HÖLDER PRUNING: LOCALIZED PRUNING FOR BACKDOOR REMOVAL IN DEEP NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

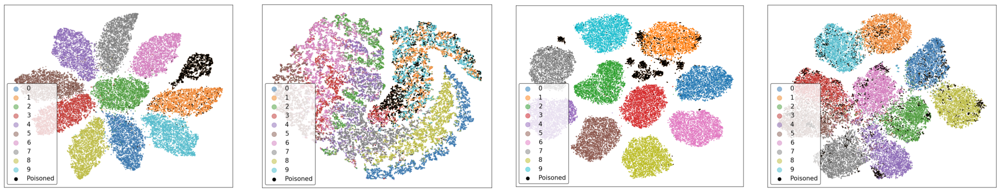
ABSTRACT

Deep Neural Networks (DNNs) have become the cornerstone of modern machine learning applications, achieving impressive results in domains ranging from computer vision to autonomous systems. However, their dependence on extensive data and computational resources exposes them to vulnerabilities such as backdoor attacks, where poisoned samples can lead to erroneous model outputs. To counter these threats, we introduce a defense strategy called *Hölder Pruning* to detect and eliminate neurons affected by triggers embedded in poisoned samples. Our method partitions the neural network into two stages: feature extraction and feature processing, aiming to detect and remove backdoored neurons—the highly sensitive neurons affected by the embedded triggers—while maintaining model performance This improves model sensitivity to perturbations and enhances pruning precision by exploiting the unique clustering properties of poisoned samples. We use the Hölder constant to quantify sensitivity of neurons to input perturbations and prove that using the Fast Gradient Sign Method (FGSM) can effectively identify highly sensitive backdoored neurons. Our extensive experiments demonstrate efficacy of *Hölder Pruning* across six clean feature extractors (SimCLR, Pretrained ResNet-18, ViT, ALIGN, CLIP, and BLIP-2) and confirm robustness against nine backdoor attacks (BadNets, LC, SIG, LF, WaNet, Input-Aware, SSBA, Trojan, BppAttack) using three datasets (CIFAR-10, CIFAR-100, GTSRB). We compare *Hölder Pruning* to eight SOTA backdoor defenses (FP, ANP, CLP, FMP, ABL, DBD, D-ST) and show that *Hölder Pruning* outperforms all eight SOTA methods. Moreover, Hölder Pruning achieves a runtime up to 1000x faster than SOTA defenses when a clean feature extractor is available. Even when clean feature extractors are not available, our method is up to 10x faster.

1 INTRODUCTION

Deep neural networks (DNNs) have demonstrated outstanding performance across a range of applications, including computer vision (He et al., 2015), speech recognition (Gulati et al., 2020), and recommendation systems (Wang et al., 2023). Specifically, their framework mainly consists of a feature extractor (Convolutional neural networks (CNNs) (He et al., 2015), Transformers (Vaswani et al., 2023)) and a classifier. DNN training requires extensive data and computational resources, often involving third-party data or servers. This dependency raises significant security concerns, especially when using large public datasets that may contain corrupted or poisoned data samples. Such malicious data can lead the DNN to produce undesired outputs (Gu et al., 2019). Among various data corruption methods, backdoor attacks pose a notable threat (Li et al., 2022). This attack involves deliberate manipulation by either controlling the training process (Ilyas et al., 2018; Kurakin et al., 2017; Li et al., 2021a; Liang et al., 2020), or corrupting a small number of data samples by inserting a predefined perturbation called *trigger* (Gu et al., 2019). Consequently, the trained DNN model behaves normally on clean (non-corrupted) data, but produces an (adversary-desired) output label for poisoned (corrupted) samples. This manipulation can hinder the detection and mitigation of corrupted data, making the development of effective defense mechanisms a priority in neural network research.

