Reconstructive Neuron Pruning for Backdoor Defense

Yige Li [1] **Xixiang Lyu** [1] **Xingjun Ma** [2 3] **Nodens Koren** [4] **Lingjuan Lyu** [5] **Bo Li** [6] **Yu-Gang Jiang** [2 3]

Abstract

Deep neural networks (DNNs) have been found to be vulnerable to backdoor attacks, raising security concerns about their deployment in mission-critical applications. While existing defense methods have demonstrated promising results, it is still not clear how to effectively remove backdoor-associated neurons in backdoored DNNs. In this paper, we propose a novel defense called *Reconstructive Neuron Pruning* (RNP) to expose and prune backdoor neurons via an unlearning and then recovering process. Specifically, RNP first unlearns the neurons by maximizing the model's error on a small subset of clean samples and then recovers the neurons by minimizing the model's error on the same data. In RNP, unlearning is operated at the neuron level while recovering is operated at the filter level, forming an asymmetric reconstructive learning procedure. We show that such an asymmetric process on only a few clean samples can effectively expose and prune the backdoor neurons implanted by a wide range of attacks, achieving a new state-of-the-art defense performance. Moreover, the unlearned model at the intermediate step of our RNP can be directly used to improve other backdoor defense tasks including backdoor removal, trigger recovery, backdoor label detection, and backdoor sample detection. Code is available at https://github.com/bboylyg/RNP.

1. Introduction

Over the past decade, deep neural networks (DNNs) have demonstrated groundbreaking performance in solving various complex real-world problems. However, despite these significant successes, recent works have shown that DNNs are susceptible to different types of adversaries (Szegedy et al., 2013; Gu et al., 2017). Backdoor attacks represent one such type of adversary that injects malicious triggers into victim models during training either by poisoning the training data or manipulating the training procedure with specifically-designed trigger patterns. A backdoored model functions normally with clean data but predicts the pre-specified backdoor class whenever the trigger patterns appear. In this era, where pre-trained models and outsourced training via Machine Learning as a Service (MLaaS) are commonly adopted to achieve optimal performance at a minimal cost, the threats posed by backdoor attacks have increasingly become an issue we cannot overlook.

Numerous methods have been proposed to defend against backdoor attacks, with detection and removal methods being the two primary types. Detection methods identify whether a model has been backdoored (Wang et al., 2019; Guo et al., 2019; Liu et al., 2019; Xu et al., 2021; Shen et al., 2021; Hu et al., 2022) or whether a test sample contains a backdoor trigger (Tran et al., 2018; Chen et al., 2019; Tang et al., 2021; Zeng et al., 2021; Chen et al., 2022). Removal (or mitigation) methods are effective in purifying backdoored models by balancing the backdoor effect and their clean performance (Liu et al., 2018a; Li et al., 2021c; Wu & Wang, 2021). It is often observed in these methods that a backdoored DNN contains both clean and backdoor neurons, with the backdoor neurons being activated only by trigger patterns. The study conducted in (Wu & Wang, 2021) also shows that backdoor neurons are more sensitive to adversarial perturbations. Intuitively, if backdoor neurons can be accurately identified from a backdoored model, we can immediately prune those neurons to obtain a clean model.

To this end, we propose a novel method called *Reconstructive Neuron Pruning* (RNP) to expose and prune backdoor neurons from a backdoored model by unlearning and then recovering the neurons. Specifically, given a backdoored model, RNP first unlearns the model by maximizing its error on clean samples through gradient ascent and then recovers (relearns) the neurons by minimizing the model's error on the same samples. Interestingly, we find that if the unlearning is performed at the neuron level while the recovering is performed at the filter level, then the network tends to relocate the backdoor neurons to compensate for

[1]Xidian University [2]Fudan University [3]Shanghai Artificial Intelligence Laboratory [4]University of Copenhagen [5]Sony AI [6]University of Illinois at Urbana–Champaign. Correspondence to: Xixiang Lyu <xxlv@mail.xidian.edu.cn>, Xingjun Ma <xingjunma@fudan.edu.cn>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

the loss of clean features caused by the unlearning. Such an asymmetric operation can be very effective in locating the backdoor neurons with only a few clean samples (e.g., 500 images for the CIFAR-10 dataset). The backdoor neurons can then be easily pruned from the network.

With RNP, we have conducted so far the most extensive defense experiments in the current literature against 12 advanced backdoor attacks. Empirical results across different datasets and model architectures show that our RNP outperforms the current state-of-the-art method ANP (Wu & Wang, 2021) against 9/12 attacks on CIFAR-10 dataset and 5/5 attacks on an ImageNet subset. In some cases, our RNP only needs to remove 41 neurons from the backdoored (by BadNets (Gu et al., 2017)) model to reduce the attack success rate from 100% to 0.20%, while causing no significant clean accuracy degradation. Moreover, we demonstrate that the unlearned model obtained at the intermediate step of RNP can be directly used to improve other backdoor defense tasks.

To summarize, our main contributions are:

- We introduce the novel technique of neuron unlearning and recovering on the same set of samples and reveal that such a simple reconstruction-based learning process can help expose backdoor neurons in DNNs.
- We propose a new defense method called *Reconstructive Neuron Pruning* (RNP), which detects and prunes backdoor neurons via a neuron-level unlearning followed by a filter-level recovering with the help of a few clean samples.
- We empirically show that RNP outperforms existing backdoor defenses by a considerable margin against 12 advanced backdoor attacks, and that the unlearned models can aid in trigger recovery, backdoor label detection, and backdoor sample detection.

2. Related Work

2.1. Backdoor Attack

Depending on how the trigger pattern is crafted, existing backdoor attacks can be primarily categorized into two types: input-space attacks and feature-space attacks.

Input-space attacks. This type of attack injects a predefined trigger pattern into a small proportion of the training data to trick the model into learning the correlation between the trigger pattern and the backdoor label. The trigger pattern can be relatively simple, such as a single pixel (Tran et al., 2018), a black-white square (Gu et al., 2017), random noise (Chen et al., 2017), or more complex patterns such as adversarial perturbation (Turner et al., 2019), natural reflection (Liu et al., 2020), and sample-wise patterns (Nguyen & Tran, 2020; Li et al., 2021a; Wang et al., 2022a).

Table 1. Whether a defense technique can help backdoor detection (BD), trigger recovery (TR), or backdoor removal (BR).

DEFENSE	BD	TR	BR
NC	✓	✓	✗
STRIP	✓	✗	✗
FINE-PRUNING	✓	✗	✗
ABL	✗	✗	✓
I-BAU	✗	✗	✓
ANP	✗	✗	✓
RNP (OURS)	✓	✓	✓

Feature-space attacks. These attacks directly manipulate the training procedure to optimize the backdoor objective in the feature space (Shafahi et al., 2018; Cheng et al., 2021; Zhao et al., 2022) or directly modify the model parameters via weight perturbation (Garg et al., 2020; Qi et al., 2022b). These two types of attacks represent two typical threat models: input-space attacks only need access to a small subset of the training data, while feature-space attacks require full access to the training procedure or the final model. Input-space attacks could occur during the data collection process, while feature-space attacks could occur in outsourced training via MLaaS or when downloading pre-trained models from untrusted sources. As we will show in our experiments, feature-space attacks are generally more difficult to defend against than input-space attacks.

2.2. Backdoor Defense

Numerous approaches have been proposed to defend DNNs against backdoor attacks, among which backdoor detection and removal methods are the two most popular ones.

Backdoor detection. Several detection works identify backdoors based on the prediction bias on different input examples (Li et al., 2020) or the statistical deviation in the feature space (Tran et al., 2018; Chen et al., 2019; Liu et al., 2022). More effective detection methods leverage reverse engineering techniques to recover the trigger pattern and then identify the backdoor label by anomaly detection (Wang et al., 2019; Liu et al., 2019; Guo et al., 2019; Hu et al., 2022). The most representative method is Neural Cleanse (NC) (Wang et al., 2019), which recovers trigger patterns that can alter the model's predictions with minimum perturbation. There are also methods that detect backdoored samples at inference time, for example, the STRIP method (Gao et al., 2019).

Backdoor removal. Backdoor removal methods aim to erase the backdoors from backdoored models without significantly reducing their performance on clean samples. This line of work includes Fine-tuning, Fine-pruning (Liu et al., 2018a), Neural Attention Distillation (NAD) (Li et al.,

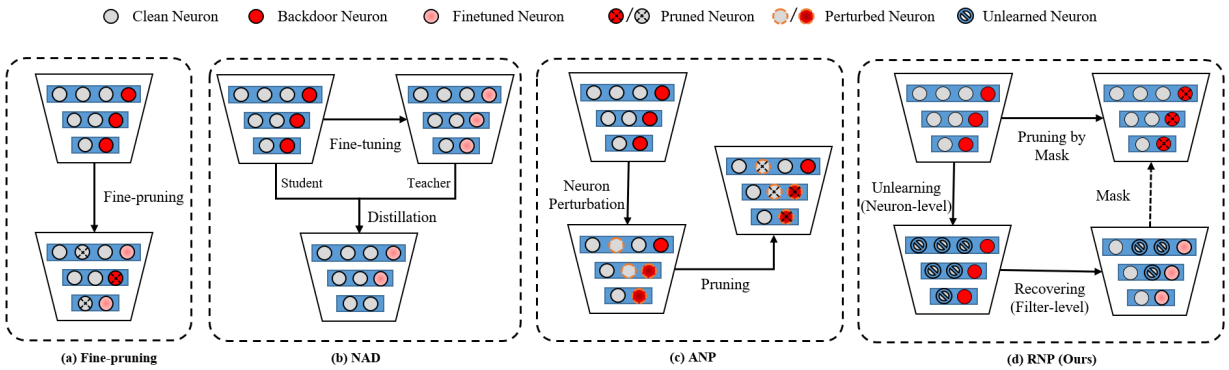

Figure 1. Overview of our proposed **RNP** framework, in comparison with 3 existing backdoor removal methods: **Fine-pruning**, **NAD**, and **ANP**. RNP exposes the backdoor via a neuron-level unlearning followed by a filter-level recovering. Pruning is then applied to remove the exposed backdoor neurons. Note that both ANP and our RNP do not need fine-tuning after the pruning.

