods against attacks is measured in four benchmark datasets, i.e., MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100, while ResNet18 is the base model throughout the experiments.

*4.1.4. Evaluation Metric.* Followed by previous studies [20, 21, 23], we adopt the two widely used metrics to evaluate the effectiveness of VarDefense, i.e., accuracy (ACC) and attack success rate (ASR). ACC refers to the accuracy of the model to clean data, measuring the degradation of the performance of the model. ASR is the portion of poison data classified into the poison label, evaluating the attack effectiveness. In short, the higher the ACC is, the lower the ASR is, and the better the defense method is.

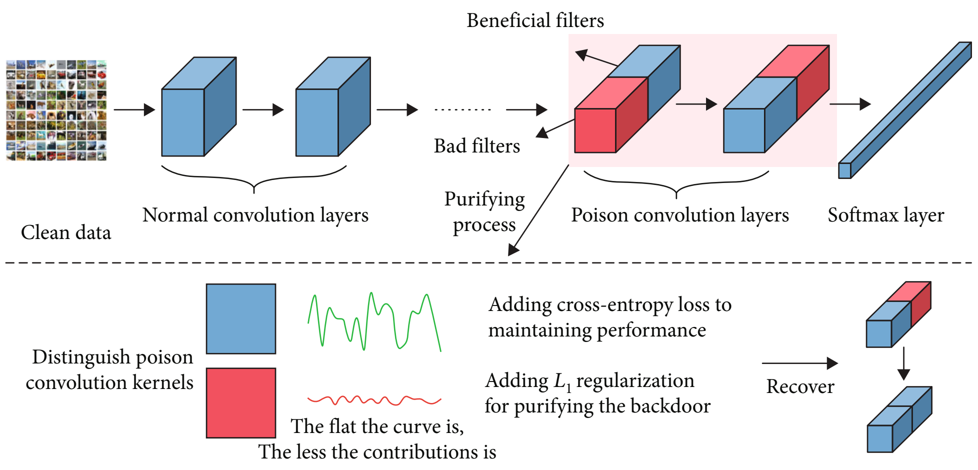

Beneficial filters
Bad filters
Clean data
Normal convolution layers
Purifying process
Poison convolution layers
Softmax layer
Distinguish poison convolution kernels
Adding cross-entropy loss to maintaining performance
Adding $L_1$ regularization for purifying the backdoor
Recover
The flat the curve is,
The less the contributions is

Figure 2: The running process of VarDefense. The images (clean data) are extracted from public dataset CIFAR-10 that can be referred to http://www.cs.toronto.edu/~kriz/cifar.html.

Table 1: The performance of four defense methods against different attack settings in MNIST and Fashion-MNIST. The numbers in italic are the best result.

