Unveiling and Mitigating Backdoor Vulnerabilities based on Unlearning Weight Changes and Backdoor Activeness

Weilin Lin[1] **Li Liu**[1*] **Shaokui Wei**[2] **Jianze Li**[3,4,2] **Hui Xiong**[1]
[1]The Hong Kong University of Science and Technology (Guangzhou)
[2]The Chinese University of Hong Kong, Shenzhen
[3]Shenzhen International Center for Industrial and Applied Mathematics
[4]Shenzhen Research Institute of Big Data

Abstract

The security threat of backdoor attacks is a central concern for deep neural networks (DNNs). Recently, without poisoned data, unlearning models with clean data and then learning a pruning mask have contributed to backdoor defense. Additionally, vanilla fine-tuning with those clean data can help recover the lost clean accuracy. However, the behavior of clean unlearning is still under-explored, and vanilla fine-tuning unintentionally induces back the backdoor effect. In this work, we first investigate model unlearning from the perspective of weight changes and gradient norms, and find two interesting observations in the backdoored model: 1) the weight changes between poison and clean unlearning are positively correlated, making it possible for us to identify the backdoored-related neurons without using poisoned data; 2) the neurons of the backdoored model are more active (*i.e.*, larger gradient norm) than those in the clean model, suggesting the need to suppress the gradient norm during fine-tuning. Then, we propose an effective two-stage defense method. In the first stage, an efficient *Neuron Weight Change (NWC)-based Backdoor Reinitialization* is proposed based on observation 1). In the second stage, based on observation 2), we design an *Activeness-Aware Fine-Tuning* to replace the vanilla fine-tuning. Extensive experiments, involving eight backdoor attacks on three benchmark datasets, demonstrate the superior performance of our proposed method compared to recent state-of-the-art backdoor defense approaches. The code is available at https://github.com/linweiii/TSBD.git.

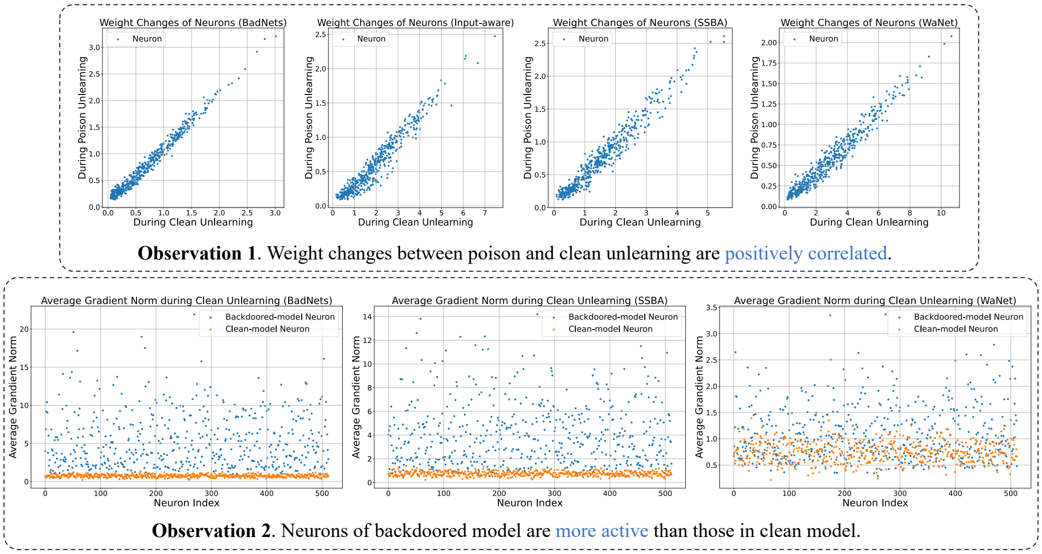
1 Introduction

Over the past few years, *deep neural networks* (DNNs) have achieved surprising success in several real-world applications, such as *face recognition* [1–3], *medical image processing* [4, 5], and *autonomous driving* [6, 7], *etc*. However, DNNs are susceptible to malicious attacks that can compromise their security and reliability. One typical example is the *backdoor attack* [8–11], where the adversary maliciously manipulates the training dataset or training process to produce a backdoored model, which performs normally on clean data while predicting any sample with a particular trigger pattern to a pre-defined target label. In this work, we focus on the *post-training defense* scenario where, given a backdoored model and a small set of clean training samples, one aims to mitigate the backdoor effect while maintaining the performance on clean data, thereby obtaining a benign model.

Up to now, several important methods have been developed for *backdoor defense* [14–18]. One promising approach is *poison unlearning*, which involves updating a backdoored model by unlearning

[*] Corresponds to Li Liu (avrillliu@hkust-gz.edu.cn)

38th Conference on Neural Information Processing Systems (NeurIPS 2024).



Observation 1. Weight changes between poison and clean unlearning are positively correlated.

Observation 2. Neurons of backdoored model are more active than those in clean model.

Figure 1: Illustration of two observations. Figures for Observation 1 show distributions of neuron weight changes during clean unlearning and poison unlearning. Figures for Observation 2 compare the average gradient norm for each neuron on the backdoored model and clean model, which are calculated with one-epoch clean unlearning. Being "more active" means a larger gradient norm. Experiments are conducted on PreAct-ResNet18 [12] using CIFAR-10 [13] for the clean model, along with additional attacks using 10% poisoning ratio for the backdoored model. The last convolutional layers are chosen for illustration.

from poisoned data. This technique has been utilized in various backdoor defenses such as ABL [14], D-BR [19], *Neural Cleanse* (NC) [20], and i-BAU [21], *etc*. To avoid approximating poisoned data, another approach called *clean unlearning* was conducted by RNP [22]. This technique only uses clean data for unlearning and then prunes the backdoored model, which has been proven to be effective. Through relevant experiments, we find an interesting connection between poison unlearning and clean unlearning, as illustrated in **Observation 1** of Figure 1. Specifically, by calculating the weight changes of each neuron during the two unlearning processes on the backdoored models[2], we find that they exhibit a strong positive correlation, *i.e.*, the neurons exhibiting significant weight changes during clean unlearning also tend to play crucial roles in poison unlearning, indicating a stronger association with backdoor-related activities. Moreover, we further investigate the **backdoor activeness** during learning processes[3], *i.e.*, comparing the average gradient norm for each neuron in both the backdoored and clean models. The results are shown in **Observation 2** of Figure 1, revealing that neurons in the backdoored model are always more active compared to those in the clean model.

Inspired by the above two observations regarding the backdoored model , we propose **T**wo-**S**tage **B**ackdoor **D**efense (**TSBD**), consisting of stage 1) *Neuron Weight Change-based Backdoor Reinitialization* and stage 2) *Activeness-Aware Fine-tuning*. In the first stage, we first conduct clean unlearning on the backdoored model, followed by the neuron weight change calculation, where both the changes of each subweight[4] and neuron are recorded. Then, we conduct zero reinitialization to mitigate the backdoor effect by reinitializing the most-changed subweights among the top-$n\%$ most-changed neurons as 0 in the original backdoored model. In the second stage, we adopt activeness-aware fine-tuning with gradient-norm regulation to recover clean accuracy and suppress the reactivation of the backdoor effect. Extensive experiments demonstrate the superior defense performance of the proposed method compared to state-of-the-art (SOTA) backdoor defense methods.

To summarize, our main contributions are three-fold. **(1) Novel Insight:** We are the first to uncover the strong positive correlation between neuron weight changes in clean unlearning and poison unlearning. We also reveal the high backdoor activeness in the backdoored model during the learning process. **(2) Effective Defense Method:** We further develop an effective two-stage defense method based

[2] Four attacked models on BadNets [8], Input-aware [23], SSBA [9], and WaNet [24], are used for illustration.

[3] Unlearning, as the opposite process of model learning, can also be considered as a kind of learning process.

[4] A subweight represents one learnable weight in a neuron weight matrix.

on unlearning weight changes and backdoor activeness, considering both backdoor mitigation and clean-accuracy recovery, respectively. **(3) SOTA Performance:** Experimental results and analysis show that our proposed method achieves SOTA performance in backdoor defense.

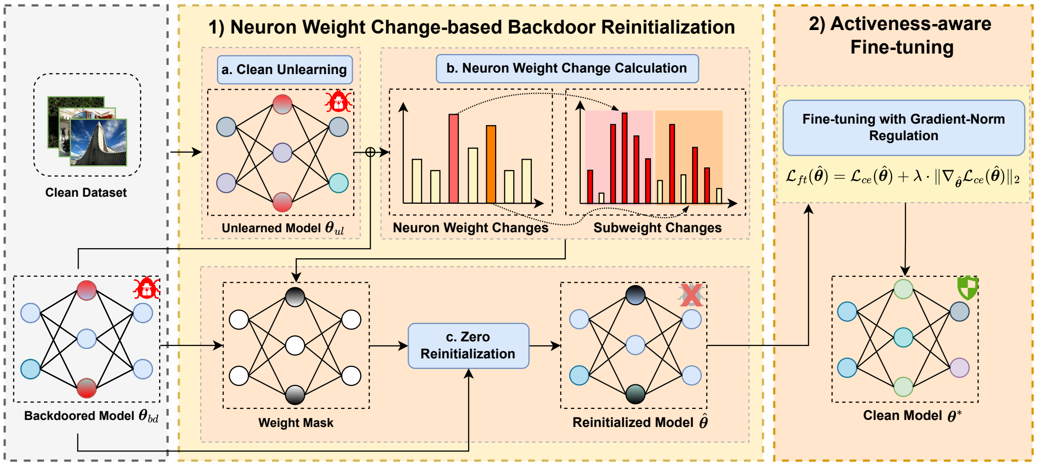

Clean Dataset
1) Neuron Weight Change-based Backdoor Reinitialization
a. Clean Unlearning
Unlearned Model θ_{ul}
b. Neuron Weight Change Calculation
Neuron Weight Changes
Subweight Changes
Backdoored Model θ_{bd}
Weight Mask
c. Zero Reinitialization
Reinitialized Model $\hat{\theta}$
2) Activeness-aware Fine-tuning
Fine-tuning with Gradient-Norm Regulation
$\mathcal{L}_{ft}(\hat{\boldsymbol{\theta}}) = \mathcal{L}_{ce}(\hat{\boldsymbol{\theta}}) + \lambda \cdot \|\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}})\|_2$
Clean Model θ^*

Figure 2: Overview of the proposed Two-Stage Backdoor Defense framework.

2 Related Work

2.1 Backdoor Attack

In the literature, various backdoor attacks on DNNs have been proposed, which can be categorized into *data poisoning attacks* and *training-controllable attacks*. BadNets [8] is one of the earliest data poisoning attacks in this field. In this attack, a small proportion of the original data is selected and patched with a pre-defined pattern, known as a *trigger*. The labels of these patched data points are then modified to a target label. The mixed dataset, containing both clean and poisoned data, is used to train the DNNs, resulting in the implantation of the backdoor. Under a similar procedure, Blended [25] was proposed as a stronger attack by blending an entire pre-defined image into the original clean data with controllable transparency. Recently, more advanced and stealthy attacks have been proposed to enhance the trigger, such as SIG [26], label consistent attacks [27, 28], SSBA [9], *etc*. Another category is training-controllable attacks [23, 24, 29–31], where the attackers design triggers with permission to control the training process. Two significant examples are WaNet [24] and Input-aware [23], which generate unique triggers for different input data by incorporating an injection function into the model training process. This approach makes these attacks more difficult to detect compared to previous attacks with fixed triggers.