2021c), Channel Lipschitzness Pruning (Zheng et al., 2022), and Shapley Estimation (Guan et al., 2022). More recently, training-time defense methods (Li et al., 2021b; Huang et al., 2022; Wang et al., 2022b) have been proposed to train clean models directly on backdoored data. Meanwhile, Adversarial Unlearning of Backdoors via Implicit Hypergradient (I-BAU) (Zeng et al., 2022) is proposed to cleanse backdoored models with adversarial training. Another recent work, Adversarial Neuron Pruning (ANP) (Wu & Wang, 2021), prunes those neurons that are more adversarially sensitive to remove backdoors. ANP has achieved a new state-of-the-art result against input-space attacks. Our experiments will show that ANP fails on more advanced feature-space attacks, indicating that adversarial perturbation may not be an ideal approach for exposing backdoor neurons that are deeply intertwined with the features. Table 1 summarizes the benefits of existing and our proposed RNP defense methods.

3. Proposed Method

In this section, we first introduce the threat model, then present our proposed defense, *Reconstructive Neuron Pruning* (RNP), and an illustrative example.

3.1. Threat Model

We assume the adversary has successfully injected a backdoor trigger into the target model. The defender's goal is to remove the backdoor trigger from the target model with minimal impact on its clean accuracy (accuracy on clean test samples), and at the same time, reveal as much information about the attack as possible, such as the trigger pattern and the backdoor label. Following prior works (Li et al., 2021c; Wu & Wang, 2021), we assume the defender has a small subset of clean data (e.g., 1% of carefully-examined training data) to develop the defense strategy, which we call *defense data*.

3.2. Reconstructive Neuron Pruning

Overview. Figure 1 illustrates our proposed RNP defense. At the core of RNP is a reconstructive learning process that first unlearns the neurons on the defense data via *Neuron Unlearning* (NU) and then recovers the neurons on the same data via *Filter Recovering* (FR). As the defense data is clean, NU tends to unlearn primarily the clean neurons, i.e., neurons associated with the clean features. The backdoor neurons, i.e., neurons associated with the backdoor features, are largely preserved in the unlearned model. As such, the unlearned model can be leveraged to improve other analyses such as trigger recovery, backdoor label detection, and backdoor sample detection.

The mechanisms of existing methods, Fine-pruning, NAD, and ANP are also illustrated in Figure 1 for comparison. Fine-pruning is a conventional pruning method that prunes those small-norm neurons from the backdoored model, while NAD (Li et al., 2021c) adopts the fine-tuned model as a teacher to distill neurons of the backdoored model. ANP (Wu & Wang, 2021) exploits adversarial perturbations to find neurons that are more sensitive to adversarial perturbations as backdoor neurons. Compared to the adversarial perturbation technique used by ANP, our RNP, with the asymmetric unlearning and recovering procedure, exposes more backdoor-associated neurons and leads to better backdoor purification at the pruning step.

Unlearning. Consider a standard K-class classification task on poisoned training data $\mathcal{D} = \mathcal{D}_c \cup \mathcal{D}_b$, with $\mathcal{D}_c = \{(\boldsymbol{x}_c^{(i)}, y_c^{(i)})\}_{i=1}^{N}$ representing the clean subset and $\mathcal{D}_b = \{(\boldsymbol{x}_b^{(j)}, y_b^{(j)})\}_{j=1}^{M}$ representing the backdoor subset. A backdoor attack, or training a backdoored model, can be viewed as a dual-task learning problem defined on both the clean subset $\mathcal{D}c$ (clean task) and the backdoor subset $\mathcal{D}b$

(backdoor task) as follows:

$$\arg\min_{\theta=\theta_c\cup\theta_b}\Big[\underbrace{\mathbb{E}_{(\boldsymbol{x}_c,y_c)\in\mathcal{D}_c}\mathcal{L}(f(\boldsymbol{x}_c,\ y_c;\ \theta_c))}_{\text{clean task}} + \underbrace{\mathbb{E}_{(\boldsymbol{x}_b,y_b)\in\mathcal{D}_b}\mathcal{L}(f(\boldsymbol{x}_b,\ y_b;\ \theta_b))}_{\text{backdoor task}}\Big], \tag{1}$$

where f is the victim model with parameters θ, $\mathcal{L}$ denotes the classification loss (e.g., cross-entropy), and θ_c and θ_b denote the clean and the backdoor neurons, respectively. The parameter space of the backdoored model can be decomposed into $\theta = \theta_c \cup \theta_b$ because there exists a high level of independence between the clean task and the backdoor task (Li et al., 2021c), i.e., backdoor attacks by design should not impact the model's performance on clean samples (Gu et al., 2017). Note that, although $\theta = \theta_c \cup \theta_b$, it does not mean θ_c cannot overlap with θ_b, i.e., it is possible that $\theta_c \cap \theta_b \neq \emptyset$.

Intuitively, a model can be unlearned with respect to a certain task by maximizing its loss on the data that defines the task, an opposite process to model training. Following the above formulation, there exist two possible strategies to unlearn a backdoored model: 1) maximize the model's loss on the backdoor data $\mathcal{D}_b$, or 2) maximize the model's loss on the defense data $\mathcal{D}_d$. The first strategy is infeasible for backdoor defense as the defender does not know the backdoor data in advance. This naturally leads us to the second strategy that solves the following maximization problem:

$$\max_{\theta}\ \mathbb{E}_{(\boldsymbol{x}_d,\ y_d)\in\mathcal{D}_d}\mathcal{L}(f\ (\boldsymbol{x}_d,\ y_d;\theta)), \tag{2}$$

where $\mathcal{L}$ is the cross-entropy loss, and $(\boldsymbol{x}_d,\ y_d)$ are the clean samples in the defense dataset and their labels. We denote the parameters of the unlearned model by $\hat{\theta}$. Note that the unlearning is performed at a neuron level for all neurons of the model, so it is termed as *Neuron Unlearning (NU)*.

The above unlearning strategy is extremely simple but can be very effective. In fact, it is not surprising that clean neurons will be effectively unlearned if the above maximization is applied to the entire clean data $\mathcal{D}_c$. Interestingly, we find that the unlearning can be easily achieved with only a few clean samples, i.e., the defense data required by the defender is extremely small (i.e., $|\mathcal{D}_d| \ll |\mathcal{D}|$). For example, on the CIFAR-10 dataset, 1% of the clean training data is sufficient to expose the backdoor neurons injected by a wide range of advanced backdoor attacks. Moreover, the unlearning can be safely terminated when the performance of the model on defense data $\mathcal{D}_d$ is close to random guess. Note that the backdoor label inference step is performed on the unlearned model with the defense data, i.e., the unlearned model tends to predict the backdoor label for all defense samples (one unique property of the unlearned model as further explained in Appendix D.3).

Algorithm 1 Reconstructive Neuron Pruning (RNP)

Input: A backdoored model $f_\theta(\cdot)$ with parameter θ, the total number of classes K, defense data $\mathcal{D}_d$, learning rate η, clean accuracy threshold CA_{min}, dynamic threshold DT in [0, 1]

Sample a mini-batch (X_d, Y_d) from $\mathcal{D}_d$
 # Neuron-level unlearning
repeat
  $\hat{\theta} \leftarrow \max_{\theta} \mathcal{L}(f\ (\boldsymbol{x}_d,\ y_d; \theta))$
until $f_{\hat{\theta}}$'s clean accuracy $CA_{f_{\hat{\theta}}}(\mathcal{D}_d) \leq CA_{min}$
Backdoor label: $y_t = \arg\max_{K} f(\boldsymbol{x}_d;\ \hat{\theta})$
 # Filter-level recovering
$\mathbf{m}^\kappa = [1]^n$ # initialized to be all ones
repeat
  $\mathbf{m}^\kappa = \mathbf{m}^\kappa - \eta\frac{\partial\mathcal{L}(f(X_d,Y_d;\ \mathbf{m}^\kappa\odot\hat{\theta}))}{\partial\mathbf{m}^\kappa}$
  $\mathbf{m}^\kappa = clip_{[0,1]}(\mathbf{m}^\kappa)$ # 0-1 clipping
until training converged
 # Pruning
$\mathbf{m}^\kappa = \mathbb{I}\,(\mathbf{m}^\kappa > DT)$ # binarization for pruning

Output: $f_{\mathbf{m}^\kappa\odot\theta}$, y_t

Recovering. This step aims to recover the clean features (features of the clean samples) erased by the previous unlearning step. Recovering can be effectively done by updating the unlearned model to minimize its classification loss on the defense data, a process similar to model training. In our RNP, the recovery is performed on the filters rather than the neurons, and a filter mask is used to locate potential backdoor filters (and their associated neurons). We term this technique as *Filter Recovering (FR)*. Formally, FR solves the following minimization problem to learn the mask:

$$\min_{\mathbf{m}^\kappa\in[0,1]^n}\ \mathbb{E}_{(\boldsymbol{x}_d,/y_d)\in\mathcal{D}_d}\mathcal{L}(f(\boldsymbol{x}_d,\ y_d;\mathbf{m}^\kappa\odot\hat{\theta})), \tag{3}$$

where $\boldsymbol{x}_d \in \mathcal{D}_d$ is the defense data, $\mathcal{L}$ is the cross-entropy loss, $\hat{\theta}$ are the parameters of the unlearned model obtained via NU, and $\mathbf{m}^\kappa$ is a mask applied on the filters. Finding the optimal mask $\mathbf{m}^\kappa$ can be viewed as a process to restore as many clean filters as possible to recover the model's clean features on $\mathcal{D}_d$.