| Dataset | Trigger size | Position | Before | | Fine-pruning | | NAD | | GAN-based defense | | VarDefense | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | ASR | ACC | ASR | ACC | ASR | ACC | ASR | ACC | ASR | ACC |
| MNIST | 3 × 3 | Fixed | 99.05 | 98.67 | 11.47 | 96.70 | 1.86 | 98.24 | 7.48 | 97.59 | *0.58* | *97.25* |
| | | Random | 99.52 | 98.87 | 11.53 | 96.42 | 2.14 | 98.05 | 7.36 | 98.00 | *0.54* | *98.52* |
| | 5 × 5 | Fixed | 99.24 | 98.74 | 10.95 | 97.52 | 2.43 | 97.74 | 6.79 | 96.08 | *0.57* | *97.91* |
| | | Random | 99.53 | 98.59 | 11.02 | 97.33 | 3.11 | 97.13 | 6.66 | 96.31 | *1.10* | *97.16* |
| | 7 × 7 | Fixed | 99.17 | 99.72 | 13.70 | 96.94 | 2.26 | 97.36 | 11.75 | 95.54 | *0.71* | *98.06* |
| | | Random | 99.87 | 99.00 | 13.98 | 97.42 | 2.64 | 97.35 | 11.32 | 95.01 | *0.80* | *97.60* |
| | Global noise | — | 98.75 | 98.83 | 10.33 | 97.19 | 2.65 | *98.12* | 7.13 | 95.91 | *0.93* | 97.79 |
| Fashion-MNIST | 3 × 3 | Fixed | 99.57 | 99.02 | 16.86 | 94.32 | 2.81 | 97.77 | 11.65 | 96.41 | *0.57* | *98.30* |
| | | Random | 99.86 | 99.25 | 16.42 | 95.12 | 2.23 | 97.61 | 11.32 | 96.31 | *1.28* | *98.08* |
| | 5 × 5 | Fixed | 99.68 | 99.23 | 14.25 | 95.24 | 1.95 | 97.60 | 12.40 | 95.80 | *0.33* | *98.19* |
| | | Random | 99.14 | 99.49 | 15.10 | 94.67 | *1.69* | 97.06 | 12.35 | 96.02 | 1.95 | *98.97* |
| | 7 × 7 | Fixed | 98.94 | 99.83 | 17.85 | 94.30 | 3.53 | 95.78 | 14.32 | 95.63 | *0.92* | *98.46* |
| | | Random | 99.08 | 99.15 | 18.32 | 94.51 | 3.51 | 96.34 | 13.57 | 96.52 | *0.41* | *98.51* |
| | Global noise | — | 99.12 | 99.23 | 18.79 | 94.48 | 4.14 | 96.70 | 13.07 | 97.06 | *1.14* | *98.36* |

*4.1.5. Others.* Following the standard setup, we used SGD with a learning rate of 0.01 and batch size 128. All experiments were conducted in one machine with Ubuntu 18.04 system equipped with two Tesla V100s.

*4.2. Comparison with State-of-the-Arts.* Tables 1 and 2 summarizes the detailed performance of four defense methods over attacks with varying trigger sizes and positions on four datasets (CIFAR-10, CIFAR-100, MNIST, and Fashion-MNIST).

Overall, VarDefense dominates the previous state-of-the-art defense approaches in most settings (demonstrated in Tables 1 and 2). For instance, for VarDefense, ASR drops dramatically by more than 99% with negligible degradation of ACC (less than 1%) in MNIST, while other defense methods, e.g., fine-pruning and GAN, only obtain about 90% drop, implying the superior performance of VarDefense compared to other methods. Besides, it is illustrated in Tables 1 and 2 that either of ACC and ASR to NAD is lightly better than VarDefense (≈1% in such cases). But, it is highlighted that VarDefense still achieves better performance compared to NAD. In such cases, compared to VarDefense, NAD only obtains fairly minimal improvement (≈1%) in either of ACC or ASR, whereas VarDefense attains notable improvement (≈10% on ACC and ≈2% on ASR) in another metric compared to NAD.

As introduced earlier, trigger is the most critical factor in poison attack, and hence, we investigate the effectiveness of VarDefense to varying triggers as demonstrated in Figure 3. The surprising aspect is that the lines in Figure 3

Table 2: The performance of four defense methods against different attack settings in CIFAR-10 and CIFAR-100.