2.2 Backdoor Defense

According to the different stages of model training, backdoor defense methods can be classified into two types: *training-stage defenses* and *post-training defenses*.

Training-stage Defenses. In training-stage defenses, defenders have access to a mixed training dataset containing both clean data and poisoned data with triggers. ABL [14] discovers that the loss-dropping speed of poisoned data during the early stages of model training is faster, and thus isolates them for poison unlearning. DBD [32] splits the training process into three steps to separate the training of feature extraction from that of the subsequent classifier to evade the learning of trigger-label correlation. Similarly, D-ST/D-BR [19] observes that the transformations of poisoned-data feature representations are more sensitive than clean ones, and thus proposes to modularize the training process.

Post-training Defenses. In post-training defenses [33–36], defenders aim to erase the backdoor effect in the learned DNNs using a small portion of clean data. FP [37] is one of the earliest defense methods, which observes that poisoned data and clean data activate different neurons in a backdoored DNN, and thus keeps pruning the less-activated neurons in response to clean data until a significant drop in accuracy occurs. After that, vanilla fine-tuning is employed to recover the lost clean accuracy. Using the pruning strategy [38–40], ANP [41] observes that the backdoor-related neurons exhibit higher sensitivity to adversarial perturbations compared to others, and thus trains a pruning mask using minimax optimization. Continuing along this line, AWM [42] and RNP [22] use a similar mask training process with main modifications in neuron perturbations to data perturbations and *clean unlearning*, respectively. Different strategies are also proposed for defense. For example, NC [20] proposes to recover the trigger before the subsequent backdoor removal. NAD [43], for the first time, adopts model distillation to guide the learning of a benign student model. Additionally, employing unlearning techniques, SAU [44] treats backdoor triggers as a form of adversarial perturbation, and generates poisoned data through optimization on clean data, which are then used in *poison unlearning*.

Unlearning for Backdoor Defense. *Model unlearning* can be considered as an opposite process against learning, aiming to remove the impact of a training subset from a trained model [45]. In the field of backdoor defense, unlearning the possible poisoned data (*i.e.*, *poison unlearning*) is an effective way to remove the learned backdoor. NC [20] and BTI-DBF [46] try to generate the possible poisoned data with either trigger inversion or poison-data generator; ABL [14] and D-BR [19] focus on filtering out the poisoned data from the training dataset according to their attributes during training; i-BAU [21] and SAU [47] assume the adversarial perturbation as a type of trigger and generate poisoned data with adversarial example. To avoid inducing bias, recent work tries to directly unlearn the available clean data (*i.e.*, *clean unlearning*) for defense. RNP [22] finds that a clean-unlearned model can help expose the backdoor neurons for the subsequent pruning-mask learning.

However, there exist some limitations among those techniques, *e.g.*, clean unlearning is still under-explored, and the vanilla fine-tuning unintentionally increases the attack success rate. In this paper, we propose a comprehensive two-stage defense method breaking through the two limitations.

3 Methods

3.1 Problem Formulation

Threat Model. We assume that the attacker has full access to the training data. Their goal is to poison a portion of the dataset by injecting triggers into the data so that the trained model misclassifies the poisoned data to the target class while still performing normally on clean data. The *poisoning ratio* (*e.g.*, 10%) is used to depict the proportion of poisoned data within the entire dataset. We denote the parameters of the backdoored model as $\boldsymbol{\theta}_{bd} = \{\boldsymbol{\theta}_{bd}^{(l)}\}_{1\leq l\leq L}$ satisfying $\boldsymbol{\theta}_{bd}^{(l)} \in \mathbb{R}^{K^{(l)}\times I^{(l)}}$, where $\boldsymbol{K} = \{K^{(l)}\}_{1\leq l\leq L}$ and $\boldsymbol{I} = \{I^{(l)}\}_{1\leq l\leq L}$ represent the neuron numbers and learnable subweight numbers, respectively. Specifically, for the $l^{th} \in \{1, \ldots, L\}$ layer, there are $K^{(l)}$ neurons in total and $I^{(l)}$ subweights for each neuron.

Defense Setting. The defender's goal is to remove the backdoor effect, which causes poisoned data to be misclassified to the target class, from the backdoored model while minimizing the impact on the prediction accuracy for clean data. Following the previous defense setting [37, 41], we assume that the defender knows nothing about the poisoned data and possesses only 5% of the total dataset as clean data, denoted as $\mathcal{D}_c$.

3.2 Neuron Weight Change & Suggestions Given by the Two Observations

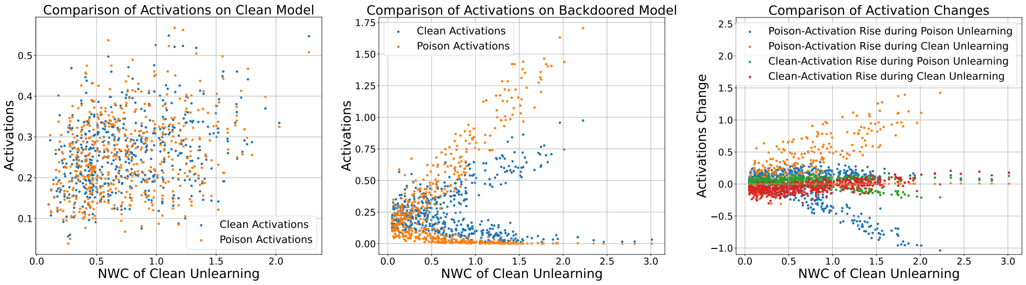
In this subsection, we provide more details on the unlearning formulation and offer suggestions based on the two observed observations.

Model Unlearning. Model Unlearning can be defined as the reverse process of model training [22], which involves maximizing the loss value on a given dataset. Given a DNN model f parameterized as $\boldsymbol{\theta}$ and a dataset $\mathcal{D}$ for unlearning, the maximization problem can be formulated as:

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},y)\in\mathcal{D}} \left[\mathcal{L}(f(\boldsymbol{x};\boldsymbol{\theta}), y)\right], \tag{1}$$

where $(\boldsymbol{x}, y) \in \mathcal{D}$ represents the images and their corresponding labels, and $\mathcal{L}$ denotes the loss function used in this task, *e.g.*, cross-entropy loss.

Intuitively, by maximizing the loss expectation, the unlearned model, parameterized as $\boldsymbol{\theta}_{ul}$, is prone to fail at the task specified in $\mathcal{D}$. In this paper, we term the process as *clean unlearning* when all



(a) Original clean model (b) Original backdoored model (c) Backdoored-model unlearning

Figure 3: Illustration of clean and poison activations of each neuron. (a) and (b) represent the activations on the original clean and backdoored model, respectively. (c) shows the activation changes during the clean and poison unlearning on backdoored model. Activations are captured from the last convolutional layer with an additional *Relu* activation function on PreAct-ResNet18 [12].

the data in $\mathcal{D}$ are clean, denoted as $\mathcal{D}_c$. On the other hand, *poison unlearning* refers to the scenario where all the data in $\mathcal{D}$ are poisoned with a trigger. By default, both clean and poison unlearning are terminated when the model performs poorly on the corresponding tasks, such as achieving only 10% clean accuracy or attack success rate.

Neuron Weight Change. To comprehensively quantify the weight changes of a neuron during the entire unlearning process, we define the *Neuron Weight Change* (NWC), where the L_1 norm is calculated on every neuron's weight differences. The NWC for the $k^{th} \in \{1, \dots, K^{(l)}\}$ neuron in layer $l \in \{1, \dots, L\}$ can be formulated as:

$$\mathrm{NWC}^{(l)k} = \sum_{i=0}^{I^{(l)}} \|\boldsymbol{\theta}_{ul}^{(l)ki} - \boldsymbol{\theta}_{bd}^{(l)ki}\|_1, \tag{2}$$

where $\sum_{i=0}^{I^{(l)}} \|\cdot\|_1$ is to calculate the L_1 norm for the differences on a neuron with totally $I^{(l)}$ subweights, $\boldsymbol{\theta}_{ul}^{(l)ki}$ and $\boldsymbol{\theta}_{bd}^{(l)ki}$ denote the $i^{th} \in \{1, \dots, I^{(l)}\}$ subweights of k^{th} neuron after and before the entire unlearning process, respectively. A larger NWC^k indicates more significant changes occurring in neuron k during the unlearning process. Similarly, in Equation (2), the term $\|\boldsymbol{\theta}_{ul}^{(l)ki} - \boldsymbol{\theta}_{bd}^{(l)ki}\|_1$ represents the changes in the i^{th} subweight of neuron k in layer l, *i.e.*, defined as *Subweight Change*.

Suggestions Given by the Two Observations. As demonstrated in Section 1, we have two interesting observations regarding the backdoored model. **Observation 1** shows that the neurons exhibiting significant weight changes during clean unlearning also tend to play crucial roles in poison unlearning. It suggests that we can employ clean unlearning to identify and eliminate backdoor-related neurons using NWC, at the expense of reducing clean accuracy. On the other hand, **Observation 2** reveals that neurons in the backdoored model are always more active compared to those in the clean model. It suggests that we should suppress the gradient norm during the learning process if we want to recover it to a clean model. These two suggestions act as the main supports to our proposed TSBD.

3.3 Further Investigations & Insights

Here, we offer insights from the perspective of neuron activations, trying to answer two important questions: **[Q1]** What causes the clean unlearning NWCs to exhibit a positive correlation with those in poison unlearning, and **[Q2]** What motivates the neurons more active in the backdoored model.

Neuron Activations & Activation Rise. The neuron activation is determined by computing the average value of all inputs to the specific neuron, *e.g.*, $h^{(l)k} \approx \sigma(\boldsymbol{\theta}^{(l)k}\boldsymbol{h}^{(l-1)})$ for simplicity, where $\sigma(\cdot)$ is the activation function. In line with the terminology used in FP [37], *clean activation* denotes the scenario where all the input samples are clean while *poison activation* refers to the presence of poisoned inputs. To better observe the changes in activation during unlearning, we calculate the activation rise from the original model to the unlearned model, *i.e.*, $\Delta h^{(l)k} = h_{ul}^{(l)k} - h_{bd}^{(l)k}$. A positive value indicates an increase in activation, while a negative value signifies a decrease.

Relationship between NWC and Activation Change. Considering that a backdoored model has learned two tasks from the clean and poisoned data [14], the main influence of NWC on a neuron can be roughly attributed to its activation change on both clean and poisoned inputs. For neuron k in layer l, we can formulate it as $\mathrm{NWC}^{(l)k} \propto |\Delta h_c^{(l)k}| + |\Delta h_p^{(l)k}|$, where $\Delta h_c^{(l)k}$ and $\Delta h_p^{(l)k}$ represent the activation rise on clean and poisoned inputs, respectively.