The question is why recovering should be applied to the filters. Intuitively, if unlearning is also applied to the neurons, it will become a direct reversal of the unlearning, ending up with the unlearned clean neurons restored to their original values. This does not help expose the backdoor neurons. Filter-level recovering restricts the freedom of recovery to a coarser granularity, thus forcing the model to reuse the dormant backdoor neurons to compensate for the loss of the clean features. The clean neurons in this process are also relearned to recover the clean functionality, which, however, is not sufficient due to the limitation of the

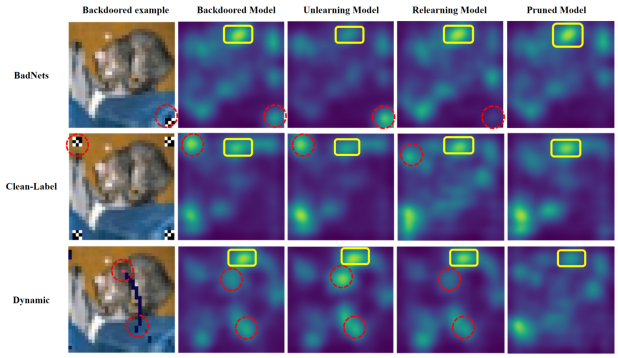

Figure 2. The feature maps (channel-wise averaged) of one backdoored image at the second residual block of a backdoored ResNet-18 model (by BadNets), and the unlearned, the recovered, and the pruned models by our RNP. The rectangles and circles highlight the regions that are mostly activated by the clean and backdoor patterns, respectively.

filter-level recovery. Neuron unlearning and filter recovering form an asymmetric unlearning-recovering (maximization-minimization) mechanism to expose backdoor neurons. We will provide empirical understandings of the mechanism and the necessity of unlearning-recovering in Section 4.3.

Pruning. The elements in the filter mask $\mathbf{m}^\kappa$ are all initialized to be one and clipped to be within the range of $[0, 1]$ during the recovering process. After recovering, a low value close to zero in $\mathbf{m}^\kappa$ indicates that the filter (and its associated neurons) contains mostly repurposed neurons that are likely to be backdoor-related. These neurons can thus be pruned to purify the backdoored model. As shown in Figure 2, in the recovered model, the activations related to the trigger pattern are greatly decreased (mask value decreases to almost zero) while those of the clean features are significantly boosted (mask value remains close to 1 due to the clipping operation).

The complete procedure of our RNP is described in Algorithm 1, where pruning is applied to the original model f_θ based on the learned filter mask $\mathbf{m}^\kappa$. The optimal pruning rate can be flexibly determined via dynamic thresholding in $[0, 1]$. The idea is to prune as many neurons as possible until the drop in the clean accuracy becomes unacceptable. In our experiments, we take dynamic thresholding as our default setting, unless otherwise stated. More analyses of different pruning rate determination strategies and the number of neurons pruned by our RNP can be found in Table 8 (see Appendix B.2) and Table 3, respectively.

3.3. An Illustrative Example

Here, we provide an illustrative example to understand and verify that asymmetric unlearning and recovering can expose backdoor neurons.

Figure 2 visualizes the feature maps of a backdoored image at the second residual block of a backdoored ResNet-18 model. Specifically, at the unlearning step, the clean neurons (features) are unlearned to have much weaker signals (indicated by the blurry feature map of the unlearned model), especially the most salient features (those in the yellow rectangles). At the recovering step, those weakened clean features will be boosted back to their initial intensity, so their corresponding mask values will increase (rather than decrease). Conversely, the backdoor neurons will be repurposed by the recovering process to compensate (or help boost) the weakened clean neurons, so their mask values will be largely decreased (close to zero), and their activations will almost "disappear". This can be observed from the dark blue region of the *recovered model* at the bottom right corner (marked by the red dashed circle) of the feature map. Note that the mask $\mathbf{m}^\kappa$ is an element-wise scaling matrix applied to each filter, as defined in Eqn. 3.

The above result indicates that the neuron-level unlearning erases the clean feature (although not completely) while keeping the backdoor feature almost unchanged. The recovered model relocates the neurons associated with the backdoor to compensate for the clean feature, and the backdoor is completely removed from the pruned model. More understanding of why asymmetric unlearning and recovering is the key to exposing backdoor neurons can be found in Section 4.3.

4. Experiments

4.1. Experimental Setup

Attack Setup. We primarily consider 12 state-of-the-art backdoor attacks. These include 7 input-space attacks: BadNets (Gu et al., 2017), Trojan (Liu et al., 2018b), Blend (Chen et al., 2017), Dynamic (Nguyen & Tran, 2020), WaNet (Nguyen & Tran, 2021), SIG (Barni et al., 2019), and CL (Turner et al., 2019), as well as 3 feature-space attacks: FC (Shafahi et al., 2018), DFST (Cheng et al., 2021), and AWP (Garg et al., 2020). In addition, two recently proposed adaptive attacks termed LIRA (Doan et al., 2021) and Adaptive-Blend (A-Blend) (Qi et al., 2022a) are also considered. We follow the default settings suggested in their original papers and the open-source codes for most attacks, including the trigger pattern, trigger size, and backdoor label. As in previous works (Wu & Wang, 2021; Li et al., 2021b), we evaluate the defense performance against the 12 attacks on the CIFAR-10 dataset and an ImageNet-12 dataset. The backdoor label of all attacks is set to class 0. The detailed settings of these attacks and datasets are summarized in Table 6. It should be noted that some of the attacks are not considered for ImageNet-12 because 1) they were not proposed for ImageNet and 2) they failed to reproduce on ImageNet-12.

Table 2. Performances of 5 backdoor defense methods against 12 backdoor attacks. The experiments were done on CIFAR-10 with ResNet-18 and ImageNet-12 subset with ResNet-34 using only 1% clean defense data. ASR: attack success rate (%); CA: clean accuracy (%). The best results are **boldfaced**.

Datasets	Backdoor Attacks	No Defense		FP		NAD		I-BAU		ANP		RNP (Ours)	
		ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA
CIFAR-10	BadNets	100.00	93.40	15.11	88.15	1.20	90.64	15.50	91.18	0.53	91.61	**0.20**	**92.22**
	Trojan	99.90	93.15	56.51	85.43	5.68	88.72	12.78	90.46	**1.00**	92.37	2.23	**92.56**
	Blend	100.00	93.10	68.22	85.21	10.92	89.77	1.62	90.16	0.50	92.31	**0.33**	**92.62**
	CL	100.00	94.84	24.38	88.75	13.17	89.76	23.12	88.82	15.20	92.86	**8.87**	**93.12**
	SIG	90.86	94.59	19.16	87.88	0.64	89.40	29.32	89.67	1.19	92.97	**0.43**	**94.62**
	Dynamic	99.97	91.36	41.73	83.28	13.60	88.64	18.74	86.87	**9.20**	89.66	15.24	**90.18**
	WaNet	99.10	93.67	68.92	86.35	17.46	82.41	23.18	87.38	13.14	92.64	**10.98**	**92.83**
	FC	100.00	94.67	98.45	87.62	36.07	88.02	17.93	86.75	74.75	81.97	**1.80**	**90.93**
	DFST	100.00	94.52	88.78	85.32	12.70	88.72	25.58	87.44	10.80	90.66	**4.61**	**92.78**
	AWP	94.39	94.30	23.17	86.04	1.71	89.13	8.71	89.62	**0.67**	92.64	1.04	**94.24**
	LIRA	100.00	92.71	87.78	83.12	32.12	86.73	51.33	82.56	20.25	87.78	**13.51**	**92.26**
	A-Blend	71.86	92.16	85.22	81.82	18.51	85.23	33.38	85.91	23.71	90.16	**1.09**	**90.38**
	Average	96.34	93.54	56.45	85.75	13.65	88.10	21.77	88.07	14.25	90.64	**5.03**	**92.18**
ImageNet-12	BadNets	100.00	88.53	91.70	83.23	9.12	83.26	15.38	85.15	10.25	85.21	**5.80**	**85.83**
	Trojan	100.00	89.79	93.69	81.40	12.31	82.52	19.61	84.11	7.48	87.41	**0.59**	**89.30**
	Blend	99.90	89.44	92.14	82.13	28.76	82.93	9.34	82.27	6.21	86.40	**5.54**	**86.89**
	SIG	73.78	88.18	87.82	81.27	21.15	83.31	29.23	81.57	25.53	52.52	**15.20**	**84.15**
	FC	95.77	88.95	90.52	79.36	31.43	81.56	38.51	79.33	42.69	53.01	**17.23**	**83.36**
	Average	93.89	88.98	91.17	81.48	20.55	82.72	22.41	82.49	18.43	72.91	**8.87**	**85.91**

Defense Setup. We consider a total of 8 backdoor defense methods, which include 2 backdoor detection methods: Neural Cleanse (NC) (Wang et al., 2019) and STRIP (Gao et al., 2019), 5 existing backdoor removal methods: Fine-pruning (FP) (Liu et al., 2018a), Neural Attention Distillation (NAD) (Li et al., 2021c), Adversarial Unlearning of Backdoors via Implicit Hypergradient (I-BAU) (Zeng et al., 2022), Adversarial Neuron Perturbation (ANP) (Wu & Wang, 2021), Anti-backdoor Learning (ABL) (Li et al., 2021b), and lastly, our RNP. All defenses share limited access to only 500 clean samples as their defense data (for both CIFAR-10 and ImageNet-12). Two typical data augmentation techniques (horizontal flip and random crop) are applied. We follow the open-source codes of FP, NAD, and ANP, and adjust their hyper-parameters to obtain the best performance on different attacks. For our RNP, the maximum unlearning epochs are set to 20 and the actual unlearning epochs range from 5 to 15, depending on the datasets, models, and attacks.

Evaluation Metrics. We adopt three metrics for defense evaluation: 1) Detection Rate (DR), which is the success rate of the defense in identifying the backdoor label or the backdoored model; 2) Clean Accuracy (CA), which is the model's accuracy on clean test data; and 3) Attack Success Rate (ASR), which is the model's accuracy on backdoored test data.