| Dataset | Trigger size | Position | Before | | Fine-pruning | | NAD | | GAN-based defense | | VarDefense | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | ASR | ACC | ASR | ACC | ASR | ACC | ASR | ACC | ASR | ACC |
| CIFAR-10 | $3\times3$ | Fixed | 98.36 | 83.33 | 32.51 | 78.54 | 11.65 | 69.48 | 10.21 | 78.97 | **8.46** | **80.41** |
| | | Random | 98.80 | 83.81 | 32.07 | 77.96 | 11.24 | 69.00 | 9.52 | 78.95 | **8.70** | **80.34** |
| | $5\times5$ | Fixed | 97.28 | 83.28 | 37.25 | 78.33 | **7.98** | 69.58 | 9.29 | 78.46 | 9.10 | **80.27** |
| | | Random | 97.99 | 83.64 | 37.41 | 78.34 | **7.68** | 69.25 | 9.32 | 78.51 | 8.83 | **79.88** |
| | $7\times7$ | Fixed | 98.93 | 82.96 | 35.15 | 76.32 | 8.92 | 70.63 | 10.79 | 78.30 | **8.15** | **80.44** |
| | | Random | 98.01 | 82.89 | 35.24 | 76.29 | 8.95 | 70.49 | 10.77 | 78.32 | **8.68** | **81.02** |
| | Global noise | — | 98.08 | 84.27 | 33.08 | 78.76 | 12.33 | 70.00 | 10.42 | 78.58 | **8.01** | **79.83** |
| CIFAR-100 | $3\times3$ | Fixed | 97.77 | 54.98 | 47.06 | 47.69 | 2.56 | 33.51 | 19.88 | 46.93 | **2.14** | **46.23** |
| | | Random | 98.06 | 54.40 | 47.58 | 48.12 | 3.09 | 33.33 | 20.19 | 47.37 | **2.75** | **46.45** |
| | $5\times5$ | Fixed | 97.65 | 54.87 | 51.07 | 44.24 | 4.75 | 31.95 | 13.84 | 46.91 | **1.93** | **46.83** |
| | | Random | 98.44 | 54.79 | 51.28 | 44.64 | 4.56 | 31.89 | 13.75 | 46.98 | **1.78** | **46.45** |
| | $7\times7$ | Fixed | 97.60 | 55.65 | 46.93 | 43.51 | 5.36 | 34.52 | 18.13 | 46.05 | **2.27** | **47.83** |
| | | Random | 97.94 | 55.17 | 46.31 | 43.57 | 4.73 | 34.27 | 17.71 | 46.11 | **1.81** | **46.98** |
| | Global noise | — | 97.38 | 54.34 | 47.79 | 48.80 | **2.60** | 32.94 | 20.24 | 47.88 | 2.66 | **46.02** |

The numbers in bold mean that this method is the best compared with other methods, under the corresponding experiment settings.