Figure 3 illustrates the clean and poison activations in (a) the original clean model, (b) the original backdoored model, and (c) the backdoored-model unlearning. We now try to answer the above two questions from these observations. **[A1]** We can observe that poison activations are the main factors affected during both clean unlearning (increase) and poison unlearning (decrease), while clean activations are only slightly influenced (see Figure 3 (c)), *i.e.*, $\mathrm{NWC}_{\uparrow}^{(l)k} \to |\Delta h_c^{(l)k}|_{\approx} + |\Delta h_p^{(l)k}|_{\uparrow}$. Also, the growing NWC during clean unlearning can indicate larger poison and clean activations (where $h_p^{(l)k} > h_c^{(l)k}$) to some extent (see Figure 3 (b)). Thus, we deduce that the co-function of clean and poison activations dominates the performance on both tasks, while the higher values of poison activation in the backdoored model make it an easier target for modification. In this case, the neurons with higher poison activations tend to decrease their values during poison unlearning, thereby reducing the attack success rate. Conversely, during clean unlearning, these neurons increase poison activations, which suppresses the function of clean activations and reduces clean accuracy. **[A2]** Similarly, the significantly lower values of mixed clean and poison activations (maximum: 0.5676) on the clean model (see Figure 3 (a)) indicate that it is less active compared to the backdoored model (maximum: 1.7053), where a similar pattern can also be seen on the bottom left of Figure 3 (b).

3.4 Two-Stage Backdoor Defense Framework

Based on the above observations, we now propose a defense framework incorporating *Neuron Weight Change-based Backdoor Reinitialization* (including *Clean Unlearning*, *Neuron Weight Change Calculation* and *Zero Reinitialization*), and *Activeness-aware Fine-tuning*. The detailed defense process is illustrated in Figure 2 and Algorithm 1 (found in Appendix A).

Stage 1) Neuron Weight Change-based Backdoor Reinitialization. We aim to mitigate the backdoor effect with acceptable clean-accuracy sacrificed in this stage. *[a. Clean Unlearning.]* To identify the backdoor-related neurons, we first conduct a full clean unlearning using the available clean data $\mathcal{D}_c$ on the backdoored model. *[b. Neuron Weight Change Calculation.]* Then, we record the subweight changes and calculate the NWC for each neuron as described in Section 3.2. The resulting sorted order of neurons reflects the backdoor strength. *[c. Zero Reinitialization.]* After that, we can now eliminate the backdoor effect through zero reinitialization. Based on the NWC neuron order, we identify the top-$n\%$ neurons as strongly backdoor-related. As suggested in Section 3.3, high-NWC neurons may also contribute to clean accuracy to some extent. Therefore, we further choose to reinitialize the subweights of the most-changing $m\%$ among the selected neurons to zero in the backdoored model, while leaving the others unchanged. The reinitialized model parameter is denoted as $\hat{\boldsymbol{\theta}}$.

Stage 2) Activeness-Aware Fine-tuning. To further repair the reinitialized subweights and avoid recovering the backdoor effect again, we conduct activeness-aware fine-tuning on the reinitialized model ($\hat{\boldsymbol{\theta}}$) using the clean dataset, $\mathcal{D}_c$. This involves incorporating gradient-norm regulation into the original loss function, such as the cross-entropy loss $\mathcal{L}_{ce}$, to penalize high gradient values. This regulation serves to suppress neuron activity during fine-tuning. The final loss function is:

$$\mathcal{L}_{ft}(\hat{\boldsymbol{\theta}}) = \mathcal{L}_{ce}(\hat{\boldsymbol{\theta}}) + \lambda \cdot \|\nabla_{\hat{\boldsymbol{\theta}}} \mathcal{L}_{ce}(\hat{\boldsymbol{\theta}})\|_2, \tag{3}$$

where $\|\nabla_{\hat{\boldsymbol{\theta}}} \mathcal{L}_{ce}(\hat{\boldsymbol{\theta}})\|_2$ represents the L_2 norm of gradients, and λ is the *penalty coefficient* controlling its impact. Hence, the objective of fine-tuning is to minimize the loss function $\mathcal{L}_{ft}(\hat{\boldsymbol{\theta}})$ using the available clean data $\mathcal{D}_c$:

$$\min_{\hat{\boldsymbol{\theta}}} \mathbb{E}_{(\boldsymbol{x}_c, y_c) \in \mathcal{D}_c} [\mathcal{L}_{ft}(f(\boldsymbol{x}_c; \hat{\boldsymbol{\theta}}), y_c)]. \tag{4}$$

During practical optimization for computational efficiency, we adopt the approximation scheme in [48], which can be formulated as:

$$\nabla_{\hat{\boldsymbol{\theta}}} \mathcal{L}_{ft}(\hat{\boldsymbol{\theta}}) \approx (1-\alpha) \nabla_{\hat{\boldsymbol{\theta}}} \mathcal{L}_{ce}(\hat{\boldsymbol{\theta}}) + \alpha \nabla_{\hat{\boldsymbol{\theta}}} \mathcal{L}_{ce}(\hat{\boldsymbol{\theta}} + r \frac{\nabla_{\hat{\boldsymbol{\theta}}} \mathcal{L}_{ce}(\hat{\boldsymbol{\theta}})}{\|\nabla_{\hat{\boldsymbol{\theta}}} \mathcal{L}_{ce}(\hat{\boldsymbol{\theta}})\|_2}). \tag{5}$$

Table 1: Comparison with the SOTA defenses on **CIFAR-10** dataset with PreAct-ResNet18 (%).

Backdoor	No Defense			FT			FP [37]			NAD [43]			NC [20]		
Attacks	ACC $\uparrow$	ASR $\downarrow$	DER $\uparrow$	ACC $\uparrow$	ASR $\downarrow$	DER $\uparrow$	ACC $\uparrow$	ASR $\downarrow$	DER $\uparrow$	ACC $\uparrow$	ASR $\downarrow$	DER $\uparrow$	ACC $\uparrow$	ASR $\downarrow$	DER $\uparrow$
BadNets [8]	91.32	95.03	-	89.96	1.48	96.10	**91.31**	57.13	68.95	89.87	2.14	95.72	89.05	1.27	95.75
Blended [25]	93.47	99.92	-	92.78	96.11	51.56	93.17	99.26	50.18	92.17	97.69	50.47	**93.47**	99.92	50.00
Input-aware [23]	90.67	98.26	-	93.12	1.72	98.27	91.74	**0.04**	**99.11**	**93.18**	1.68	98.29	92.61	0.76	98.75
LF [49]	93.19	99.28	-	92.37	78.44	60.01	**92.90**	98.97	50.01	92.37	47.83	75.31	91.62	**1.41**	**98.15**
SIG [26]	84.48	98.27	-	**90.80**	2.37	97.95	89.10	26.20	86.03	90.02	10.66	93.81	84.48	98.27	50.00
SSBA [9]	92.88	97.86	-	92.14	74.79	61.16	92.54	83.50	57.01	91.91	77.40	59.74	90.99	**0.58**	**97.69**
Trojan [50]	93.42	100.00	-	92.42	5.99	96.51	92.46	71.17	63.94	91.88	3.73	**97.36**	91.76	8.22	95.06
WaNet [24]	91.25	89.73	-	**93.48**	17.10	86.32	91.46	1.09	94.32	93.17	22.98	83.38	91.80	7.53	91.10
Average	91.34	97.29	-	**92.13**	34.75	80.98	91.84	54.67	71.19	91.82	33.01	81.76	90.72	27.24	84.56

Backdoor	ANP [41]			CLP [38]			i-BAU [21]			RNP [22]			TSBD **(Ours)**		
Attacks	ACC $\uparrow$	ASR $\downarrow$	DER $\uparrow$	ACC $\uparrow$	ASR $\downarrow$	DER $\uparrow$	ACC $\uparrow$	ASR $\downarrow$	DER $\uparrow$	ACC $\uparrow$	ASR $\downarrow$	DER $\uparrow$	ACC $\uparrow$	ASR $\downarrow$	DER $\uparrow$
BadNets [8]	90.94	5.91	94.37	90.06	77.50	58.14	89.15	**1.21**	95.83	89.81	24.97	84.28	90.72	1.31	**96.53**
Blended [25]	93.00	84.90	57.28	91.32	99.74	49.01	87.00	50.53	71.46	88.76	79.74	57.73	91.61	**2.61**	**97.73**
Input-aware [23]	91.04	1.32	98.47	90.30	2.17	97.86	89.17	27.08	84.84	90.52	1.84	98.13	93.06	1.94	98.16
LF [49]	92.83	54.99	71.96	92.84	99.18	49.88	84.36	44.96	72.75	88.43	7.02	93.75	91.20	2.64	97.32
SIG [26]	83.36	36.43	80.36	83.80	98.91	49.66	85.67	3.68	97.29	84.48	98.27	50.00	90.41	**1.27**	**98.50**
SSBA [9]	**92.67**	60.16	68.74	91.38	68.13	64.11	87.67	3.97	94.34	88.60	17.89	87.84	91.57	1.66	97.44
Trojan [50]	92.97	46.27	76.64	**92.98**	100.00	49.78	90.37	**2.91**	97.02	90.89	3.59	96.94	91.76	5.06	96.64
WaNet [24]	91.32	2.22	93.76	81.91	78.42	50.99	89.49	5.21	91.38	90.43	0.96	93.98	93.26	**0.88**	**94.43**
Average	91.02	36.53	80.20	89.32	78.01	58.68	87.86	17.44	88.11	88.99	29.28	82.83	91.70	**2.18**	**97.09**

Here, r is used for appropriating the Hessian multiplication operation, and $\alpha = \frac{\lambda}{r}$ is the *balance coefficient*. Due to the space limit, the detailed derivation and algorithm are provided in Appendix B. After the activeness-aware fine-tuning stage, we can obtain a repaired clean model, which demonstrates outstanding performance in the experiments.

4 Experiments

4.1 Experimental Setup

Attack Setup. We consider 8 SOTA backdoor attacks in the main experiment. There are BadNets [8], Blended [25], Input-aware [23], LF [49], SIG [26], SSBA [9], Trojan [50] and WaNet [24]. For a fair comparison, we follow the default attack configuration as in BackdoorBench [51], including the trigger pattern, trigger size, the target label (*i.e.*, the 0^{th} label), *etc*. We choose 10% poisoning ratio as the default setting. To fully evaluate the effectiveness of our proposed framework, all the attacks are implemented on three benchmark datasets, *i.e.*, CIFAR-10 [13], Tiny ImageNet [52], and GTSRB [53], over two popular DNNs, *i.e.*, PreAct-ResNet18 [12] and VGG19-BN [54]. In particular, SIG is only applied to CIFAR-10 since it cannot reach a 10% poisoning ratio on Tiny ImageNet and GTSRB. Besides, we also evaluate the defense methods under 5% and 1% poisoning ratios to verify the robustness of our proposed framework. Due to the space limit, we only exhibit parts of the main results in this section. More implementation details can be found in Appendix C.