To address these challenges, several novel backdoor defense methods, including pruning, have been proposed (Chen et al., 2022; Guo et al., 2022; Li et al., 2021b; Wu & Wang, 2021; Zheng et al., 2022). Pruning methods (Wu & Wang, 2021; Zheng et al., 2022) attempt to prevent backdoor attacks by eliminating neurons that process trigger features. Corrupted samples processed by poisoned neurons,



(a) (b) (c) (d)

Figure 1: t-SNE visualizations illustrating that a backdoored model with a pre-trained clean feature extractor independently obtained from a third-party (e.g., HuggingFace (Jain, 2022)) performs well under Fine-Pruning (Liu et al., 2018a) using CIFAR-10 dataset with the following cases: (a) backdoored model with poisoned feature extractor and rest of model trained concurrently on data that includes poisoned samples; (b) result of Fine-Pruning (Liu et al., 2018a) on model in (a); (c) backdoor model equipped with clean feature extractor; (d) result of Fine-Pruning (Liu et al., 2018a) on model in (c).

even if containing features from different classes, tend to form distinct cluster(s) at outputs of the penultimate layer in backdoored models (Chen et al., 2018; Hayase et al., 2021). Fine-Pruning (Liu et al., 2018a)—a classical pruning defense without a clean feature extractor (see Fig. 1(b)) results in blurred boundaries between different classes after pruning. In contrast, a pre-trained clean feature extractor (Fig. 1(c)) maintains clear boundaries for each class and restores poisoned data to their original clustering state under Fine-Pruning, as shown in Fig. 1(d), enhancing the robustness of model output to perturbations. However, in the process of removing these poisoned neurons, pruning methods can fail to precisely differentiate between poisoned and clean neurons due to the subtle nature of trigger features and their overlap with clean features. This can lead to unintentional deletion of clean neurons and a subsequent decline in model performance (Huang & Bu, 2024; Liu et al., 2018a). To overcome this limitation, we investigate utilization of clean feature extractors to improve pruning while preventing performance degradation.

Fig. 2 (a) shows a schematic consisting of a CNN feature extractor and a classifier, where the classifier has benign (blue) and toxic (red) neurons. Fig. 2 (b) presents a pre-trained clean feature extractor (e.g., independently obtained from a thirdparty such as HuggingFace (Jain, 2022)) and a classifier. Unlike traditional CNNs, using a clean feature extractor forces backdoored neurons to be contained in only layers of the classifier. In this setup, such confined poisoned neurons must adapt to fit backdoor trigger features to achieve high attack success (Wang et al., 2020). This adaptation makes such neurons overfitted to trigger features and highly sensitive to input perturbations (Jin et al., 2022).

To quantify sensitivity of (poisoned) neurons, recent research (Zheng et al., 2022) has proposed use of the Lipschitz constant. The Lipschitz constant measures neuron sensitivity across the whole training space. While such an approach provides a ‘global’ measure of sensitivity of output to an input (Lera & Sergeyev, 2010; Kalton, 2004), it is not sufficiently sensitive to small changes in neurons within localized regions. A measure that can effectively capture local neuron sensitivity is required to improve identification and removal of poisoned neurons. We propose the Hölder constant (Knill, 1994) as a metric for this purpose; a large value of this constant indicates high sensitivity to input perturbations. If each training sample, along with its perturbation, represents a local region in Hölder space, then we expect that poisoned neurons will have high Hölder constants in at least one local region. Consequently, removing neurons with Hölder constant higher than a threshold will make the model more insensitive to poisoned samples.

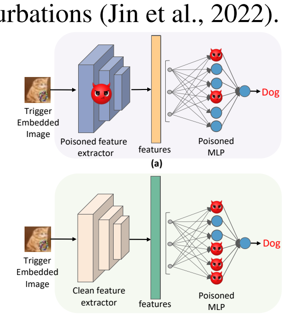


Figure 2: Schematic showing (a) CNN feature extractor and classifier (MLP); (b) pretrained clean feature extractor and classifier.