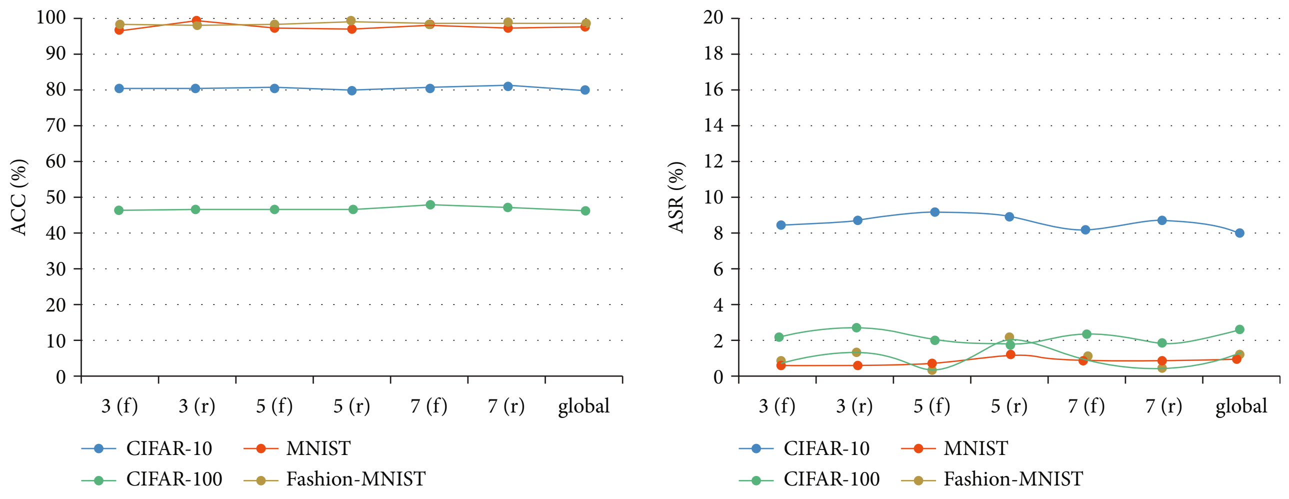


Figure 3: The performance of VarDefense over various trigger settings in four datasets. The f and r in this figure denote the fixed and random position while 3, 5, and 7 denote the trigger with $3\times3$, $5\times5$, and $7\times7$ resolution, respectively. The global means that the trigger is global imperceptible noise.

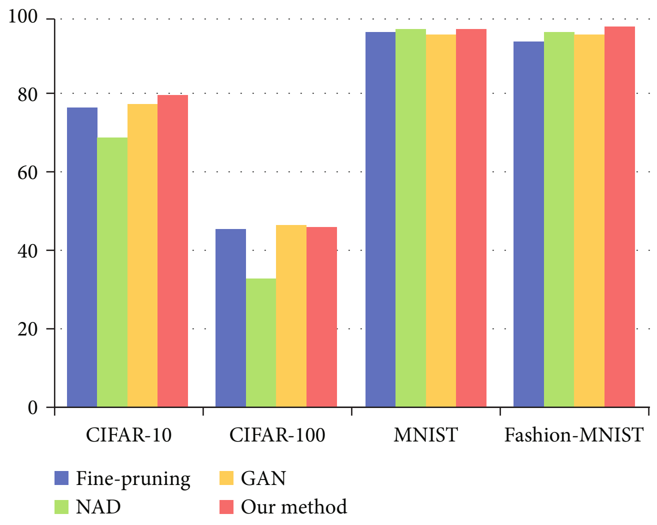
are fairly flattened, which suggests excellent and consistent resistance of VarDefense over various triggers.

In practice, the stability of performance over different tasks (i.e., datasets) is crucial and Figure 4 shows the average performance of different defense methods over various datasets. For ACC, we observed that VarDefense consistently performs well over four datasets, and the NAD performs unsteadily over four datasets. Meanwhile, for ASR, both fine-pruning and GAN have quite a few fluctuations over four datasets, and VarDefense still consistently works well. In a nutshell, the consistent and great performance of VarDefense over various datasets is more favored in practice.

### *4.3. Further Exploration to VarDefense*

*4.3.1. Impact of $\alpha$.* We are also interested in hyperparameter $\alpha$ that controls the purifying magnitude, and Table 3 demonstrates the results of coarse tuning $\alpha$ in four benchmark datasets.

It is noticed that lower $\alpha$ induces lower ASR, i.e., more effective against poison attack, but, lower $\alpha$ also results in bigger degradation of ACC. For instance, in CIFAR-10, when $\alpha = 0.0001$, despite the fact that the ASR drops to 4.5%, the ACC also degrade minimum (76.47%) and such cases also happen in MNIST, Fashion-MNIST, and CIFAR-100. In