4.2. Main Defense Results

Results on CIFAR-10. Table 2 reports the defense performances of 5 backdoor defense methods against the 12 backdoor attacks on CIFAR-10. It is clear that our RNP achieves the best defense performance by reducing the average ASR from 96.34% to 5.03% with the minimum drop of CA (less than 2% on average). In contrast, FP, NAD, I-BAU, and ANP only reduce the average ASR to 56.45%, 13.65%, 21.77%, and 14.25%, respectively.

We notice that each defense method has its limitations against some specific attacks. For instance, even though RNP has the best overall performance, it is weaker than ANP in defending against the Dynamic attack by approximately 6.04% of ASR. On the other hand, while ANP achieves considerable results against most attacks, it has much poorer performance on FC, reducing the ASR only to 74.75%. We speculate that the adversarial perturbation in ANP cannot effectively locate the backdoor neurons when the backdoor features were optimized to collide with the clean features (and so as the neurons) by the FC attack. NAD also mirrors a similar pattern, which struggles to defend against FC, WaNet, and DFST. Meanwhile, our RNP also outperforms the recent defense I-BAU on all attacks. In terms of ASR, RNP surpasses I-BAU by more than 15% against BadNets, CL, SIG, and FC attacks, more than 10% against WaNet, DFST, and more than 1% against Blend and Dynamic. In terms of CA, RNP outperforms I-BAU

Table 3. The number of neurons pruned by ANP and our RNP against 12 types of backdoor attacks on the CIFAR-10 dataset. $Neurons \downarrow$ indicates the total number of neurons being pruned, and *on All* means the pruning across all the blocks of ResNet-18.

Defense	Metric	BadNets	Trojan	Blend	CL	SIG	Dynamic	WaNet	FC	DFST	AWP	LIRA	A-Blend
ANP	$Neurons \downarrow$ (on All)	94	96	42	135	88	69	126	199	165	56	158	96
	ASR (%)	0.53	1.00	0.50	15.20	1.19	9.20	13.14	74.75	10.80	0.67	20.25	23.71
	CA (%)	91.61	92.37	92.31	92.86	92.97	89.66	92.64	81.97	90.66	92.64	87.78	90.16
RNP	$Neurons \downarrow$ (on All)	**41**	**48**	**28**	**103**	**73**	**59**	**92**	**155**	**83**	**40**	**112**	**78**
	ASR (%)	0.20	2.23	0.33	8.87	0.43	15.24	10.98	1.80	4.61	1.04	13.51	1.09
	CA (%)	92.22	92.56	92.62	93.12	94.62	90.18	92.83	90.93	92.78	94.24	92.96	90.38

by more than 3% against SIG, Dynamic, WaNet, and FC. Finally, FP has the poorest overall performance with an average ASR higher than 50% against most attacks, indicating that pruning the most dormant neurons on the clean inputs is not accurate enough against advanced attacks. More results of our RNP with different DNN architectures can be found in Appendix B.3.

Results on ImageNet-12. The results on the ImageNet-12 dataset are also presented in Table 2. It is evident that our RNP surpasses all 4 baselines in terms of either ASR or CA. Particularly, RNP outperforms FP, NAD, I-BAU, and ANP by 82.3%, 11.68%, 13.54%, and 9.56% more ASR reduction respectively, with only $\leq 3\%$ decline in CA. Note that the two current SOTA defenses I-BAU and ANP failed to defend against the SIG and FC attacks to a large extent. More specifically, I-BAU/ANP only reduces the ASR to 29.23%/25.53% against SIG and 38.51%/42.69% against FC attacks. Our RNP is more effective against these two attacks than I-BAU and ANP, decreasing their ASRs to 15.20%, and 17.23%, respectively. This is somewhat unsurprising that the two attacks become more effective on high-resolution images, as SIG adds large and repetitive sinusoidal patterns into the background while FC directly attacks the representation space.

In summary, our RNP achieves superior performance against a wide range of attacks compared to the 4 baselines. This is due to RNP's ability to expose and identify backdoor neurons via its asymmetric unlearning-recovering process, leading to a more precise pruning effect. The sensitivity of RNP to different model architectures and poisoning rates are provided in Appendix B.3 and Appendix B.6, respectively.

The Number of Neurons Pruned by RNP. Here, we show the number of neurons pruned by ANP and our RNP against 12 backdoor attacks and report the results on the CIFAR-10 dataset in Table 3. These results show that 1) only a few neurons in a backdoored DNN are responsible for the backdoor functionality; and 2) complex attacks such as CL, WaNet, FC, and LIRA tend to create more backdoor neurons, and in this case, ANP has to prune more neurons than our RNP to reduce the ASR. For instance, ANP needs to prune 94, 96, and 42 neurons to defend against input-space attacks BadNets, Trojan, and Blend, respectively. By contrast, our RNP can achieve comparable or even better defense performance by pruning roughly half the number of neurons. A similar advantage of RNP can also be observed against other advanced attacks. The above findings shed new light on the working mechanism of backdoor attacks and also verify the preciseness of our RNP in locating the backdoor neurons. The results of RNP under different pruning strategies can be found in Appendix B.2.

RNP against Strong Adaptive Attacks. Here, we test the resistance of our RNP defense to strong adaptive attacks. We design two adaptive attacks: 1) Adaptive-distillation, which leverages knowledge distillation to align the neuron activation of a backdoored student network with that of a cleanly trained teacher network; and 2) Adv-training, which adversarially perturbs the backdoor neurons identified by our RNP and then fine-tunes the model on the clean subset of defense samples. Both attacks are designed to make exposing backdoor neurons very difficult. Table 4 reports the defense results of ANP and our RNP against the 2 adaptive attacks. It is clear that ANP has failed to defend against the 2 adaptive attacks to some extent (ASR=24.81% and 54.01%). Our RNP, on the contrary, is fairly robust to both attacks with the ASR reduced to 13.22% and 18.09% respectively while maintaining a high CA.

In the future, it is certainly possible that more advanced attacks could break our defense. However, in the current literature, we believe our RNP, as a simple and general framework, has proven itself to be the most effective defense against the most diverse attacks. We have also tested RNP's effectiveness against all-to-all attacks in Appendix B.4 and various trigger sizes in Appendix B.5.

Impact of the Defense Data Size. In this part, we explore the impact of the size of the defense data used to unlearn the backdoored model. We monitor the performance of RNP under varying sizes of the defense (unlearning) set, including 0.5% (250), 1% (500), 5% (2500), and 10% (5000) of the original clean training set, respectively, and present the results in Figure 3. We find that the more clean data used

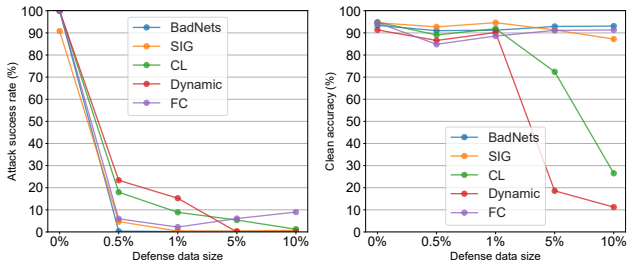

Figure 3. Performance (ASR/CA) of RNP under *different defense data sizes* (%) against 5 attacks.

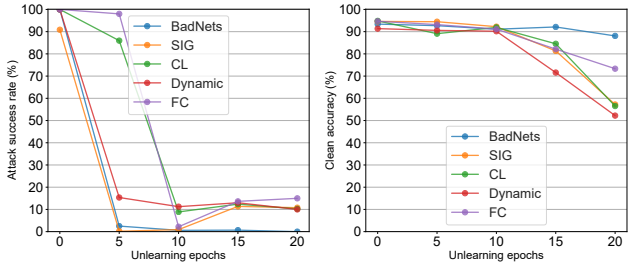

Figure 4. Performance (ASR/CA) of RNP under *different unlearning epochs* against 5 attacks.

Table 4. Defense results of ANP and our RNP on CIFAR-10 dataset against 2 adaptive attacks: 1) Adaptive-distillation (Adapt-D) and 2) Adv-training (Adv-T).

Adaptive Attacks	No Defense		ANP		RNP (Ours)	
	CA	ASR	CA	ASR	CA	ASR
Adapt-D	91.56	84.76	89.78	24.81	**90.58**	**13.22**
Adv-T	88.56	73.36	83.06	54.01	**86.15**	**18.09**

in unlearning, the stronger RNP's ability to reduce the ASR for most attacks. However, the results are counter-intuitive for CA, especially for the Dynamic and CL attacks. Surprisingly, the CA drops as the size of the unlearning set grows. More specifically, the CA drops rapidly from nearly 90% to less than 30% as we increase the size of the defense set to 10%. We conjecture that this is because both CL and Dynamic cover the whole image with complex trigger patterns (possibly confusing with the clean feature representation), and thus the more clean data used in unlearning, the higher the chance for RNP to accidentally prune clean neurons. To summarize, our RNP achieves a higher CA with fewer data (i.e., 0.5% and 1%) for all these attacks, which is more practical for data-limited situations.

Impact of Unlearning Epochs. We are also interested in the impact of unlearning epochs on the performance of RNP. Therefore, we investigate the defense performances of RNP against 5 backdoor attacks (i.e., BadNets, SIG, CL, Dynamic, and FC) after the 5th, 10th, 15th, and 20th epochs of unlearning, respectively, and plot them in Figure 4. We find that unlearning the model for an excessive number of epochs hinders defense performance. More specifically, as we increase the unlearning epoch from the 10th to the 15th, the ASR does not drop further (in fact, it even becomes slightly higher), while the CA declines substantially for 4 of the 5 attacks. A possible explanation is that a gradient explosion occurs when maximizing the loss on the clean defense data, causing the unlearned model to collapse in later epochs (e.g., the 15th). This is why we terminate the unlearning immediately when the CA drops to a random guess (i.e., 10%).