Based on this observation, we introduce a new defense strategy that we term **Hölder Pruning**. Hölder pruning employs a clean feature extractor in a classifier to concentrate and enhance features of poisoned neurons. Such a clean feature extractor can be pre-trained (He et al., 2015), transformer-based (Vaswani et al., 2023), or obtained through self-supervised learning (Chen et al., 2020). Our experiments demonstrate that using a clean feature extractor increases the sensitivity of poisoned neurons, and the Hölder constant provides a quantifiable interpretation of sensitivity. In practice, large organizations (Mengara et al., 2024) utilize proprietary feature extractors, which are presumed to be clean. However, when adapting to varied business requirements, these companies might need to fine-tune their models using additional data. In such scenarios, employing contaminated data risks producing a compromised model. **Hölder Pruning** effectively integrates these feature extractors without compromising model performance while also achieving robust defense against backdoor attacks. When a clean feature extractor is unavailable, or when it is inevitable to rely on untrustworthy training data to construct a secure model, we design the **Hölder Iteration Defense**. The Hölder Iteration Defense first uses self-supervised learning to obtain a clean feature extractor and then uses Hölder Pruning to reliably obtain clean samples. We show that an iterative application of these two steps simultaneously achieves high classification accuracy, robust accuracy, and low attack success rates.

Contributions: The main contributions of this paper are: **(a)** we discover that concentrating trigger features on the classifier using a clean feature extractor can significantly enhance effectiveness of pruning methods; **(b)** We introduce a novel pruning technique, Hölder Pruning, adaptable across different types of feature extractors. This method leverages Hölder constants to measure neuron sensitivity to perturbations. Furthermore, we formally prove the efficacy of applying the Fast Gradient Sign Method (FGSM) at the neuron level for identifying the maximum sensitivity of each neuron using the Hölder constant. **(c)** when a clean feature extractor is unavailable, we propose the Hölder Iteration Defense (HID) and show that the HID can obtain a clean model from a poisoned dataset; **(d)** we perform extensive experiments to show that Hölder Pruning seamlessly integrates with six clean feature extractors; Hölder Pruning and the Hölder Iteration Defense are robust to nine backdoor attacks while maintaining accuracy and efficiency, and outperforms eight SOTA backdoor defenses; **(e)** we show that Hölder Pruning runs up to 1000x faster than SOTA defenses when using a clean feature extractor, while Hölder Iteration Defense offers a speedup of up to 10x compared to similar methods when a clean feature extractor is not available. To the best of our knowledge, our Hölder Pruning strategy is the first known in-training defense against backdoor attacks.

2 PRELIMINARIES

This section introduces necessary preliminaries on DNNs and backdoor attacks, defines the Hölder condition that informs our neuron pruning-based defense strategy, and presents performance metrics used to evaluate our defense. Finally, we describe the threat and defense models considered.

2.1 DNNS AND BACKDOOR ATTACKS

In the context of supervised learning (Goodfellow et al., 2016), we examine a DNN represented as $f_\theta : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ denotes the input space, $\mathcal{Y}$ is the set of class labels, and θ denotes the model's parameters. The training dataset is denoted $D = \{(x_i, y_i)\}_{i=1}^n$, where $\{x_i\}_{i=1}^n \subset \mathcal{X}$ and $y_i \in \mathcal{Y}$. We consider *backdoor attacks* (Gu et al., 2019; Barni et al., 2019; Nguyen & Tran, 2021), in which an adversary subtly modifies (poisons) a small subset of training dataset by inserting a trigger Δ into a subset of training inputs $\{x_i\}_{i=1}^p \subset \{x_i\}_{i=1}^n$, where $p \ll n$, and altering their corresponding labels $\{y_i\}_{i=1}^p$ to an adversary-desired target label y'. The poisoned dataset is denoted $D_{\text{poison}} = \{(e(x_i, \Delta), y')\}_{i=1}^p$, where $e(x_i, \Delta)$ is a trigger embedding function (e.g., pixel patch placed at a fixed location on a subset of training images). The dataset used to train the DNN is $D' = D \cup D_{\text{poison}}$. A DNN trained with D' (backdoored DNN) outputs y' with high probability whenever a poisoned input sample $(e(x, \Delta))$ is presented.