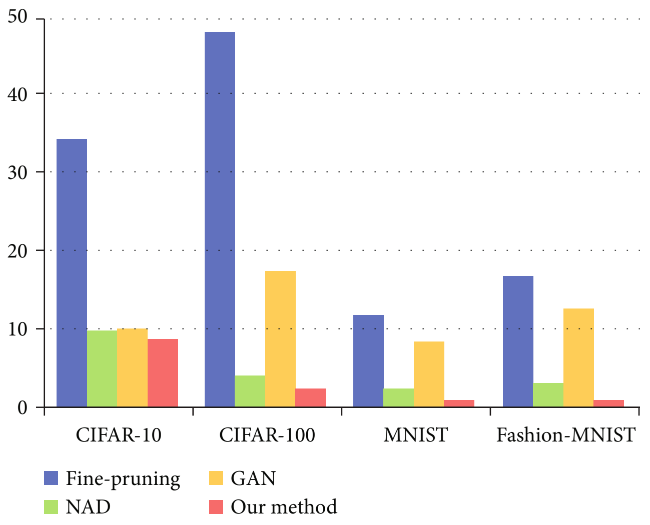
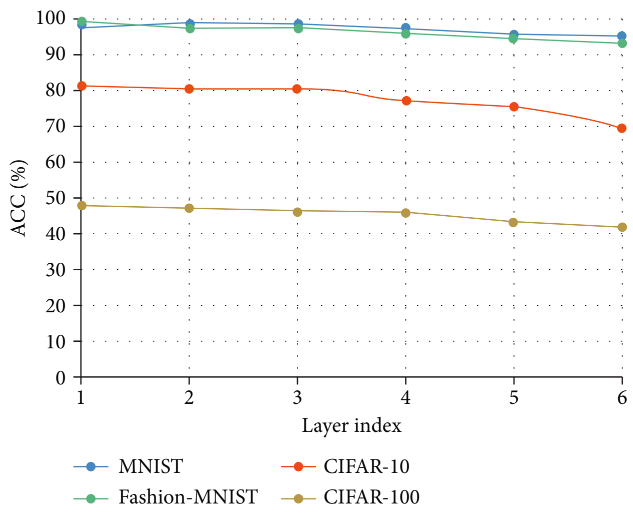


FIGURE 4: Average performance of four defense methods on CIFAR-10, CIFAR-100, MNIST, and Fashion-MNIST.

TABLE 3: The performance of VarDefense over different $\alpha$.

| $\alpha$ | MNIST |  | Fashion-MNIST |  | CIFAR-10 |  | CIFAR-100 |  |
|---|---|---|---|---|---|---|---|---|
|  | ACC | ASR | ACC | ASR | ACC | ASR | ACC | ASR |
| 0.0001 | 95.96 | **0.22** | 95.96 | **0.19** | 76.47 | **4.50** | 45.96 | **0.50** |
| 0.001 | 96.15 | 0.28 | 96.15 | 0.21 | 80.25 | 4.90 | 46.04 | 0.94 |
| 0.01 | 97.37 | 0.28 | 96.37 | 0.22 | 80.27 | 5.80 | 46.05 | 1.01 |
| 0.1 | 97.85 | 0.39 | 96.85 | 0.38 | 80.29 | 7.59 | 46.05 | 1.19 |
| 1 | 98.30 | 0.57 | 97.25 | 0.58 | 80.41 | 8.46 | 46.23 | 2.14 |
| 10 | 97.43 | 0.59 | 97.47 | 0.56 | 80.97 | 12.50 | 46.29 | 3.03 |
| 100 | 97.52 | 0.63 | 97.79 | 0.67 | 81.03 | 15.53 | 46.80 | 3.73 |
| 1000 | 97.95 | 0.69 | 97.88 | 0.63 | 81.04 | 16.85 | 46.92 | 5.07 |
| 10000 | 98.06 | 0.77 | 98.02 | 0.83 | 81.59 | 17.50 | 47.97 | 5.65 |
| 100000 | **98.19** | 1.92 | **98.07** | 1.87 | **81.72** | 19.41 | **48.27** | 5.84 |

The numbers in bold mean that this method is the best compared with other methods, under the corresponding experiment settings.