4.3. Understanding the Mechanism of RNP

Asymmetric Unlearning-Recovering. We first show that asymmetric unlearning and recovering are key to successfully exposing backdoor neurons. We demonstrate this by investigating the deep features obtained via different combinations of neuron unlearning (NU), neuron recovering (NR), filter unlearning (FU), and filter recovering (FR).

As can be observed in the left subfigure of Figure 5, symmetric unlearning-recovering, i.e., NU-NR and FU-FR, can hardly affect the backdoor features. Neuron recovering in NU-**NR** and FU-**NR** can recover more clean features than filter recovering, as it offers more capacity to recover the neurons. Filter recovering (i.e., NU-**FR** and FU-**FR**), on the other hand, leads to darker feature maps (fewer activations) at the shallow layers (i.e., Block 2 and Block 3). This implies that filter recovering indeed limits the capacity for clean feature recovery.

Filter unlearning followed by a fine-grained neuron recovering (i.e., FU-NR) can mitigate the backdoor at the deep layers (Block 4) but not at the shallow layers (Block 2). The clean features are largely recovered, yet some backdoor features still exist. Only the neuron unlearning followed by a coarse-grained filter recovering (i.e., NU-FR) can effectively force the model to erase the backdoor features (and the associated backdoor neurons). The defense performance of the 4 strategies shown in the right subfigure of Figure 5 also confirms its effectiveness in locating and removing the backdoor neurons. In Appendix C.2, we also show that filter recovering can be effectively achieved at a few layers, e.g., layers before each batch normalization. However, it has to be applied across the network at both shallow and deep layers to achieve the best result.

Necessity of Unlearning and Recovering. Apart from the "asymmetric" working mechanism of RNP, we have also run 3 experiments (with CIFAR-10, ResNet-18, and 500 clean images as defense data) to show whether we need both neuron unlearning (maximization) and filter recovering (minimization) in Table 13 (Appendix C.1). The findings can be summarized as follows. 1) Unlearning without re-

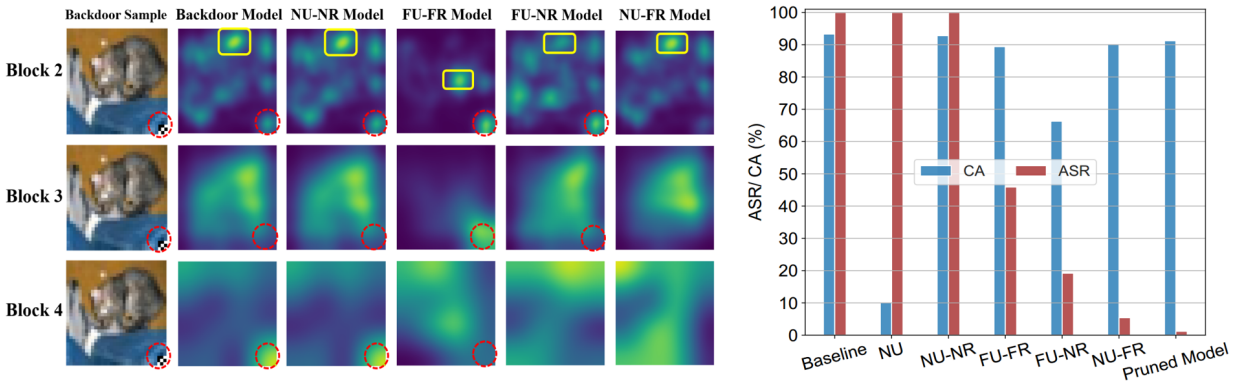

Figure 5. **Left:** The feature maps (channel-wise averaged output at the 2nd, 3rd, and 4th residual blocks of ResNet-18) of a backdoored image under 4 different unlearning-recovering strategies: NU-NR, FU-FR, FU-NR, and NU-FR. The squares and circles highlight the backdoored and clean regions, respectively. **Right:** The defense performance (ASR and CA) of the 4 unlearning-recovering strategies. These experiments were conducted with ResNet-18 and BadNets attack on CIFAR-10. **NU**: neuron unlearning; **FU**: filter unlearning; **NR**: neuron recovering; **FR**: filter recovering.

covering causes CA to drop significantly, which verifies the usefulness of the recovery step. 2) recovering without unlearning cannot decrease ASR, which proves that unlearning is a MUST. 3) Unlearning by learning incorrectly (i.e., minimizing towards a wrong target rather than maximizing) is notably worse than our RNP. The detailed discussion and analysis of the above findings are provided in Appendix C.

More Explorations with the Unlearned Model. Here, we demonstrate one important benefit of our RNP, that is, the unlearned model (NU-model) produced at its unlearning step can be leveraged to improve other backdoor defense tasks. First of all, the unlearned model directly exposes the backdoor label (class) since the functionality of the clean classes has been unlearned. When applying the trigger recovery and backdoor detection method Neural Cleanse (NC) (Wang et al., 2019) on the unlearned model (rather than the original model), one can expose the potential backdoor target more easily and improve the quality of the recovered triggers. When applying the backdoor sample detection method STRIP (Gao et al., 2019) on the unlearned model, one can detect backdoor samples that are notably more complex and stealthy. Moreover, the unlearned model can also help boost the defense performance of existing backdoor removal methods like Fine-Pruning (Liu et al., 2018a). More detailed analyses can be found in Appendix D.

5. Limitation

While our RNP achieves promising results against existing attacks, it does have several limitations: 1) It is not theoretically guaranteed to defend against all unforeseen attacks, thus having the risk of being compromised by more advanced attacks; 2) RNP faces a noticeable challenge when defending against backdoor attacks with low poisoning rates (e.g., the poisoning rate $\leq 1\%$), in which backdoor-related neurons/features are much easier to hide within clean neurons/features, leading to inaccurate pruning; 3) RNP fails to erase backdoors trained on lightweight model architectures like EfficientNet, sacrificing too much CA to reduce the ASR without fine-tuning. We will address these limitations of our RNP defense in our future work.

6. Conclusion

This paper proposes a novel and effective method called *Reconstructive Neuron Pruning* (RNP) to expose and prune the backdoor neurons from backdoored DNNs. At the core of RNP is an asymmetric unlearning-recovering scheme that first unlearns the neurons on a few clean samples via a *neuron-level unlearning*, and then recovers the neurons on the same clean samples via a *filter-level recovering*. We revealed the phenomenon that asymmetric unlearning-recovering from fine-grained unlearning to coarse-grained recovering can help expose backdoor neurons. We empirically demonstrated the effectiveness of RNP as a backdoor defense method and the benefit of the unlearning technique itself to other backdoor defense tasks, including trigger recovery, backdoor label detection, and backdoor sample detection. We hope our work could provide a new perspective for the community to develop more powerful backdoor defenses in the future.

Acknowledgements

This work is supported by China National Science Foundation under grant numbers 62072356 and 62276067.

References

Barni, M., Kallas, K., and Tondi, B. A new backdoor attack in cnns by training set corruption without label poisoning.

In *ICIP*, 2019.

Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I., and Srivastava, B. Detecting backdoor attacks on deep neural networks by activation clustering. In *AAAI Workshop*, 2019.

Chen, W., Wu, B., and Wang, H. Effective backdoor defense by exploiting sensitivity of poisoned samples. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *NeurIPS*, 2022.

Chen, X., Liu, C., Li, B., Lu, K., and Song, D. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

Cheng, S., Liu, Y., Ma, S., and Zhang, X. Deep feature space trojan attack of neural networks by controlled detoxification. In *AAAI*, 2021.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Doan, K., Lao, Y., Zhao, W., and Li, P. Lira: Learnable, imperceptible and robust backdoor attacks. In *ICCV*, 2021.

Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D. C., and Nepal, S. Strip: A defence against trojan attacks on deep neural networks. In *ACSAC*, 2019.

Garg, S., Kumar, A., Goel, V., and Liang, Y. Can adversarial weight perturbations inject neural backdoors. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020.

Gu, T., Dolan-Gavitt, B., and Garg, S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Guan, J., Tu, Z., He, R., and Tao, D. Few-shot backdoor defense using shapley estimation. In *CVPR*, 2022.

Guo, W., Wang, L., Xing, X., Du, M., and Song, D. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. *arXiv preprint arXiv:1908.01763*, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Hu, X., Lin, X., Cogswell, M., Yao, Y., Jha, S., and Chen, C. Trigger hunting with a topological prior for trojan detection. In *ICLR*, 2022.

Huang, K., Li, Y., Wu, B., Qin, Z., and Ren, K. Backdoor defense via decoupling the training process. In *ICLR*, 2022.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Li, Y., Zhai, T., Wu, B., Jiang, Y., Li, Z., and Xia, S. Rethinking the trigger of backdoor attack. *arXiv preprint arXiv:2004.04692*, 2020.

Li, Y., Li, Y., Wu, B., Li, L., He, R., and Lyu, S. Invisible backdoor attack with sample-specific triggers. In *ICCV*, pp. 16463–16472, 2021a.

Li, Y., Lyu, X., Koren, N., Lyu, L., Li, B., and Ma, X. Anti-backdoor learning: Training clean models on poisoned data. In *NeurIPS*, 2021b.

Li, Y., Lyu, X., Koren, N., Lyu, L., Li, B., and Ma, X. Neural attention distillation: Erasing backdoor triggers from deep neural networks. In *ICLR*, 2021c.

Liu, K., Dolan-Gavitt, B., and Garg, S. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *RAID*, 2018a.

Liu, Y., Ma, S., Aafer, Y., Lee, W.-C., Zhai, J., Wang, W., and Zhang, X. Trojaning attack on neural networks. In *NDSS*, 2018b.

Liu, Y., Lee, W.-C., Tao, G., Ma, S., Aafer, Y., and Zhang, X. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1265–1282, 2019.

Liu, Y., Ma, X., Bailey, J., and Lu, F. Reflection backdoor: A natural backdoor attack on deep neural networks. In *ECCV*, 2020.