2.2 HÖLDER CONDITION

A real-valued function f, defined on a Euclidean space $\mathbb{R}^d$, satisfies a *Hölder Condition* (Knill, 1994) if for all $x, x' \in \mathbb{R}^d$, there exists a constant $C \geq 0$ and $0 < \alpha \leq 1$ such that $|f(x) - f(x')| \leq C \cdot ||x - x'||^\alpha$. Here, C and α are termed the *Hölder constant* and the *Hölder exponent*, respectively. When $\alpha = 1$, f satisfies the Lipschitz condition, with C termed the Lipschitz constant (Knill, 1994). For fixed α, a high value of C indicates that the function f is highly sensitive to input variations, suggesting greater changes in the function's output in response to small changes in the input. Conversely, a smaller C implies that f is less sensitive to input variations. We assess sensitivity of a backdoored DNN to its training samples D' in terms of C, and use this to inform design of a neuron pruning technique on the backdoored DNN to mitigate effects of backdoor attacks.

2.3 Definitions of the Performance Metrics

We present metrics to evaluate our defense against backdoor attacks below (Li et al., 2021c).

Clean Accuracy (ACC). Clean Accuracy measures the model's performance on clean (non-poisoned) samples, and defined as *ACC = (# Correctly Classified Clean Samples) / (# of Clean Samples)*.

Attack Success Rate (ASR). ASR quantifies the effectiveness of an attack, and is defined as *ASR = (# of Poisoned Samples Misclassified to Target Class) / (# of Poisoned Samples)*.

Robust Accuracy (RA). RA measures the number of poisoned samples that are correctly classified after the implementation of the defense mechanisms, and is defined as *RA = (# of Poisoned Samples Correctly Classified After the Defense) / (# of Poisoned Samples)*.

2.4 Threat Model

Adversary Assumptions: The adversary is assumed to have the capability to inject poisoned samples into a clean dataset. The adversary does not have direct access to or control over the model architecture, training process, or ability to retrain the model. The attacker's control is thus limited to poisoning the dataset, as described in (Barni et al., 2019; Gu et al., 2019; Nguyen & Tran, 2021).

Adversary Goals and Actions: The adversary's primary goal is to introduce a backdoor into a model trained by a user by injecting poisoned data into either (i) the training process or (ii) the fine-tuning process, while remaining undetected. In the first scenario, the adversary embeds triggers in publicly available datasets, which developers or data owners may inadvertently download and use for model training, unknowingly incorporating backdoors into their models. In the second scenario, the adversary targets the fine-tuning phase, where downstream users rely on pretrained models from large companies. While these pretrained models are assumed to be clean, the external or additional data used for fine-tuning may be contaminated with backdoor triggers. In both cases, the adversary's influence is restricted to modifying the training data by injecting poisoned samples, without direct access to the model architecture or control over the training process itself.

The adversary aims to ensure that: (i) models trained or fine-tuned with the poisoned dataset output adversary-specified target classes when presented with inputs containing embedded triggers; and (ii) the attack evades standard detection mechanisms by keeping the set of poisoned data small and blending the backdoor triggers seamlessly with the original data. This ensures the model's performance on clean data remains intact, avoiding suspicion or detection.

Attack Performance Metrics: Clean accuracy (CA) and attack success rate (ASR) are used to evaluate attack effectiveness. The adversary's goal is to ensure high values of CA and ASR.

2.5 Defender Model

Defender Assumptions: The defense is assumed to have access to the entire training dataset, but cannot reliably distinguish between clean and poisoned samples without further analysis (Li et al., 2021b; Zheng et al., 2022; Huang et al., 2022; Chen et al., 2022). In Sec. 3.1, we initially assume that the defense has access to a discriminative feature extractor (e.g., transformers, outputs of final pooling layer in CNNs) trained exclusively on clean data, which we term a *clean feature extractor*. In Sec. 3.2, we relax this assumption. When clean feature extractors are unavailable, self-supervised learning and iterative semi-supervised learning can be used to continually refine the feature space.