Defense Setup. We compare the proposed framework with 8 SOTA backdoor defense methods: Fine-tuning (FT), Fine-pruning (FP) [37], NAD [43], NC [20], ANP [41], CLP [38], i-BAU [21], and RNP [22]. We use the recommended configurations in BackdoorBench [51]. Since the defense settings for training-stage defenses are different [14, 32], we only compare the post-training defenses, where 5% clean data can be accessed following the previous settings [41]. For TSBD, we set the learning rates for clean unlearning to 10^{-4} and fine-tuning to 10^{-2}. The default neuron ratio n and weight ratio m are set to 0.15 and 0.7, respectively. We follow the suggested settings of hyper-parameters $r = 0.05$ and $\alpha = 0.7$ for fine-tuning [48].

Evaluation Metrics. We use three metrics to evaluate the performance of each defense method: Accuracy on clean data (**ACC**), Attack Success Rate (**ASR**), and Defense Effectiveness Rating (**DER**) [55]. Specifically, ACC measures the proportion of clean data correctly predicted; ASR measures the proportion of poisoned data misclassified to the target label; DER $\in [0, 1]$ evaluates the cost of ACC for reducing ASR, which is defined as: $\mathrm{DER} = [\max(0, \Delta\mathrm{ASR}) - \max(0, \Delta\mathrm{ACC}) + 1]/2$, where $\Delta\mathrm{ASR}$ and $\Delta\mathrm{ACC}$ are the drop in ASR and ACC after applying defense on the backdoored model, respectively. Larger ACC, DER, and smaller ASR are desired for a successful defense. Note that in the following result tables, "-" indicates that the value is inapplicable. The **boldface** values indicate the best performance and the underline values denote the second-best result.

4.2 Main Results

We validate the effectiveness of our proposed framework on 8 SOTA backdoor attacks and compare it with 8 defenses. In this section, we present the main results on CIFAR-10 and Tiny ImageNet with

Table 2: Comparison with the SOTA defenses on **Tiny ImageNet** dataset with PreAct-ResNet18 (%).

Backdoor	No Defense			FT			FP [37]			NAD [43]			NC [20]		
Attacks	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑
BadNets [8]	56.23	100.00	-	55.18	0.09	**99.43**	51.73	99.99	47.76	46.37	0.27	94.93	48.26	0.10	95.96
Blended [25]	56.03	99.71	-	55.04	97.73	50.49	51.89	95.94	49.81	46.89	95.00	47.79	52.55	93.21	51.51
Input-aware [23]	57.45	98.85	-	57.45	1.65	98.60	55.28	62.92	66.88	47.91	1.86	93.73	56.20	0.09	**98.76**
LF [49]	55.97	98.57	-	54.80	94.87	51.26	51.44	95.25	49.40	45.45	50.49	68.78	52.99	85.56	55.02
SSBA [9]	55.22	97.71	-	54.80	91.57	52.86	50.47	88.87	52.04	45.32	57.32	65.25	52.47	53.47	70.75
Trojan [50]	55.89	99.98	-	55.42	0.50	**99.50**	50.22	8.82	92.74	48.48	0.83	95.87	52.69	0.15	98.31
WaNet [24]	56.78	99.49	-	**56.74**	0.19	**99.63**	53.84	3.94	96.30	46.98	0.43	94.63	52.33	0.23	97.40
Average	56.22	99.19	-	55.63	40.94	78.83	52.12	65.11	64.99	46.77	29.46	80.14	52.50	33.26	81.10

Backdoor	ANP [41]			CLP [38]			i-BAU [21]			RNP [22]			TSBD **(Ours)**		
Attacks	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑
BadNets [8]	50.55	7.74	93.29	**55.94**	100.00	49.86	51.48	97.36	48.95	21.91	**0.00**	82.84	53.72	0.34	98.58
Blended [25]	54.99	84.61	57.03	**55.70**	99.68	49.85	53.03	91.90	52.40	34.60	**0.11**	89.08	53.62	2.30	**97.50**
Input-aware [23]	53.17	0.17	97.20	**57.75**	99.58	50.00	52.48	72.98	60.45	15.57	**0.00**	78.49	55.38	0.10	98.34
LF [49]	54.66	95.39	50.94	**55.61**	98.49	49.86	51.13	85.32	54.21	49.18	**0.00**	95.89	52.47	1.90	**96.59**
SSBA [9]	52.83	91.44	51.94	**55.17**	97.65	50.01	49.86	81.90	55.22	37.64	**0.00**	90.06	52.93	1.38	**97.02**
Trojan [50]	50.37	1.40	96.53	**55.86**	8.39	95.78	52.65	98.49	49.12	46.27	**0.00**	95.18	53.66	0.31	98.72
WaNet [24]	53.87	0.75	97.91	56.21	98.50	50.21	53.71	75.23	60.60	20.50	**0.00**	81.60	55.01	0.71	98.50
Average	52.92	40.21	77.83	**56.03**	86.04	56.51	52.05	86.17	54.42	32.24	**0.02**	87.59	53.83	1.01	**97.89**

a 10% poisoning ratio on PreAct-ResNet18 for illustration, which is shown in Table 1 and Table 2. More results on GTSRB and VGG19-BN can be found in Appendix D and E, respectively.

Results on CIFAR-10. Table 1 shows the results on CIFAR-10. Results show that our TSBD outperforms all the other SOTA defenses on the average of ASR (2.18%) and DER (97.09%), as well as a promising ACC (91.70%) higher than the original "No Defense" models (91.34%), which indicates its effectiveness in removing the backdoor effect with the least cost. Though most defenses fail in strong attacks Blended, LF, or SSBA, *e.g.*, FT, FP, NAD, ANP, CLP, i-BAU, and RNP, our proposed TSBD success with the best ASR and DER on Blended and second best ASR and DER on LF and SSBA. FT and NAD perform similarly on each attack with a promising ACC, but they also fail on WaNet with a high ASR except for the mentioned strong attacks. FP and ANP perform well in ACC among all the defenses, while the defense performances on ASR and DER are unstable, which may be due to the unsuccessful backdoor-related neuron locating. CLP fails on most of the attacks with high ASR and low DER, which may be due to the structure constraint of computing channel Lipschitz only on the convolutional-batch normalization layer combination. i-BAU performs the second best on average ASR and DER, with failures on three attacks. TSBD outperforms RNP on almost all performances, which indicates that using clean unlearning with NWC is more effective than the unlearn-recovery process in RNP.

Results on Tiny ImageNet. Table 2 presents the results on Tiny ImageNet with PreAct-ResNet18. We observe that most defenses also fail on Blended, SSBA, and LF with high ASR. Similar performances of other attacks are shown on most of the defenses compared to CIFAR-10, where FT and FP also perform well in ACC and CLP fails on most attacks. Although i-BAU can successfully defend against most of the attacks in CIFAR-10, it fails on all attacks here, indicating that its adversarial training fails with large classification categories. For RNP, though it performs well in ASR on almost all attacks, the ACC is sacrificed too much to be unacceptable, indicating an unbalanced defense performance. In comparison, our TSBD can achieve SOTA on the average of DER, and perform second best on the average of ASR, which validates its superior defense performance.

4.3 Ablation Studies

Effectiveness of NWC order for Backdoor Strength. To verify the effectiveness of employing clean-unlearning NWC order in gauging the backdoor strength, we borrow the *Trigger-activated Change* (TAC) [38] order as the ground truth and compare the neuron coverage ratio on TAC under different proportions, *i.e.*, measuring the overlap of the selected neurons on both metrics. Specifically, TAC measures the change in neuron activation before and after the input image is attached with a trigger, where the larger value indicates a stronger backdoor effect. We select the following SOTA metrics for comparison: 1) the average neuron activations in FP [37]; 2) the channel Lipschitz in CLP [38]; 2) the perturb-recovery learned mask in ANP [41]; 3) the unlearn-recovery learned mask in RNP [22]. The result is illustrated in Figure 4, where the x-axis represents the (reinitializing/pruning) neuron ratio and the y-axis represents the neuron coverage ratio on TAC. The higher values on the y-axis indicate a better matching of the current metric and the TAC metric,

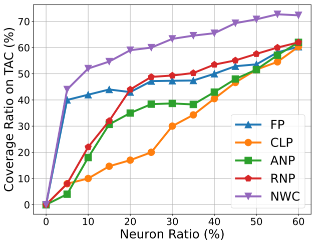


Figure 4: Comparison of neuron coverage ratio on TAC under different neuron ratios.

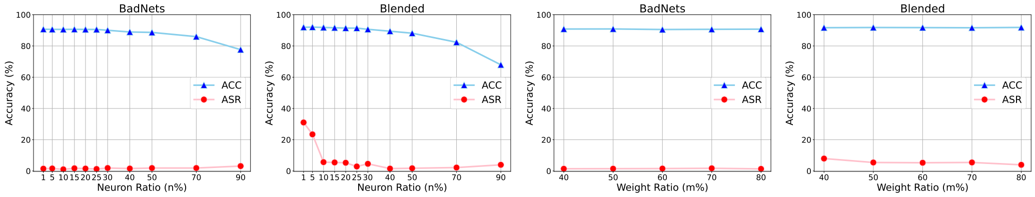


Figure 5: Performance with different neuron ratios (two subfigures on the left) and weight ratios (two subfigures on the right) under the attacks of BadNets and Blended.

i.e., more backdoor-related neurons are chosen. We can observe that NWC is always the best under different neuron ratios compared to others. It validates the effectiveness of using NWC order as the metric for neuron reinitialization/pruning. Moreover, we also test the performance of using NWC in FP to show its superiority in assisting other defenses. The results can be found in Appendix F.

Table 3: Comparison of different reinitialization schemes on CIFAR-10 with PreAct-ResNet18 (%).

Backdoor Attacks	$\mathbf{V}_1$		$\mathbf{V}_2$		$\mathbf{V}_3$ (Ours)	
	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓
BadNets [8]	69.53	0.00	68.20	0.00	**80.94**	0.00
Blended [25]	76.13	0.00	64.20	0.05	**82.72**	0.00
LF [49]	76.41	0.00	78.64	0.02	**82.42**	0.00
SSBA [9]	74.55	0.00	75.43	0.02	**83.10**	0.00

Effectiveness of Zero Reinitialization on Subweight. We compare different combinations of reinitializing neurons and weights to validate the effectiveness of zero reinitialization on the selected subweights. Specifically, three versions are designed for comparison: (1) $\mathbf{V}_1$: all weights on the selected top-NWC neurons are reinitialized to zero. (2) $\mathbf{V}_2$: $m\%$ (70% is used as default) of the top weights on each selected top-NWC neuron are reinitialized to zero. (3) $\mathbf{V}_3$ (Ours): our final version, where $m\%$ of the top weights among all selected top-NWC neurons are reinitialized to zero. As the ACC and ASR drop monotonously with more neurons selected to be reinitialized, we record the results when ASR first drops to almost zero (*i.e.*, lower equal than 0.05%). Table 3 shows the performances on four attacks. It validates that reinitializing the top weights among the selected neurons may help alleviate the hurt of clean accuracy. The failure of $\mathbf{V}_2$ may be blamed on the different backdoor strength of each neuron, *i.e.*, some strong backdoor-related neurons fail to be removed thoroughly and thus more neurons are reinitialized to reach zero ASR.

Table 4: Comparison of different fine-tuning schemes on CIFAR-10 with PreAct-ResNet18 (%).