Liu, Y., Shen, G., Tao, G., Wang, Z., Ma, S., and Zhang, X. Complex backdoor detection by symmetric feature differencing. In *CVPR*, 2022.

Nguyen, A. and Tran, A. Input-aware dynamic backdoor attack. In *NeurIPS*, 2020.

Nguyen, A. and Tran, A. Wanet–imperceptible warping-based backdoor attack. In *ICLR*, 2021.

Qi, X., Xie, T., Mahloujifar, S., and Mittal, P. Circumventing backdoor defenses that are based on latent separability. *arXiv preprint arXiv:2205.13613*, 2022a.

Qi, X., Xie, T., Pan, R., Zhu, J., Yang, Y., and Bu, K. Towards practical deployment-stage backdoor attack on deep neural networks. In *CVPR*, 2022b.

Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *NeurIPS*, 2018.

Shen, G., Liu, Y., Tao, G., An, S., Xu, Q., Cheng, S., Ma, S., and Zhang, X. Backdoor scanning for deep neural networks through k-arm optimization. In *ICML*, 2021.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *ICLR*, 2013.

Tang, D., Wang, X., Tang, H., and Zhang, K. Demon in the variant: Statistical analysis of dnns for robust backdoor contamination detection. In *USENIX Security*, 2021.

Tran, B., Li, J., and Madry, A. Spectral signatures in backdoor attacks. In *NeurIPS*, 2018.

Turner, A., Tsipras, D., and Madry, A. Clean-label backdoor attacks. *https://people.csail.mit.edu/madry/lab/*, 2019.

Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., and Zhao, B. Y. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *S&P*. IEEE, 2019.

Wang, T., Yao, Y., Xu, F., An, S., Tong, H., and Wang, T. An invisible black-box backdoor attack through frequency domain. In *ECCV*, 2022a.

Wang, Z., Ding, H., Zhai, J., and Ma, S. Training with more confidence: Mitigating injected and natural backdoors during training. In *NeurIPS*, 2022b.

Wu, D. and Wang, Y. Adversarial neuron pruning purifies backdoored deep models. *NeurIPS*, 2021.

Xu, X., Wang, Q., Li, H., Borisov, N., Gunter, C. A., and Li, B. Detecting ai trojans using meta neural analysis. In *S&P*, 2021.

Zeng, Y., Park, W., Mao, Z. M., and Jia, R. Rethinking the backdoor attacks' triggers: A frequency perspective. In *ICCV*, 2021.

Zeng, Y., Chen, S., Park, W., Mao, Z. M., Jin, M., and Jia, R. Adversarial unlearning of backdoors via implicit hypergradient. In *ICLR*, 2022.

Zhao, Z., Chen, X., Xuan, Y., Dong, Y., Wang, D., and Liang, K. Defeat: Deep hidden feature backdoor attacks by imperceptible perturbation and latent representation constraints. In *CVPR*, 2022.

Zheng, R., Tang, R., Li, J., and Liu, L. Data-free backdoor removal based on channel lipschitzness. In *ECCV*, 2022.

A. Implementation Details of RNP

A.1. Datasets and Classifiers

The datasets and DNN models used in our experiments are summarized in Table 5.

Table 5. Detailed information of the datasets and classifiers used in our experiments.

Dataset	Labels	Input Size	Training Images	Classifier
CIFAR-10	10	32 x 32 x 3	50000	ResNet-18/ VGG-16/ MobileNet-V2
ImageNet subset	12	224 x 224 x 3	12406	ResNet-34

A.2. Attack Details

We mainly considered 12 state-of-the-art backdoor attacks, including 7 input-space attacks: BadNets (Gu et al., 2017), Trojan (Liu et al., 2018b), Blend (Chen et al., 2017), Dynamic (Nguyen & Tran, 2020), WaNet (Nguyen & Tran, 2021), SIG (Barni et al., 2019), and CL (Turner et al., 2019), and 3 feature-space attacks: FC (Shafahi et al., 2018), DFST (Cheng et al., 2021), and AWP (Garg et al., 2020). In addition, we evaluated two recently proposed adaptive attacks termed LIRA (Doan et al., 2021) and Adaptive-Blend (A-Blend) (Qi et al., 2022a).

Figure 6 shows a few examples of the backdoor triggers used in our experiments. All attacks were mainly trained on CIFAR-10 (Krizhevsky et al., 2009) to attack the ResNet-18 model (He et al., 2016) or an ImageNet-12 (Deng et al., 2009) subset to attack ResNet-34. We trained all models for 200 epochs using Stochastic Gradient Descent (SGD) with an initial learning rate of 0.1, a batch size of 128, and a weight decay of 5e-4 to obtain the backdoored models. The learning rate was divided by 10 at the 60th and 120th epochs. We used two standard data augmentation techniques (horizontal flip and random crop with padding 4×4) during model training. We followed the default settings suggested in the original papers and the open-source codes for most attacks; this included the trigger pattern, trigger size, and backdoor label. We also tuned the hyperparameters of several attacks that were negatively affected by the two data augmentations to obtain the best attack performance. We carefully altered the hyperparameter configurations for several feature-space attacks to ensure that they achieved the best attack performances. The backdoor label of all attacks was set to class 0. We also evaluated the defense performance of our RNP on an ImageNet-12 subset. Following previous work (Li et al., 2021b), we reproduced 5 attacks on ImageNet-12: BadNets, Blend, Trojan, SIG, and FC. We omitted the other attacks due to failed reproductions. Table 6 summarizes the detailed settings of these attacks.

Table 6. Attack settings of 12 backdoor attacks. ASR (%): attack success rate; CA (%): clean accuracy.

Attacks	Trigger Type	Trigger Pattern	Target Label	Poisoning Rate
BadNets	Fixed	Grid	0	0.1
Trojan	Fixed	Reversed Watermark	0	0.1
Blend	Fixed	Random Pixel	0	0.1
Dynamic	Varied	Mask Generator	0	0.1
SIG	Fixed	Sinusoidal Signal	0	0.08
CL	Fixed	Grid and PGD Noise	0	0.08
FC	Varied	Optimization-based	source 1, target 0	0.08
DFST	Varied	Style Generator	0	0.1
WaNet	Varied	Optimization-based	0	0.1
AWP	Fixed	Weight Perturbation	0	0.1
LIRA	Varied	Optimization-based	0	0.1
A-Blend	Fixed	Mixer Construction	0	0.1

A.3. Defense Details

We experimented with 8 backdoor defenses in total, including 2 backdoor detection methods: Neural Cleanse (NC) (Wang et al., 2019) and STRIP (Gao et al., 2019), and 5 backdoor removal methods: Fine-pruning (FP) (Liu et al., 2018a), Neural

Figure 6. Examples of 12 backdoor trigger patterns on CIFAR-10.

Attention Distillation (NAD) (Li et al., 2021c), Adversarial Unlearning of Backdoors via Implicit Hypergradient (I-BAU) (Zeng et al., 2022), Adversarial Neuron Perturbation (ANP) (Wu & Wang, 2021), Anti-backdoor Learning (ABL) (Li et al., 2021b), and our RNP. All defenses had limited access to only 500 defense data from the CIFAR-10 training set (or ImageNet-12).

We used the open-source PyTorch code for NC [1] to reproduce the results of backdoor detection and trigger recovery. To combine our NU with the existing method NC (i.e., NU+NC), we replaced only the original model f used by NC with our unlearned model $f_{\hat{\theta}}$ and kept other settings unchanged. For STRIP, we calculated the relative entropy between the backdoored model's output distributions on clean vs. backdoored examples. We then compared the difference in relative entropies between the original backdoored model and the unlearned backdoored model $\hat{\theta}_{NU}$.

We reimplemented FP using PyTorch and pruned the last convolutional layer (i.e., Layer4.conv2) of the model until the CA of the network became lower than 80%. For NAD, we adopted the same settings used in the open-sourced code [2] and cautiously selected the best hyper-parameter β from $[0, 5000]$ with an interval of 500. For I-BAU, we followed the settings used in the open-sourced code [3] to present the best defense results. We used the open-source code for ANP [4], and followed the suggested settings with the perturbation budget $\epsilon = 0.4$ and the trade-off coefficient $\alpha = 0.2$ to optimize the mask. We also combined our NU with NC to recover the trigger patterns and then erased the triggers from the backdoored model using the unlearning technique used in ABL.

For our RNP defense, we maximized the unlearned model $f_{\hat{\theta}}$ for 20 epochs until its clean accuracy dropped to 10% (random guess) with a learning rate of 0.01, a batch size of 128, and a weight decay of 5e-2. For the recovering step, we optimized the filter mask for 20 epochs with a learning rate of 0.2. In comparison to pruning by fraction, we found that pruning the neurons by a dynamic threshold often yielded better performance, and adopting a threshold within $[0.4, 0.7]$ consistently produced the best results of RNP (low ASR and high CA) against all backdoor attacks. Note that ANP also suggested the dynamic threshold strategy. The neurons were pruned based on the learned mask using the dynamic thresholding strategy for an accuracy drop $\sim 2\%$. See Table 8 and Table 3 for more detailed analyses of different pruning strategies and pruning rates.

All defense methods were trained using the same data augmentation techniques, i.e., random crop ($padding = 4$) and horizontal flipping as mentioned in the attack settings.

[1] https://github.com/VinAIResearch/input-aware-backdoor-attack-release/tree/master/defenses/neural_cleanse
[2] https://github.com/bboylyg/NAD
[3] https://github.com/YiZeng623/I-BAU
[4] https://github.com/csdongxian/ANP_backdoor

Table 7. Comparison between ANP and our RNP against 6 backdoor attacks on GTSRB dataset. Note that only 500 clean samples are used as defense data.