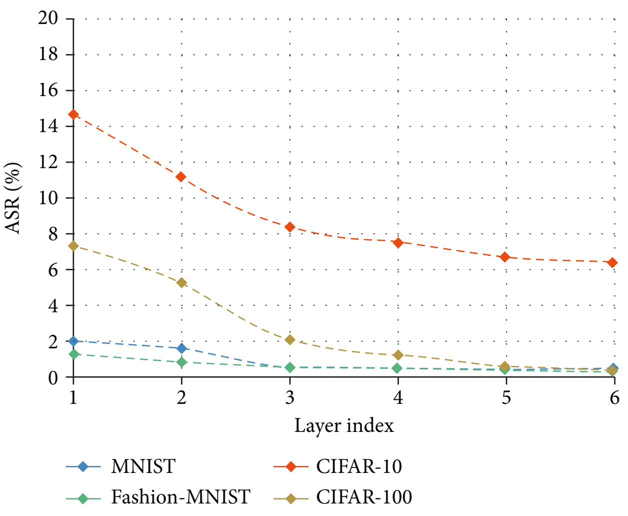


FIGURE 5: The performance of VarDefense with different purified layers.

short, there is a trade-off between maintaining model performance and removing the backdoor, and $\alpha$ can control the trade-off. In practice, to search proper $\alpha$, it is suitable that $\alpha$ iteratively increases from the small value until the ACC degrades below a predetermined threshold.

*4.3.2. Impact of Purifying Layers.* Selecting which layers to purify also is a critical factor, and Figure 5 presents the performance of VarDefense with purifying different layers. The layer index $n$ denotes purifying the last $2 \times n$ convolutional layers. We observe that, as the number of convolutional layers being purified (layer index) increases, the trend of performance change (ACC and ASR) is consistent with the inference as mentioned in Approach: purifying more layers induces lower ASR but with raising performance loss.

In conclusion, purifying layers play a similar role to $\alpha$, in which both of them adjust the balance between ACC and ASR. Therefore, the defenders should carefully tune the layer index and $\alpha$ to achieve the desired trade-off between ACC and ASR.

## 5. Conclusions

In this paper, we point out the existence of bad neurons that causes the poison attack and identify removing the bad neurons as the main challenge in backdoor defense. Moreover, we design an effective defense method VarDefense which distinguishes the bad neurons based on variance and adopts a great purifying strategy. To solve unstable training of VarDefense, the momentum trick is introduced in VarDefense, remarkably increasing the efficiency of purifying. Extensive evaluations on four benchmark datasets show that the proposed defense method can reliably remove the backdoor with a slight performance loss compared to three state-of-the-art defense methods.

## Data Availability

The dataset of MNIST is available on the site http://yann.lecun.com/exdb/mnist/. The dataset of Fashion-MNIST is available on the site https://github.com/zalandoresearch/fashion-mnist. The dataset of CIFAR-10 and CIFAR-100 is available on the site http://www.cs.toronto.edu/~kriz/cifar.html.

## Conflicts of Interest

We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 62072109, No. U1804263) and the Natural Science Foundation of Fujian Province (No. 2021J06013).

## References

[1] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: a survey," 2019, http://arxiv.org/abs/1905.05055.

[2] A. Torfi, R. A. Shirvani, Y. Keneshloo, N. Tavaf, and E. A. Fox, "Natural language processing advancements by deep learning: a survey," 2020, http://arxiv.org/abs/2003.01200.

[3] Z. Batmaz, A. Yurekli, A. Bilge, and C. Kaleli, "A review on deep learning for recommender systems: challenges and remedies," *Artificial Intelligence Review*, vol. 52, no. 1, pp. 1–37, 2019.

[4] J. Xiong, R. Bi, Y. Tian, X. Liu, and D. Wu, "Towards lightweight, privacy-preserving cooperative object classification for connected autonomous vehicles," *IEEE Internet of Things Journal*, pp. 1–15, 2021.

[5] J. Xiong, R. Bi, M. Zhao, J. Guo, and Q. Yang, "Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 24–30, 2020.

[6] J. Xiong, M. Zhao, M. Bhuiyan, L. Chen, and Y. Tian, "An AI-enabled three-party game framework for guaranteed data privacy in mobile edge crowdsensing of IoT," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 922–933, 2021.