Backdoor Attacks	No-FT		Vanilla-FT		Aa-FT (Ours)	
	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓
BadNets [8]	80.94	0.00	90.66	1.69	90.72	1.37
Blended [25]	82.72	0.00	91.40	6.39	91.61	2.61
LF [49]	82.42	0.00	91.51	5.57	91.20	2.64
SSBA [9]	83.10	0.00	91.12	7.6	91.57	1.66

Effectiveness of Gradient-norm Regulation in Fine-tuning. To validate the essential role of gradient-norm regulation in the fine-tuning stage, we compare the performance on "no fine-tuning" (No-FT), "vanilla fine-tuning" (Vanilla-FT), and our "activeness-aware fine-tuning" (Aa-FT). For a fair comparison, we follow the default settings and choose to fine-tune the reinitialized model after the second stage. The results are shown in Table 4. It shows that fine-tuning is effective in improving clean accuracy, which is sacrificed to erase the backdoored effect in reinitialization, though it is prone to bring back some extent of ASR. Furthermore, compared to vanilla fine-tuning, our implemented activeness-aware fine-tuning can suppress the rise of ASR as well as improve ACC to the same level.

4.4 Further Analysis

Table 5: Performance of hyper-parameter tuning for stage 2.

	r	α	ACC↑	ASR↓
Our Settings	0.05	0.70	90.72	1.31
Tuning r	0.01	0.70	90.68	1.30
	0.02	0.70	91.03	1.50
	0.10	0.70	90.90	1.30
	0.20	0.70	89.64	1.04
Tuning α	0.05	0.10	91.32	1.40
	0.05	0.30	91.24	1.59
	0.05	0.50	91.00	1.63
	0.05	0.90	90.68	1.26

Performance on Different Neuron Ratio and Weight Ratio. To test the sensitivity of our method towards hyper-parameter tuning, we record the performance of TSBD under a wide range of neuron ratio ($n\%$) and weight ratio ($m\%$) tuning. The experiments are conducted on CIFAR-10 with a 10% poisoning ratio on PreAct-ResNet18. The neuron ratios are tuned from 1% to 90%, and weight ratios are tuned from 40% to 80%. The results of BadNets and Blended attacks are depicted in Figure 5. It shows that TSBD is insensitive to both neuron ratio and weight ratio, where the ASR is kept at a very low level, while ACC is maintained at a top level under a wide range of tuning. The high consistency in the performances may come from the promising recovery ability of the fine-tuning stage. More results on other attacks are shown in the Appendix G.

Performance on Different r and α for Activeness-Aware Fine-tuning. To test the hyper-parameters sensitivity for stage 2, *i.e.*, r and α in Activeness-Aware Fine-tuning, we follow the tuning range in [48] under our experimental settings. The results are shown in Teble 5, which are obtained from a BadNets-attacked PreAct-ResNet18. We observe that the performance is insensitive (Changes <2% in ACC and <1% in ASR) across different hyper-parameter settings, maintaining a high level of performance.

Performance on Different Poisoning Ratio. We further investigate the performance of TSBD on different poisoning ratios, *e.g.*, 10%, 5%, and 1%. Note that a larger poisoning ratio represents a stronger attack mode. We test the performance with six attacks on these three ratios. Figure 6 shows the performances of ACC and ASR. We can observe that TSBD successfully defends all the attacks on 10% and 5% with a low ASR and a high ACC while performing less effectively on 1%. A possible reason is that the unlearning weight changes of backdoor neurons are less obvious in the weak attack mode compared to the strong attack mode with 10% or 5% poisoning ratios.

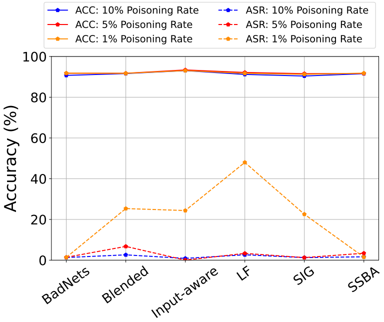


Figure 6: ACC and ASR on different poisoning ratios.

More Experiments and Analysis. Due to the space limit, we postpone the detailed discussion of the clean data ratio and the fine-tuning learning rate to Appendix H and Appendix I, respectively. We also evaluate the defense performance on the clean model in Appendix J and on the ViT in Appendix K. Further, we provide the computational overhead in terms of runtime in Appendix L.

5 Conclusion

In this work, we propose an effective two-stage backdoor defense method, TSBD, to eliminate the backdoor effect in DNNs. Our research reveals two important observations regarding the backdoored models to support our method. First, there is a positive correlation between weight changes during poison and clean unlearning in backdoored models. This finding enables us to identify and eliminate backdoor-related neurons through clean unlearning and zero reinitialization. Second, neurons in backdoored models are more active compared to those in clean models , which suggests regulating the gradient norm during fine-tuning. Furthermore, we also provide insights into these two observations from the perspective of neuron activations, which may be a valuable contribution to the field of backdoor defense. Extensive experiments demonstrate the superiority of our method over recent defenses. One current challenge as well as promising future work involves defending against backdoor attacks without any accessible clean data. The data generation techniques and data-free techniques may be the potential solutions.

6 Acknowledgements

This work was supported in part by the Guangzhou Municipal Science and Technology Project: Basic and Applied Basic research projects (No. 2024A04J4232), National Natural Science Foundation of China (No. 62101351), Guangzhou-HKUST(GZ) Joint Funding Program (Grant No.2023A03J0008), Education Bureau of Guangzhou Municipality, and Hetao Shenzhen-Hong Kong Science and Technology Innovation Cooperation Zone Project (No.HZQSWS-KCCYB-2024016).

References

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

[2] Divyarajsinh N Parmar and Brijesh B Mehta. Face recognition methods applications. *arXiv preprint arXiv:1403.0485*, 2014.

[3] Ratnawati Ibrahim and Zalhan Mohd Zin. Study of automated face recognition system for office door access control application. In *ICCSN*, 2011.

[4] Yixiong Chen, Chunhui Zhang, Li Liu, Cheng Feng, Changfeng Dong, Yongfang Luo, and Xiang Wan. Uscl: pretraining deep ultrasound image diagnosis model through video contrastive representation learning. In *MICCAI*, 2021.

[5] Yixiong Chen, Li Liu, Jingxian Li, Hua Jiang, Chris Ding, and Zongwei Zhou. Metalr: Meta-tuning of learning rates for transfer learning in medical imaging. In *MICCAI*, 2023.

[6] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 2020.

[7] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.

[8] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 2019.

[9] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. Invisible backdoor attack with sample-specific triggers. In *ICCV*, 2021.

[10] Baoyuan Wu, Li Liu, Zihao Zhu, Qingshan Liu, Zhaofeng He, and Siwei Lyu. Adversarial machine learning: A systematic survey of backdoor attack, weight attack and adversarial example. *arXiv preprint arXiv:2302.09457*, 2023.

[11] Runkai Zheng, Rongjun Tang, Jianze Li, and Li Liu. Pre-activation distributions expose backdoor neurons. In *NeurIPS*, 2022.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

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

[14] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Anti-backdoor learning: Training clean models on poisoned data. In *NeurIPS*, 2021.

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

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

[17] Xiangyu Qi, Tinghao Xie, Jiachen T Wang, Tong Wu, Saeed Mahloujifar, and Prateek Mittal. Towards a proactive {ML} approach for detecting backdoor poison samples. In *USENIX Security*, 2023.

[18] Min Liu, Alberto Sangiovanni-Vincentelli, and Xiangyu Yue. Beating backdoor attack at its own game. In *ICCV*, 2023.

[19] Weixin Chen, Baoyuan Wu, and Haoqian Wang. Effective backdoor defense by exploiting sensitivity of poisoned samples. In *NeurIPS*, 2022.

[20] 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 *SP*, 2019.

[21] Yi Zeng, Si Chen, Won Park, Zhuoqing Mao, Ming Jin, and Ruoxi Jia. Adversarial unlearning of backdoors via implicit hypergradient. In *International Conference on Learning Representations*, 2021.

[22] Yige Li, Xixiang Lyu, Xingjun Ma, Nodens Koren, Lingjuan Lyu, Bo Li, and Yu-Gang Jiang. Reconstructive neuron pruning for backdoor defense. *arXiv preprint arXiv:2305.14876*, 2023.

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

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

[25] 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.

[26] Mauro Barni, Kassem Kallas, and Benedetta Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. In *ICIP*, 2019.

[27] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *NeurIPS*, 2018.

[28] Shihao Zhao, Xingjun Ma, Xiang Zheng, James Bailey, Jingjing Chen, and Yu-Gang Jiang. Clean-label backdoor attacks on video recognition models. In *CVPR*, 2020.

[29] Khoa Doan, Yingjie Lao, Weijie Zhao, and Ping Li. Lira: Learnable, imperceptible and robust backdoor attacks. In *ICCV*, 2021.

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

[31] Khoa Doan, Yingjie Lao, and Ping Li. Backdoor attack with imperceptible input and latent modification. In *NeurIPS*, 2021.

[32] Kunzhe Huang, Yiming Li, Baoyuan Wu, Zhan Qin, and Kui Ren. Backdoor defense via decoupling the training process. In *ICLR*, 2022.

[33] Junfeng Guo, Ang Li, and Cong Liu. Aeva: Black-box backdoor detection using adversarial extreme value analysis. *arXiv preprint arXiv:2110.14880*, 2021.

[34] Wei Jiang, Xiangyu Wen, Jinyu Zhan, Xupeng Wang, Ziwei Song, and Chen Bian. Critical path-based backdoor detection for deep neural networks. *TNNLS*, 2022.

[35] Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. Universal litmus patterns: Revealing backdoor attacks in cnns. In *CVPR*, 2020.

[36] Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A Gunter, and Bo Li. Detecting ai trojans using meta neural analysis. In *SP*, 2021.

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

[38] Runkai Zheng, Rongjun Tang, Jianze Li, and Li Liu. Data-free backdoor removal based on channel lipschitzness. In *ECCV*, 2022.

[39] Tianlong Chen, Zhenyu Zhang, Yihua Zhang, Shiyu Chang, Sijia Liu, and Zhangyang Wang. Quarantine: Sparsity can uncover the trojan attack trigger for free. In *CVPR*, 2022.

[40] Jiyang Guan, Zhuozhuo Tu, Ran He, and Dacheng Tao. Few-shot backdoor defense using shapley estimation. In *CVPR*, 2022.

[41] Dongxian Wu and Yisen Wang. Adversarial neuron pruning purifies backdoored deep models. In *NeurIPS*, 2021.

[42] Shuwen Chai and Jinghui Chen. One-shot neural backdoor erasing via adversarial weight masking. In *NeurIPS*, 2022.

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

[44] Shaokui Wei, Mingda Zhang, Hongyuan Zha, and Baoyuan Wu. Shared adversarial unlearning: Backdoor mitigation by unlearning shared adversarial examples. *Advances in Neural Information Processing Systems*, 36, 2024.

[45] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 141–159. IEEE, 2021.

[46] Xiong Xu, Kunzhe Huang, Yiming Li, Zhan Qin, and Kui Ren. Towards reliable and efficient backdoor trigger inversion via decoupling benign features. In *The Twelfth International Conference on Learning Representations*, 2024.