Backdoor Attacks	No Defense		ANP		RNP (Ours)	
	ASR	CA	ASR	CA	ASR	CA
BadNets	100	96.83	3.59	95.37	0.16	95.49
Blend	100	96.71	2.59	94.57	1.64	95.54
Trojan	99.97	96.14	5.73	94.27	6.96	95.35
CL	68.65	95.53	5.8	91.32	0	94.24
SIG	85.13	96.33	0	22.89	5.74	94.91
Dynamic	100	97.11	7.27	96.58	6.11	96.65

B. Additional Experimental Results of RNP

B.1. Comparison between RNP and ANP on the GTSRB Dataset

We conducted a set of new experiments on the suggested GTSRB dataset. Table 7 below reports the performance of ANP and our RNP against 6 typical attacks (BadNets, Trojan, Blend, CL, SIG, Dynamic). The results indicate that our RNP method generalizes well and consistently outperforms ANP on this new dataset against the 6 attacks.

B.2. RNP with Different Pruning Strategies

Existing work (Wu & Wang, 2021) implemented neuron pruning with two different strategies: threshold or fraction. The threshold-based strategy prunes the neurons with a mask value smaller than the pre-specified threshold (by setting the neuron weight to 0), while the fraction-based strategy prunes a certain proportion of the neurons via the mask. To verify their effectiveness, we trained BadNets models using ResNet-18 with similar settings as described in Section 4. For a fair comparison, we reported the performance metrics such as ASR, CA, and the number of pruned neurons in Table 8. We found that the threshold-based strategy achieves better results on both ASR and CA than fraction-based pruning (ASR: 92.22 VS. 91.82, CA: 0.20 VS. 0.47). Interestingly, the superiority of the threshold-based strategy is also reflected by the number of pruned neurons (Neurons: 41 vs. Neurons: 48); fewer pruned neurons result in less effect on the CA. This finding provides another piece of evidence of how effective and precise our RNP can expose the backdoor neurons. As a result, we set the threshold-based strategy as the default setting throughout all experiments unless otherwise specified.

Table 8. Comparison to the pruning strategy by *dynamic threshold* and *by fraction* against BadNets on ResNet-18, respectively. $Neurons \downarrow$ indicates the number of neurons pruned.

By Threshold	0	0.2	0.5	0.6	0.65	0.7	0.75	0.8
$Neurons \downarrow$	0	0	5	11	23	**41**	75	178
ASR (%)	100	100	100	95.50	81	**0.2**	0	0
CA (%)	93.4	93.4	93.39	92.84	92.8	**92.22**	91.17	86.92
By Fraction (%)	0	1	2	3	4	5	8	10
$Neurons \downarrow$	0	**48**	93	144	192	240	384	432
ASR (%)	100	**0.47**	0.16	0	0	0	0	0
CA (%)	93.4	**91.82**	89.81	89.02	85.71	81.68	72.20	64.93

B.3. RNP with Different Model Architectures

We also evaluated our RNP with additional model architectures, including ResNet-18, VGG-19, MobileNet-V2, and EfficientNet-B0. Table 9 reports the performance of ANP and our RNP against 3 typical attacks (BadNets, Trojan, Blend) on the CIFAR-10 dataset. The results indicate that our RNP generalizes well across most different model architectures (i.e., ResNet-18, VGG-19, and MobileNet-V2) but faces challenges with the EfficientNet architecture.

Specifically, we found that RNP achieves better defense performance than ANP on ResNet-18, VGG-19, and MobileNet-V2 while facing the same challenge as ANP on EfficientNet-B0, where the accuracy of the pruned model dropped significantly. We speculate that this result may be related to the design of lightweight model structures. The sparse neurons in lightweight

model structures limit the asymmetric unlearning-recovering and make it difficult to locate the backdoor neurons accurately, thereby leading to decreased performance of RNP. We will investigate and address the performance degradation of RNP on such lightweight model structures in our future research.

Table 9. Defense performance (ASR and CA) of ANP and our RNP with different model architectures on the CIFAR-10 dataset. Note that only 500 images are used for defense.

Model Architectures	Metric	ResNet-18			Vgg-19			MobileNet-V2			EfficienNet-B0		
		No Defense	ANP	RNP (Ours)	No Defense	ANP	RNP (Ours)	No Defense	ANP	RNP (Ours)	No Defense	ANP	RNP (Ours)
BadNets	ASR	100	0.53	0.20	100	3.31	0.38	100	5.92	3.17	100	3.12	2.79
	CA	93.40	91.61	92.22	92.63	92.37	92.57	92.89	90.48	91.67	76.74	34.31	65.62
Trojan	ASR	99.90	1.00	2.23	99.98	2.60	2.49	100	2.03	2.96	100	3.63	1.18
	CA	93.15	92.37	92.56	92.67	92.44	92.31	93.01	90.91	90.21	76.48	28.83	68.17
Blend	ASR	100	0.50	0.33	100	2.93	2.72	100	1.79	1.21	100	6.32	4.38
	CA	93.10	92.31	92.62	92.60	92.17	92.06	92.80	86.71	90.26	75.52	21.43	63.77

Table 10. Comparison between ANP and our RNP against 3 all-to-all backdoor attacks on CIFAR-10. Note that only 500 clean samples are used as defense data.

Backdoor Attacks	No Defense		ANP		RNP (Ours)	
	ASR	CA	ASR	CA	ASR	CA
BadNets (all-to-all)	92.18	92.65	6.16	88.58	**1.24**	**92.56**
Trojan (all-to-all)	91.98	92.57	18.84	88.14	**8.78**	**90.67**
Blend (all-to-all)	84.23	92.38	13.81	86.26	**11.56**	**90.02**

Table 11. Defense performance of our RNP against BadNets under various trigger sizes on CIFAR-10. Note that only 500 clean samples are used as the defense data.

Backdoor Attacks	No Defense		ANP		RNP (Ours)	
	ASR	CA	ASR	CA	ASR	CA
3×3	100	93.40	0.53	91.61	**0.20**	**92.22**
5×5	100	92.94	**0.74**	89.72	1.44	**91.38**
10×10	100	91.78	9.08	88.53	**1.76**	**91.66**

B.4. RNP against All-to-All Attacks

We reproduced 3 typical all-to-all (i.e., target_label $=$ original_label $+1$) attacks with ResNet-18 on the CIFAR-10 dataset.

Here, we report the defense results for both our RNP and ANP. For ANP, we directly used its open-sourced code and selected its best defense results for comparison. The results, presented in Table 10, show that ANP achieves good defense performance in terms of both ASR and CA. However, our RNP method still outperformed ANP with noticeable margins. For example, the CA of our RNP is all above 90% while ANP is around 86% to 88%. The ASR reduction of our method is $\sim 5\%$, $\sim 10\%$, and $\sim 2\%$ better than ANP against BadNets, Trojan, and Blend attacks, respectively.

B.5. RNP against Different Trigger Sizes

The performance of our RNP against BadNets under different trigger sizes is reported in Table 11. It shows that our RNP can defend against the attack with different trigger sizes from 3×3 to 10×10. Note that a 10×10 trigger appears rather large and obvious on 32×32 CIFAR-10 images. Increasing the trigger size does bring slightly more resistance to our defense but is very limited. Please note that, except for BadNets, Trojan, and AWP attacks, the triggers of other attacks span the entire image, that is, they have a fixed trigger size – the image/input size.

B.6. RNP against Different Poisoning Rate

We evaluated RNP's performance against the BadNets attack on CIFAR-10 with diverse poisoning rates. From Table 12, we observed that both ANP and RNP inevitably encounter clean accuracy degradation as the poisoning rate decreases. When

Table 12. Defense performance of ANP and our RNP against BadNets attack on CIFAR-10 with diverse poisoning rates. Note that only 500 images are used for defense.

CIFAR-10: BadNets	No Defense		ANP		RNP		RNP + Finetuning	
	ASR	CA	ASR	CA	ASR	CA	ASR	CA
10% (5000)	100	93.4	0.53	91.61	0.2	92.22	3.78	92.64
5% (2500)	100	92.59	1.63	78.83	0.32	90.26	2.51	90.18
1% (500)	99.96	92.67	75.8	59.42	5.29	84.8	4.45	86.59
0.05% (250)	99.18	92.36	0	14.42	0	56.32	0.46	77.29

the poisoning rate is 0.05%, the clean accuracy of ANP drops to 14.42%, while our RNP drops to 56.32% but improves to 77.29% when fine-tuning is applied. We conjecture that this is because the low poisoning rate hides the backdoor neurons more stealthily among the benign neurons, making pruning more challenging. However, low poisoning rates pose a challenge for most current state-of-the-art backdoor defenses, including backdoor sample detection (e.g., STRIP (Gao et al., 2019)), backdoor model detection (e.g., Neural Cleanse (Wang et al., 2019)), backdoor model/neuron removal (e.g., ANP (Wu & Wang, 2021)), and robust backdoor learning (e.g., ABL (Li et al., 2021b)). Effective solutions to such a challenge deserve further investigation.

Table 13. Defense results of 1) pruning without filter recovering; 2) filter recovering with/without neuron unlearning; and 3) unlearning by learning incorrectly. The experiments are implemented on CIFAR-10 with ResNet-18, and 500 clean images as the defense data.

Pruning Results	No Defense		Pruning w/o Recovering		Filter Recovering w/o Unlearning		Filter Recovering w/ Unlearning (RNP)		Learning Incorrectly	
	CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR
BadNets	93.40	100	40.5	0	93.39	100	92.22	0.20	80.58	3.64
SIG	94.59	90.86	29.49	0.38	94.57	90.61	94.62	0.43	93.78	0.73
CL	94.84	100	11.82	0.44	93.84	100	91.92	8.87	71.12	4.00
Dynamic	91.36	99.97	31.2	0	91.26	99.85	90.18	15.24	83.33	76.06
FC	94.67	100	18.23	7.52	94.41	100	90.93	1.80	80.23	17.33

C. More Understandings of RNP

C.1. Necessity of Neural Unlearning and Filter Recovering

In addition to "asymmetric unlearning-recovering," we have conducted 3 new experiments (with CIFAR-10, ResNet-18, and 500 clean images as defense data) to demonstrate the key element of *neural unlearning and filter recovering*. The conclusions have also been carefully checked against other attacks. The settings and findings of these experiments are summarized below, with the results reported in Table 13.

a) Pruning without filter recovering, where the backdoored neurons are directly pruned from the unlearned model. This is to validate the usefulness of the recovering step.