[7] X. Hu, L. Liang, L. Deng et al., "Neural network model extraction attacks in edge devices by hearing architectural hints," 2019, http://arxiv.org/abs/1903.03916.

[8] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song, "The secret revealer: generative model-inversion attacks against deep neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 253–261, Seattle, WA, USA, 2020.

[9] J. Zhang and C. Li, "Adversarial examples: opportunities and challenges," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 7, pp. 2578–2593, 2020.

[10] Y. Jia, Y. Lu, J. Shen, Q. A. Chen, Z. Zhong, and T. Wei, "Fooling detection alone is not enough: first adversarial attack against multiple object tracking," 2019, http://arxiv.org/abs/1905.11026.

[11] K. Xu, G. Zhang, S. Liu et al., "Adversarial T-shirt! evading person detectors in a physical world," in *European Conference on Computer Vision*, pp. 665–681, Springer, Cham, 2020.

[12] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "BadNets: evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47230–47244, 2019.

[13] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017, http://arxiv.org/abs/1712.05526.

[14] C. Xie, K. Huang, P. Y. Chen, and B. Li, "DBA: distributed backdoor attacks against federated learning," in *International Conference on Learning Representations*, New Orleans, LA, USA, 2019.

[15] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International conference on machine learning. PMLR*, pp. 634–643, Long Beach, CA, USA, 2019.

[16] B. Wang, Y. Yao, S. Shan et al., "Neural cleanse: identifying and mitigating backdoor attacks in neural networks," in *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 707–723, San Francisco, CA, USA, 2019.

[17] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, “DeepInspect: a black-box Trojan detection and mitigation framework for deep neural networks,” *IJCAI*, pp. 4658–4664, 2019.

[18] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, “STRIP: a defence against Trojan attacks on deep neural networks,” in *Proceedings of the 35th annual computer security applications Conference*, pp. 113–125, San Juan, Puerto Rico, December 2019.

[19] Y. Liu, Y. Xie, and A. Srivastava, “Neural Trojans,” in *2017 IEEE international conference on computer design (ICCD)*, pp. 45–48, Boston, MA, USA, 2017.

[20] X. Qiao, Y. Yang, and H. Li, “Defending neural backdoors via generative distribution modeling,” 2019, http://arxiv.org/abs/1910.04749.

[21] K. Liu, B. Dolan-Gavitt, and S. Garg, “Fine-pruning: defending against backdooring attacks on deep neural networks,” in *International Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 273–294, Springer, Cham, 2018.

[22] K. Yoshida and T. Fujino, “Disabling backdoor and identifying poison data by using knowledge distillation in backdoor attacks on deep neural networks,” in *Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security*, pp. 117–127, Virtual Event, USA, 2020.

[23] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, “Neural attention distillation: erasing backdoor triggers from deep neural networks,” 2021, http://arxiv.org/abs/2101.05930.

[24] Y. Li, T. Zhai, B. Wu, Y. Jiang, Z. Li, and S. Xia, “Rethinking the trigger of backdoor attack,” 2020, http://arxiv.org/abs/2004.04692.

[25] B. Chen, W. Carvalho, N. Baracaldo et al., “Detecting backdoor attacks on deep neural networks by activation clustering,” 2018, http://arxiv.org/abs/1811.03728.

[26] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” 2015, http://arxiv.org/abs/1503.02531.

[27] K. Yoshida and T. Fujino, “Countermeasure against backdoor attack on neural networks utilizing knowledge distillation,” *Journal of Signal Processing*, vol. 24, no. 4, pp. 141–144, 2020.

[28] L. Muñoz-González, B. Biggio, A. Demontis et al., “Towards poisoning of deep learning algorithms with back-gradient optimization,” in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 27–38, Dallas, TX, USA, 2017.

[29] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, “Pruning convolutional neural networks for resource efficient inference,” 2016, http://arxiv.org/abs/1611.06440.