[47] Shaokui Wei, Mingda Zhang, Hongyuan Zha, and Baoyuan Wu. Shared adversarial unlearning: Backdoor mitigation by unlearning shared adversarial examples. In *NeurIPS*, 2023.

[48] Yang Zhao, Hao Zhang, and Xiuyuan Hu. Penalizing gradient norm for efficiently improving generalization in deep learning. In *International Conference on Machine Learning*, pages 26982–26992. PMLR, 2022.

[49] Yi Zeng, Won Park, Z Morley Mao, and Ruoxi Jia. Rethinking the backdoor attacks' triggers: A frequency perspective. In *ICCV*, 2021.

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

[51] Baoyuan Wu, Hongrui Chen, Mingda Zhang, Zihao Zhu, Shaokui Wei, Danni Yuan, and Chao Shen. Backdoorbench: A comprehensive benchmark of backdoor learning. In *NeurIPS Datasets and Benchmarks Track*, 2022.

[52] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 2015.

[53] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *IJCNN*, 2011.

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

[55] Mingli Zhu, Shaokui Wei, Li Shen, Yanbo Fan, and Baoyuan Wu. Enhancing fine-tuning based backdoor defense with sharpness-aware minimization. In *ICCV*, 2023.

Appendix Outline

This appendix is organized as follows:

- In Section A, we detail the algorithm of TSBD.
- In Section B, we detail the approximation process of the fine-tuning optimizations.
- In Section C, we introduce the implementation details, including the details of datasets, models, attacks, and defenses with our proposed method.
- In Section D, we compare the defense results on the GTSRB dataset.
- In Section E, we compare the defense results on VGG19-BN structure.
- In Section F, we show the effectiveness of using NWC in other defense.
- In Section G, we show the comprehensive results with different neuron ratios and weight ratios.
- In Section H, we discuss different clean data ratios on the defense performance.
- In Section I, we discuss different fine-tuning learning rates on the defense performance.
- In Section J, we evaluate the influence of using our method on the clean model.
- In Section K, we test the performance of scaled-up experiments on the ViT model.
- In Section L, we show the computational overhead in terms of runtime.

A Detailed Algorithm of TSBD

To clearly illustrate our proposed method, we provide the detailed algorithm of the entire process of TSBD, which is shown in Algorithm 1.

B Details of the Approximated Fine-Tuning Optimizations

As illustrated in Section 3.4, our proposed Activaness-aware fine-tuning involves calculating an additional gradient-norm regulation in the loss function, making it computationally inefficient. Therefore, we adopt the approximation scheme from [48] during practical optimization. Here, we provide the details of the approximated fine-tuning optimization. The approximation deviation is shown in the following. Specifically, for each step of the gradient calculation, we formulate it as:

$$\begin{aligned}
\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ft}(\hat{\boldsymbol{\theta}}) &= \nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}}) + \lambda \cdot \nabla^2_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}}) \frac{\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}})}{\|\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}})\|_2} \\
&\approx \nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}}) + \frac{\lambda}{r} \cdot (\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}} + r\frac{\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}})}{\|\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}})\|_2}) - \nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}})) \\
&= (1-\alpha)\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}}) + \alpha\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}} + r\frac{\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}})}{\|\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}})\|_2}).
\end{aligned} \tag{6}$$

To avoid the Hessian computation, the second term is further approximated through an additional parameter update:

$$\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}} + r\frac{\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}})}{\|\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}})\|_2}) \approx \nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}})\big|_{\hat{\boldsymbol{\theta}}=\hat{\boldsymbol{\theta}}+r\frac{\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}})}{\|\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}})\|_2}}. \tag{7}$$

Based on the approximation, the practical fine-tuning process is illustrated in Algorithm 2.

C More Implementation Details

We further illustrate the implementations here, covering the details of datasets, models, attacks, and defenses.

Dataset Details. The experiments are conducted on CIFAR-10 [13], Tiny ImageNet [52], and GTSRB [53].

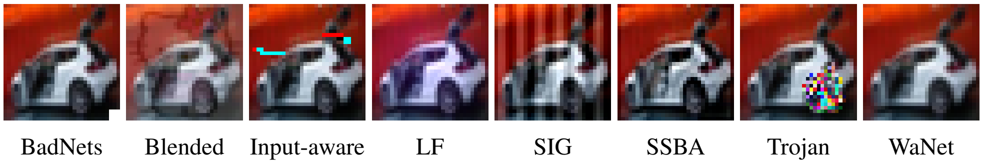


Figure 7: Examples of 8 backdoor-attack triggers on the same image of CIFAR-10.

- *CIFAR-10.* It contains 60,000 32×32 colored images with 10 classes. Each class owns 6,000 images, consisting of 5,000 for training and 1,000 for testing.
- *Tiny ImageNet.* It is a subset of the full ImageNet, consisting of 100,000 training data and 10,000 testing data. There are 200 classes in total and 500 images per class for training. All images are 64×64 with color.
- *GTSRB.* GTSRB (German Traffic Sign Recognition Benchmark) contains 39,209 images for training and 12,630 images for testing with 43 classes. All images are 32×32 colored images.

Models. We choose PreAct-ResNet18 [12] and VGG19-BN [54] as the target models to conduct attacks and defenses following the default configurations in BackdoorBench [51]. Both of them contain convolutional layers and batch normalization layers, which can be implemented with all kinds of defense methods, *e.g.*, FP [37] for the last convolutional layer and ANP [41] for the batch normalization layers. The extensive experiments on ablation study and further analysis are conducted on PreAct-ResNet18 by default.

Attack Details. We conduct 8 SOTA attacks for comprehensive testing, consisting of BadNets [8], Blended [25], Input-aware [23], LF [49], SIG [26], SSBA [9], Trojan [50] and WaNet [24]. All the attacks follow BackdoorBench's default configurations. Figure 7 shows all 8 attack triggers with the same example of CIFAR-10. Specifically, for BadNets, a 3×3 white square is patched at the bottom-right corner of the images for CIFAR-10 and GTSRB, and a 6×6 white square is for Tiny ImageNet. For Blended, a Hello-Ketty image is blended in the images with a 0.2 transparent ratio. We choose the 10% poisoning ratio and 0^{th} label as the default setting to conduct attacks and test all defenses following the previous works [55, 47]. 5% and 1% poisoning ratios are conducted only for testing our proposed method.

Defense Details. We conduct 8 SOTA defenses for a comprehensive comparison, containing Fine-tuning (FT), Fine-pruning (FP) [37], NAD [43], NC [20], ANP [41], CLP [38], i-BAU [21], and RNP [22]. The defenses also follow the BackdoorBench's default configurations. Note that we compare only the post-training defenses with 5% benign data provided. The learning rate for all methods is set to 10^{-2}, the batch size is set to 256. For RNP, the clean data ratio is set to 0.5% since we found that it failed to defend well under the 5% setting. For our proposed method, we set the default learning rates for unlearning as 10^{-4} and fine-tuning as 10^{-2}. The unlearning is stopped when clean accuracy drops to or below 10%. The default fine-tuning epoch is set to 20. The default neuron ratio n and weight ratio m are set to 0.15 and 0.7, respectively. We follow the suggested settings of hyper-parameters $r = 0.05$ and $\alpha = 0.7$ for fine-tuning [48]. Other settings are set following the default BackdoorBench configuration.

All experiments are conducted on a server with GPU RTX 3090 and CPU AMD EPYC 7543 32-Core Processor. These experiments were successfully executed using less than 24G of memory on a single GPU card.

D Evaluations on GTSRB dataset

We validate the effectiveness of our proposed method in the dataset GTSRB other than CIFAR-10 and Tiny ImageNet. Table 6 shows the corresponding performance on PreAct-ResNet18 with 10% poisoning ratio and 5% clean data ratio. We can observe that TSBD performs consistently with the lowest average ASR and the largest average DER as in CIFAR-10. Most of the defenses also fail in the strong attacks Blended, LF, and Trojan, while NC performs the second best with comparable average ASR and DER. Compared to the other two datasets, TSBD performs much superior with

Algorithm 1 Two-Stage Backdoor Defense

Input: Small clean set $\mathcal{D}_c$, backdoored model with parameter $\boldsymbol{\theta}_{bd}$, max iteration number T, neuron ratio $n\%$ and weight ratio $m\%$ for reinitialization
Output: Clean model with parameter $\boldsymbol{\theta}^*$

```
/* Neuron Weight Change-based Backdoor Reinitialization */
// a. Clean Unlearning
while Clean accuracy on $\boldsymbol{\theta}_{ul} > 10\%$ do
    Sample a mini-batch $\mathcal{B}_c$ from $\mathcal{D}_c$
    $\boldsymbol{\theta}_{ul} \leftarrow \max_{\boldsymbol{\theta}_{bd}} \mathcal{L}\left(f\left(\mathcal{B}_c; \boldsymbol{\theta}_{bd}\right)\right)$
end while
// b. Neuron Weight Change Calculation.
Record subweight changes and calculate NWC by Equation (2) w.r.t. $\boldsymbol{\theta}_{ul}$ and $\boldsymbol{\theta}_{bd}$
// c. Zero Reinitialization.
Obtain and sort the weight changes from the top-$n\%$ neurons w.r.t. NWC
$\hat{\boldsymbol{\theta}} \leftarrow$ reinitialize $m\%$ of the most-changing weights into zero on $\boldsymbol{\theta}_{bd}$
/* Activeness-aware Fine-tuning */
for $t = 1$ to $T$ do
    Sample a mini-batch $\mathcal{B}_c$ from $\mathcal{D}_c$
    Update $\hat{\boldsymbol{\theta}}$ by Equation (4) with the approximated Equation (6)
end for
$\boldsymbol{\theta}^* \leftarrow$ fine-tuned $\hat{\boldsymbol{\theta}}$
```

Algorithm 2 The Approximated Optimization of Activeness-aware Fine-tuning

Input: Small clean set $\mathcal{D}_c$, the reinitialized model with parameter $\hat{\boldsymbol{\theta}}$, max iteration number T, approximation scalar r, balance coefficient α
Output: Clean model with parameter $\boldsymbol{\theta}^*$

```
/* Activeness-aware Fine-tuning */
for $t = 1$ to $T$ do
    Sample a mini-batch $\mathcal{B}_c$ from $\mathcal{D}_c$
    Calculate the gradient $g_1 = \nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}})$ with $\mathcal{B}_c$
    Temporally update the current parameter $\hat{\boldsymbol{\theta}}' = \hat{\boldsymbol{\theta}} + r\frac{\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}})}{\|\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}})\|_2}$
    Calculate the gradient $g_2 = \nabla_{\hat{\boldsymbol{\theta}}'}\mathcal{L}_{ce}(\hat{\boldsymbol{\theta}}')$ with $\mathcal{B}_c$
    Calculate the final gradient $g = (1-\alpha)g_1 + \alpha g_2$
    Update the parameter $\hat{\boldsymbol{\theta}}$ based on $g$
end for
$\boldsymbol{\theta}^* \leftarrow$ fine-tuned $\hat{\boldsymbol{\theta}}$
```

most ASRs lower than 0.5% except for Blended attack. This suggests that TSBD might perform better when the model is learned with the input images with similar characteristics.