Finding: The backdoor can be reliably erased from the model (most of the ASR is close to 0), but the clean accuracy (CA) is below 41%. The low ASR indicates that the backdoor neurons can also be removed even without recovering, but this could also remove a certain amount of clean neurons (CA drops significantly). This can be explained by the visualizations (Unlearned model vs. Recovered model) in Figure 2 in our main paper. Without the recovering, the key clean neurons (in the yellow rectangles) will be largely damaged, and some clean neurons outside the rectangles will be changed too slightly. This will cause two negative effects: 1) the clean accuracy is hard to recover (caused by damaged clean neurons), and 2) some clean neurons may accidentally get removed. This also verifies the importance of the recovering step (a comparison of neuron recovering vs. filter recovering has already been analyzed in Figure 5). Please note that, without the recovering, the neurons that DO NOT change much during the unlearning process should be removed (as only the clean neurons will be unlearned). However, with recovering, the neurons that change the MOST should be removed (as backdoor neurons are repurposed).

b) Filter recovering with/without neuron unlearning, which is to show that neuron unlearning is a MUST.

Table 14. Filter recovering at different residual blocks of ResNet-18. $Neurons \downarrow$ indicates the number of neurons pruned at the corresponding block of ResNet-18 against BadNets attack.

Mask Location	Baseline		RNP		$Neurons \downarrow$
	ASR (%)	CA (%)	ASR (%)	CA (%)	-
Block 1	100	93.40	9.98	73.37	56
Block 2	100	93.40	9.40	88.60	67
Block 3	100	93.40	9.01	93.10	62
Block 4	100	93.40	6.50	93.61	55
All Block	100	93.40	**0.20**	**92.22**	**41**

Finding: The result shows that filter recovering alone cannot decrease the ASR (over 90%) although the clean accuracy is as good as the original model. This is somewhat unsurprising, as filter recovering without unlearning is the same as directly fine-tuning the model on the defense data.

c) Unlearning by learning incorrectly, where the backdoored model is fine-tuned on incorrectly labeled clean samples (the labels are set to be their predicted labels of the RNP-unlearned model). That is, this model optimizes the same objective as our RNP but via a process of minimization (learning incorrectly) rather than maximization (unlearning).

Finding: This defense is notably worse than our RNP, which indicates that minimization and maximization are quite different in terms of their impact on the neurons. Maximization (unlearning) can be understood from the perspective of adversarial perturbation applied to the neurons, i.e., it updates the most influential neurons to maximize the model's error (as it is the quickest way to reduce the loss). Minimization, on the other hand, tends to update all neurons (even repurpose some of the neurons, as we have shown in Figure 5) to reduce the classification loss. We believe this finding itself is interesting even out of the scope of backdoor defense.

C.2. How Does Filter Location Impact RNP?

Table 14 shows how the location of filter recovering affects the defense performance of our RNP. We restrict the filter mask to be applied before the last batch normalization layer at different residual blocks of the network and report the defense performance after pruning. It shows that the defense is more effective if the mask is applied at deeper layers (Block 4 vs. Block 1), in terms of both ASR and CA. This indicates that the attack strength accumulates from shallow to deep layers and becomes more malicious in the deep layers (pointing to the backdoor class). It is worth mentioning that, when only recovering and pruning Block 4, the clean accuracy is largely preserved, or even slightly improved: 93.61% vs. 93.40% (in Table 14). This implies that sweeping the deep layers may be a good option if clean accuracy is the primary concern in real-world applications. RNP becomes most effective when filter recovering is applied across the entire network (All Block), which is also the default setting of our experiments. It achieves the best ASR performance, but the CA is reduced by approximately 2%. Therefore, we recommend this strategy if security is the primary concern in real-world applications.

D. Exploration with the Unlearned Model

Here, we will show that the unlearned model (NU-model) produced by our NU technique can be leveraged not only to improve existing defense methods but also to assist with trigger recovery, backdoor label detection, and backdoor sample detection.

D.1. Improving Backdoor Removal

Previous work proposed fine-tuning on the recovered trigger (denoted as 'NC+FT') (Wang et al., 2019), trigger unlearning (denoted as 'NC+ABL') (Li et al., 2021b), and fine-pruning (denoted as 'FP') (Liu et al., 2018a) to repair the backdoored model. To further explore our NU technique, we propose to perform backdoor removal on the NU-unlearned model. We set NC+FT as the baseline. The experiment results in Table 15 show that using the unlearned model can significantly improve the performance of the three defenses against all attacks. More specifically, in comparison to the baseline (i.e., NC+FT), adding NU can further lower the average ASR for NC+FT (row 2), NC+ABL (row 3), and FP (row 4) by 4.18%, 3.78%, and 6.01%, respectively. Meanwhile, NU also boosts clean accuracy to a certain extent. These results verify that NU can indeed be a central part of effective backdoor removal.

Table 15. Performance of 3 defense methods with/without our proposed NU respectively against backdoor attacks on CIFAR-10. ASR: attack success rate (%); CA: clean accuracy (%); *AvgDev*: the average % deviation in ASR/CA compared to the baseline 'NC+FT'.

Metric	Defense	BadNets	Trojan	Blend	CL	SIG	Dynamic	WaNet	FC	DFST	AWP	AvgDev
ASR	NC+FT (Baseline)	8.31	1.64	3.71	3.51	5.62	9.71	8.19	43.69	12.50	3.20	-
	NU+NC+FT	0.72	0.78	0.24	0.58	1.18	8.67	1.10	34.58	9.60	0.87	↓ **4.18**
	NU+NC+ABL	0.28	1.02	1.87	0.24	3.37	46.86	7.79	0	0.9	0.01	↓ **3.78**
	NU+FP	0.37	1.42	0	6.48	0.04	11.16	10.19	4.91	4.89	0.56	↓ **6.01**
CA	NC+FT (Baseline)	93.21	92.96	92.63	92.28	82.06	91.28	83.32	87.62	87.93	94.16	-
	NU+NC+FT	92.13	93.14	92.64	83.79	80.36	91.08	88.32	92.76	88.93	94.70	↑ **0.03**
	NU+NC+ABL	93.92	93.28	91.32	93.23	88.60	81.19	93.30	85.06	93.22	94.36	↑ **1.00**
	NU+FP	89.94	85.30	92	92.57	93.88	86.84	92.23	91.87	92.58	94.53	↑ **1.42**

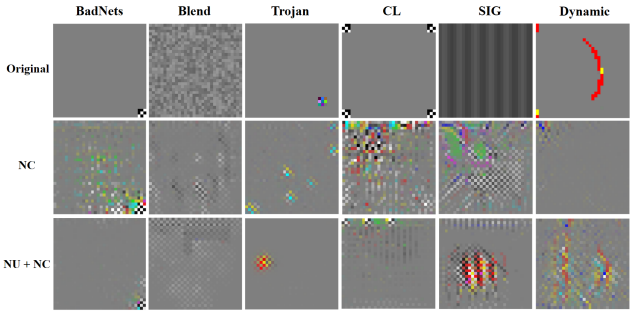

Figure 7. Side-by-side comparison of the original trigger patterns and their recovered versions by NC on the backdoored models and by our NU (NU+NC) on the unlearned models.

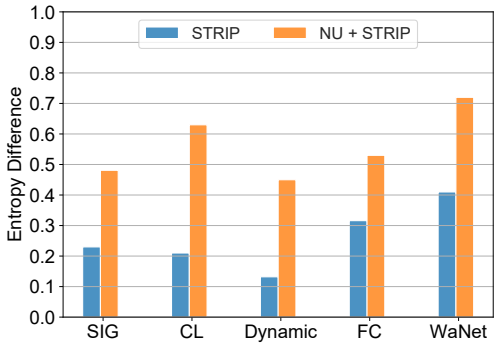

Figure 8. Overall entropy difference between clean and backdoor examples. A greater difference in entropy indicates a better detection performance.

Table 16. Detection accuracy (top-1, %) of the backdoor label.

Types	BadNets	Trojan	Blend	CL	SIG	Dynamic	WaNet	FC	DFST	AWP
NC	65	95	90	90	15	80	85	0	20	95
NU+NC	100	100	100	100	100	100	100	90	95	100
NU	100	100	100	100	100	100	100	100	100	100

D.2. Improving Trigger Recovery

Figure 7 shows a few examples of recovered triggers. We observe that, compared to the NC alone (second row), the triggers recovered by the NU-unlearned model present a more precise trigger quality and a reasonable size for almost all attacks. NC alone fails to recover the trigger on CL, SIG, and Dynamic, due to the recovered trigger having more image noise. We speculate that the success of 'NU+NC' is because the unlearned model contains fewer clean but more backdoor neurons than the original backdoored model. In other words, the unlearned model makes trigger reverse engineering easier.

D.3. Improving Backdoor Label Detection

Table 16 shows the NU-unlearned model can efficiently boost the performance of backdoor label detection for Neural Cleanse (NC). Interestingly, NU alone achieves the best detection rate by 100% on all the attacks. This is because, as clean neurons are gradually removed from the model, the backdoor class naturally arises. As a result, the samples are predicted by the unlearned model to be of the backdoor class.

D.4. Improving Backdoor Sample Detection

Figure 8 shows the average relative entropy between the outputs of the clean and the backdoored examples by the original STRIP and our NU combined with STRIP (denoted as NU+STRIP). We find that NU+STRIP creates a greater relative entropy (nearly 0.2) than the ordinary STRIP. The greater the relative entropy, the better the detection performance on filtering poisoned examples. Our NU amplifies the gap between clean and backdoor outputs in relative entropy, consequently contributing to detecting more advanced triggers.