E Performance on VGG19-BN model structure

Except for PreAct-ResNet18, we also test another model, VGG19-BN. The performance on CIFAR-10 with 10% poisoning ratio and 5% clean data ratio is illustrated in Table 7. We follow similar settings in PreAct-ResNet18 on CIFAR-10, conducting 8 attacks and comparing our method with 8 defenses. The performance is also evaluated by ACC, ASR, and DER. As shown in the table, TSBD owns SOTA performance on VGG19-BN with the best average ACC and second-best average ASR and DER. Most of the other performances are in a similar pattern as on PreAct-ResNet18. Although ANP performs almost the best in ASR and DER for all attacks, it damages the corresponding ACC as well, especially for the WaNet attacked model, where ACC decreases from 84.58% to 78.04%. On the contrary, our methods succeed in most attacks with high ACC and comparable ASR.

Table 6: Comparison with the SOTA defenses on GTSRB dataset with PreAct-ResNet18 (%).

Backdoor	No Defense			FT			FP [37]			NAD [43]			NC [20]		
Attacks	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑
BadNets [8]	97.24	59.25	-	**98.73**	5.09	77.08	98.21	0.09	79.58	98.69	0.63	79.31	97.48	0.01	79.62
Blended [25]	98.58	99.99	-	98.57	100.00	50.00	98.38	100.00	49.90	98.61	100.00	50.00	97.76	8.03	**95.57**
Input-aware [23]	97.26	92.74	-	98.40	29.81	81.46	98.08	2.32	95.21	98.27	40.65	76.04	98.55	0.01	96.36
LF [49]	97.93	99.57	-	98.01	79.98	59.80	97.59	99.70	49.83	**98.14**	51.83	73.87	97.97	1.34	99.11
SSBA [9]	97.98	99.56	-	97.92	99.10	50.20	97.75	99.46	49.94	97.95	99.39	50.07	97.72	0.29	**99.50**
Trojan [50]	98.57	100.00	-	98.54	0.02	**99.97**	98.31	71.30	64.22	98.20	0.11	99.76	87.53	0.83	94.07
WaNet [24]	97.74	94.25	-	98.54	0.29	96.98	97.62	88.07	53.03	98.61	0.56	96.84	98.25	**0.00**	**97.12**
Average	97.90	92.19	-	**98.39**	44.90	73.64	97.99	65.85	63.10	98.35	41.88	75.13	96.47	1.50	94.48
Backdoor	ANP [41]			CLP [38]			i-BAU [21]			RNP [22]			TSBD **(Ours)**		
Attacks	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑
BadNets [8]	96.89	0.06	79.42	97.67	66.85	50.00	96.47	0.02	79.23	97.69	0.67	79.29	98.38	**0.00**	**79.63**
Blended [25]	98.75	99.82	50.09	98.48	100.00	49.95	92.35	86.35	53.71	**98.84**	99.64	50.18	95.57	**5.94**	95.52
Input-aware [23]	99.14	**0.00**	**96.37**	98.64	96.01	50.00	97.09	0.52	96.03	97.17	**0.00**	96.32	**99.23**	0.29	96.22
LF [49]	97.80	81.38	59.03	97.70	99.50	49.92	95.78	16.15	90.64	97.91	99.06	50.25	96.83	**0.03**	**99.22**
SSBA [9]	97.86	98.93	50.26	**98.08**	99.24	50.16	96.14	1.88	97.92	97.78	99.43	49.96	96.06	**0.10**	98.77
Trojan [50]	98.08	**0.00**	99.75	98.17	95.20	52.20	95.98	0.02	98.70	**98.56**	100.00	50.00	97.16	0.15	99.22
WaNet [24]	97.08	**0.00**	96.80	7.16	100.00	4.71	96.72	**0.00**	96.62	96.77	**0.00**	96.64	**98.80**	0.03	97.11
Average	97.94	40.03	75.96	85.13	93.83	43.85	95.79	14.99	87.55	97.81	56.97	67.52	97.43	**0.93**	**95.10**

Table 7: Comparison with the SOTA defenses on CIFAR-10 dataset with VGG19-BN(%).

Backdoor	No Defense			FT			FP [37]			NAD [43]			NC [20]		
Attacks	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑
BadNets [8]	90.42	94.43	-	88.19	27.59	82.31	88.96	10.23	91.37	86.48	5.47	92.51	89.21	11.31	90.96
Blended [25]	91.91	99.50	-	90.08	86.82	55.42	89.95	87.46	55.04	88.60	83.86	56.17	90.07	83.33	57.16
Input-aware [23]	88.66	94.58	-	**91.56**	13.08	90.75	91.35	6.08	94.25	91.00	14.11	90.23	89.70	97.02	50.00
LF [49]	83.28	13.83	-	87.67	1.82	56.01	88.31	1.23	56.30	83.72	1.14	56.34	86.64	1.36	56.24
SIG [26]	83.48	98.87	-	88.01	4.28	97.29	**88.34**	15.26	91.81	86.07	7.39	95.74	83.48	98.87	50.00
SSBA [9]	90.85	95.11	-	89.26	70.22	61.65	89.28	65.80	63.87	88.33	56.64	67.97	**90.85**	95.11	50.00
Trojan [50]	91.57	100.00	-	89.60	7.17	95.43	89.74	50.90	73.64	87.29	2.30	96.71	89.28	7.76	94.98
WaNet [24]	84.58	96.49	-	**91.35**	5.72	95.39	91.12	4.74	95.88	90.73	10.33	93.08	91.20	6.88	94.81
Average	88.09	86.60	-	89.47	27.09	79.28	89.63	30.21	77.77	87.78	22.65	81.10	88.80	50.20	68.02
Backdoor	ANP [41]			CLP [38]			i-BAU [21]			RNP [22]			TSBD **(Ours)**		
Attacks	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑
BadNets [8]	88.39	0.48	**95.96**	**89.38**	6.61	93.39	86.01	2.28	93.87	79.90	**0.12**	91.90	88.76	3.58	94.60
Blended [25]	89.19	**4.77**	**96.01**	**90.66**	98.59	49.83	87.58	69.90	62.64	26.92	43.20	45.66	90.51	6.68	95.71
Input-aware [23]	86.56	**0.71**	**95.88**	87.54	2.66	95.40	88.29	69.56	62.33	60.76	87.33	39.67	91.52	13.29	90.64
LF [49]	84.33	0.08	**56.88**	81.99	14.61	49.36	87.68	1.47	56.18	79.89	**0.00**	55.22	**88.71**	1.77	56.03
SIG [26]	82.69	**0.00**	**99.04**	82.12	98.68	49.41	83.41	5.37	96.72	35.36	0.01	75.37	88.14	2.58	98.14
SSBA [9]	89.81	1.34	**96.36**	85.82	98.56	47.49	87.56	22.26	84.78	69.02	**0.03**	86.62	88.97	3.41	94.91
Trojan [50]	89.39	**0.00**	**98.91**	**90.56**	99.70	49.65	88.63	8.23	94.41	53.40	57.30	52.27	90.22	6.63	96.01
WaNet [24]	78.04	**0.03**	94.96	88.46	1.75	97.37	89.76	1.61	**97.44**	87.90	81.03	57.73	91.28	2.50	97.00
Average	86.05	**0.93**	**91.75**	87.07	52.64	66.49	87.37	22.58	81.05	61.64	33.63	63.05	**89.76**	5.06	90.38

F Effectiveness of using NWC in Other Defense

As illustrated in Section 4.3, it is effective to utilize NWC order in gauging the backdoor strength. To further test its ability to improve other defenses, we substitute the average neuron activation in FP to NWC (denoted as NWC-FP) and conduct the experiments on the default settings in PreAct-ResNet18 on CIFAR-10. Different from zero reinitialization, the pruned neurons will not be updated on the following fine-tuning. The performances of 8 attacks are shown in Figure 8. We can investigate that, by using NWC in FP, performances on ASR and DER are improved on most of the attacks, with few effects on ACC. It exhibits the potential of using our NWC in more defense methods.

G Performance with Different Neuron Ratio and Weight Ratio

Section 4.4 shows some of the performances with different neuron ratios and weight ratios and verifies the robustness of the hyper-parameter tuning in our method. Here, we exhibit the full results under all

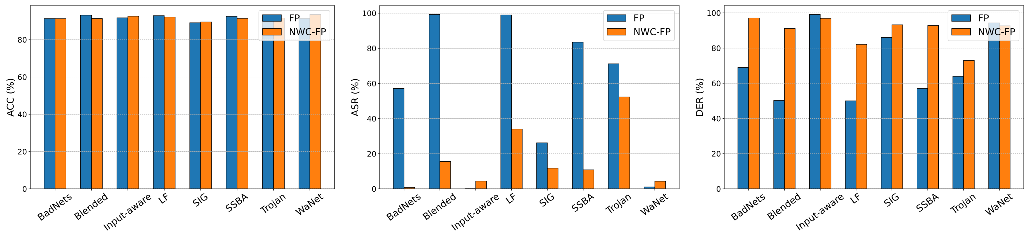


Figure 8: Performance comparisons between the original FP and the FP with NWC (NWC-FP) under 8 attacks. Left: ACC; Middle: ASR; Right: DER.

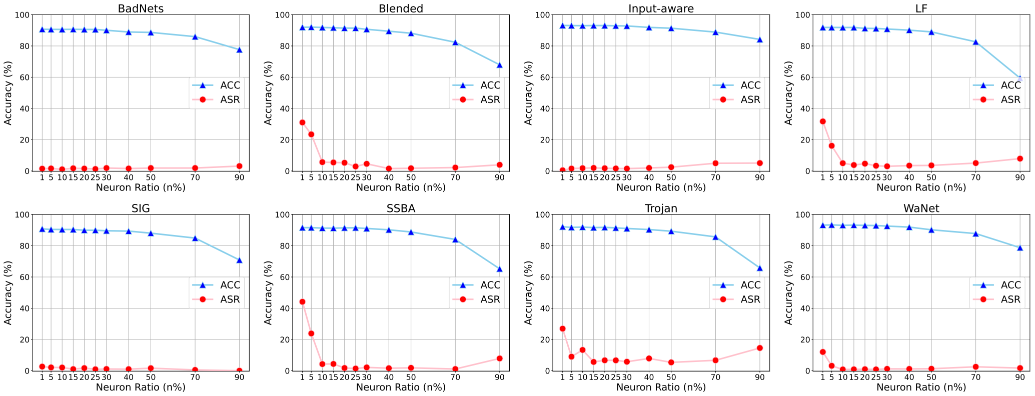


Figure 9: Performance with different neuron ratios under 8 attacks.

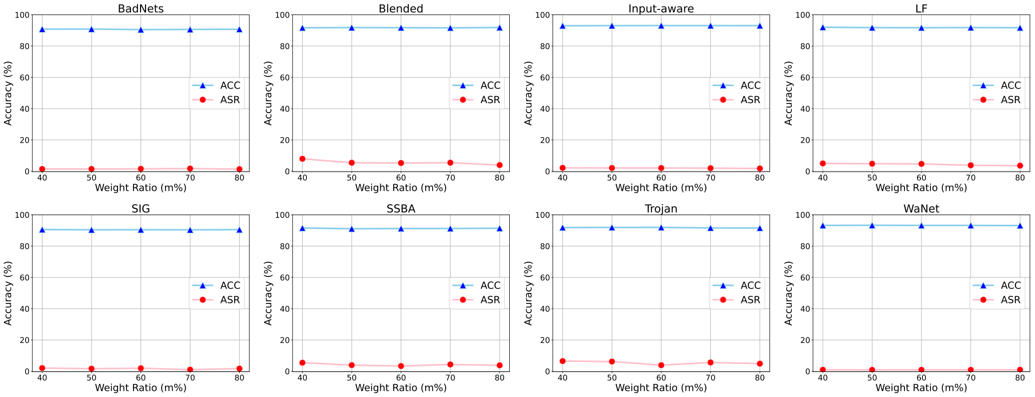


Figure 10: Performance with different weight ratios under 8 attacks.

8 attacks in PreAct-ResNet18 on CIFAR-10 with 10% poisoning ratio. Figure 9 and Figure 10 exhibit the tuning of neuron ratio and weight ratio, respectively. All performances are as good as expected.

H Evaluations with Different Clean Data Ratios

We are here to test the performance of TSBD under different clean data ratios. The performance on CIFAR-10 and PreAct-ResNet18 with 10% poisoning ratio is illustrated in Table 8. Except for the default 5% clean data ratio, we also test the performance under 10%, 1%, and 0.5%. We can observe that a larger ratio of clean data can always bring better performance in ACC, *e.g.*, 92.18% > 91.70% > 89.63% > 86.42% on average values of the four ratios in decreasing order, respectively. For the defense performance in ASR, all of them can successfully defend the attacks to under 10%, and large clean data ratios can achieve promising DERs. Overall, TSBD is robust to the clean data ratio, with only 0.5% clean data can also defend most tested defenses successfully.

I Evaluations with Different Learning Rates on Fine-tuning

We test the performance of our proposed method under different learning rates on fine-tuning. Table 9 illustrates the results. Specifically, with the same settings on CIFAR-10 and PreAct-ResNet18 with 10% poisoning ratio and 5% clean data ratio, we test the performance when the learning rate is set to 0.1, 0.01, 0.001, and 0.0001, where 0.01 is our default setting in the previous experiments. We can observe that the default learning rate of 0.01 is the most suitable one chosen for defense, which can achieve SOTA ASR and DER on average with comparable average ACC. Besides, setting the learning rate to 0.001 can perform the best in ACC, while making it fail on some strong attacks, *e.g.*,

Table 8: Performance with Different Clean Data Ratios on CIFAR-10 dataset with PreAct-ResNet18 (%).

Clean Data Ratio	- (No Defense)			10% (TSBD)			5% (TSBD)			1% (TSBD)			0.5% (TSBD)		
Attacks	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑
BadNets [8]	91.32	95.03	-	**91.44**	**0.82**	**97.11**	90.72	1.37	96.53	87.42	2.00	94.57	81.13	1.18	91.83
Blended [25]	93.47	99.92	-	**91.76**	5.79	96.21	91.61	2.61	**97.73**	89.36	**1.32**	97.25	86.68	4.80	94.17
Input-aware [23]	90.67	98.26	-	**93.65**	1.56	**98.35**	93.06	1.94	98.16	91.54	2.62	97.82	89.51	**1.43**	97.83
LF [49]	93.19	99.28	-	**92.24**	4.62	96.85	91.20	**2.64**	**97.32**	89.58	8.41	93.63	88.49	5.00	94.79
SIG [26]	84.48	98.27	-	**90.41**	1.01	98.63	**90.41**	1.27	98.50	88.53	**0.96**	**98.65**	83.55	2.44	97.45
SSBA [9]	92.88	97.86	-	**91.94**	2.14	97.39	91.57	1.66	**97.44**	89.24	1.71	96.25	84.46	**0.70**	94.37
Trojan [50]	93.42	100.00	-	**92.37**	7.00	95.98	91.76	5.06	**96.64**	90.18	**4.66**	96.05	87.98	7.21	93.68
WaNet [24]	91.25	89.73	-	**93.66**	0.98	94.38	93.26	0.88	**94.43**	91.19	1.24	94.22	89.57	**0.69**	93.68
Average	91.34	97.29	-	**92.18**	2.99	96.86	91.70	**2.18**	**97.09**	89.63	2.87	96.05	86.42	2.93	94.72

Table 9: Performance with Different Learning Rates of Fine-tuning on CIFAR-10 dataset with PreAct-ResNet18 (%).

Learning Rate	- (No Defense)			0.1 (TSBD)			0.01 (TSBD)			0.001 (TSBD)			0.0001 (TSBD)		
Attacks	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑	ACC ↑	ASR ↓	DER ↑
BadNets [8]	91.32	95.03	-	78.11	4.19	88.82	90.72	**1.37**	**96.53**	**91.62**	2.10	96.47	91.44	4.87	95.08
Blended [25]	93.47	99.92	-	74.79	**0.57**	90.34	91.61	2.61	**97.73**	92.85	15.19	92.06	**92.88**	29.33	85.00
Input-aware [23]	90.67	98.26	-	88.70	3.00	96.64	93.06	1.94	98.16	**93.22**	2.05	98.10	91.46	**0.64**	**98.81**
LF [49]	93.19	99.28	-	79.38	3.50	90.98	91.20	**2.64**	**97.32**	**92.90**	11.98	93.50	92.79	37.03	80.92
SIG [26]	84.48	98.27	-	73.15	3.68	91.63	90.41	**1.27**	**98.50**	**90.87**	4.81	96.73	89.23	14.47	91.90
SSBA [9]	92.88	97.86	-	79.90	4.21	90.33	91.57	**1.66**	**97.44**	92.64	9.26	94.18	**92.72**	15.27	91.21
Trojan [50]	93.42	100.00	-	73.83	32.87	73.77	91.76	**5.06**	**96.64**	**92.75**	10.76	94.29	92.73	21.07	89.12
WaNet [24]	91.25	89.73	-	87.61	1.48	92.31	**93.26**	0.88	94.43	93.11	0.79	94.47	91.36	**0.31**	**94.71**
Average	91.34	97.29	-	79.43	6.69	89.35	91.70	**2.18**	**97.09**	**92.50**	7.12	94.97	91.83	15.37	90.84

Blended and LF. On the contrary, setting the learning rate to 0.1 will hurt the ACC greatly, which may imply that clean knowledge cannot be re-learned properly.

J Evaluations on Clean Model

To test whether our defense method will hurt the performance of the clean model if no backdoor attack occurs, we compare it with several defense methods on CIFAR-10 and Tiny ImageNet datasets on PreAct-ResNet18. Figure 11 illustrates the results, where the ACC and ASR on the original clean model, ANP, and RNP, are used for comparison. We can observe that most of the defense methods (including ours) will barely damage the clean model though there is no backdoor occurs, except for RNP on Tiny ImageNet. It exhibits the potential to largely employ our defense methods in real-world AI systems.

K Performance on Scaled-Up Experiments

To further verify the scalability of our method, we evaluate its performance on a ViT-b-16 model with the CIFAR10 dataset following the basic settings in Section 4.1, *e.g.*, the poisoning ratio is set to 10% and the target label is set to 0. The results are shown in Table 10. The results demonstrate that TSBD performs effectively on the scaled-up model, achieving a low ASR and acceptable ACC. In contrast, CLP and ANP fail completely with ASR still at a high level, particularly for the WaNet attack.

L Computational Overhead

To show the computational overhead of TSBD, we record the average computational time of each defense step and its practical runtime compared to other defense methods. Table 11 exhibits the computational time of TSBD, including *Clean Unlearning*, *NWC Calculation*, *Zero Reinitialization*, and *Activeness-Aware Fine-Tuning*. We observe that the main computational overhead lies in the fine-tuning process. In contrast, the time required for clean unlearning does not increase proportionally with dataset complexity. This means that TSBD is as efficient as other fine-tuning-based methods. Moreover, we present a practical runtime comparison with other SOTA defenses in Table 12, including the loading and testing time needed in practice. As we can see, TSBD is faster than most of the existing methods.

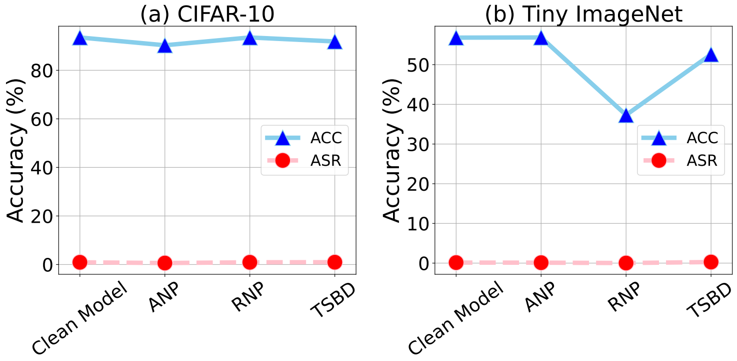


Figure 11: Performance of Defenses on Clean Model.

Table 10: Performance on CIFAR-10 with ViT-b-16 (%).

Attacks	No Defense		CLP [38]		ANP [41]		TSBD	
	ACC↑	ASR↓	ACC↑	ASR↓	ACC↑	ASR↓	ACC↑	ASR↓
Input-aware [23]	91.65	92.30	90.55	79.34	90.40	50.69	86.11	**4.43**
WaNet [24]	89.12	80.95	89.12	80.95	89.12	80.95	88.43	**1.59**

Table 11: Computational Time of Each Defense Step of TSBD

Defense Step	CIFAR-10	Tiny ImageNet
Clean Unlearning	20.84s	17.90s
NWC Calculation	0.03s	0.03s
Zero Reinitialization	1.34s	1.29s
Activeness-Aware Fine-Tuning	21.08s	174.36s

Table 12: Practical Runtime Comparison on BackdoorBench

Datasets	FT	FP [37]	ANP [41]	NC [20]	RNP [22]	TSBD
CIFAR-10	358s	855s	505s	733s	**123s**	159s
Tiny-ImageNet	1649s	20429s	2578s	37101s	285s	**269s**

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The contributions and scope are accurately written in the Introduction Section.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The last sentence of the Conclusion section clearly states the limitation and potential solution.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: All the results are empirically illustrated.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All experimental settings are clearly illustrated in the Experiment section and Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All the data are open-sourced. The code is provided in **Supplementary Material**, and will be publicly available.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: All settings and details are provided in the Experiment section and Appendix C.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: We strictly follow the basic settings in the open-sourced BackdoorBench [51], which ensures the fairness of all the comparisons in the Experiment section.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The computer resources are included in Appendix C, and the official information from BackdoorBench, since we follow their default settings.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We strictly conform the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The experiment in Appendix K exhibits the positive societal impacts of potentially employing our defense method without hurting the model performance in the real-world AI system.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our defense method has only a positive impact on use.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All licenses are explicitly mentioned.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide the documentation for the summit code in **Supplementary Material**, and it will be publicly available.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.