Defender Goals and Actions: The primary goal of the defense is to detect and neutralize effects of poisoned samples during training, ensuring that the model does not learn correlations between the trigger embedded by the adversary and the adversary-desired target class. In our defense pipeline, each sample is initially passed through a 'clean feature extractor'. The extracted features, along with their corresponding labels, are then used to train an classifier. After training, the defense prunes neurons in the hidden layers of classifier based on each neuron's output sensitivity to input noise. This pruning process helps identify and remove neurons that are activated by poisoned samples (poisoned neurons), thereby enhancing the overall DNN's resilience to backdoor attacks.

Defense Performance Metrics: We assess the pruned DNN using three metrics: clean accuracy (CA), attack success rate (ASR), and robust accuracy (RA).

3 Proposed Method

This section introduces our two defense methods based on the Hölder condition (Knill, 1994): **(1):** We design **Hölder Pruning**, a pruning defense that uses a clean feature extractor to effectively eliminate poisoned neurons without compromising model performance. **(2):** We develop a defense strategy named **Hölder Iteration Defense**, which can obtain a clean feature extractor from a poisoned dataset.

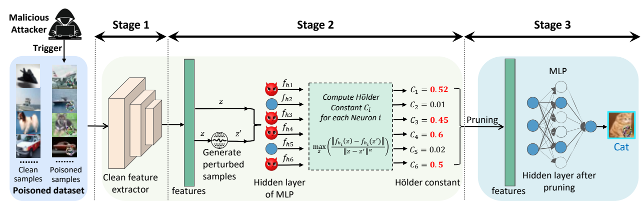
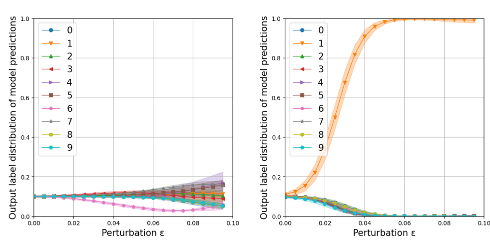


Figure 3: Proposed ***Hölder Pruning (HP)*** defense pipeline against backdoor attacks. **Stage 1:** Employ a clean feature extractor (e.g., self-supervised feature extractor, transformer, pre-trained CNN) to restrict the poisoned neurons to the hidden layers of classifier (shown as a MLP in the figure). **Stage 2:** Perturb features to identify poisoned neurons using their Hölder constants. **Stage 3:** Prune neurons with high Hölder constant values to eliminate poisoned neurons. This results in classifying poisoned inputs to their true class with a high probability, ensuring a low attack success rate and maintaining high classification accuracy on clean samples.

3.1 Hölder Pruning

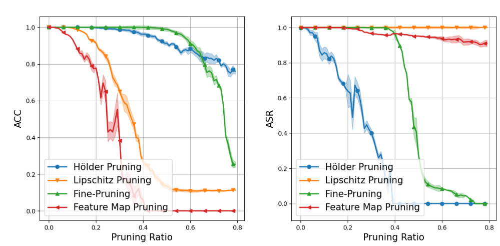
Our pruning strategy is based on the Hölder condition (Knill, 1994) and the availability of a clean feature extractor. The whole process is shown in Fig. 3.

Clean Feature Extractor. Using a feature extractor that exclusively trained on clean images will ensure the absence of poisoned neurons, meaning that trigger features corresponding to poisoned samples will not be amplified. We term such feature extractor *clean feature extractor*. Examples include self-supervised learning based feature extractors (Chen et al., 2020), vision and vision-language transformers (Dosovitskiy et al., 2021; Jia et al., 2021; Li et al., 2023; Radford et al., 2021), and outputs from the final pooling layer in pre-trained CNNs (He et al., 2016). Let z denote the feature vector extracted by the clean feature extractor for an input sample $x \in D'$.



(a) (b)

Figure 4: This figure compares effects of perturbing inputs on output label of a benign ((a)) and a backdoored ((b)) model with a BadNets (Gu et al., 2019) trigger. The backdoored model tends to classify samples to the target-class (Class 1) even for small perturbations. In contrast, label predictions of the benign model are erroneous only when perturbations are higher, suggesting that backdoored models are highly sensitive to specific perturbations of small magnitude.



(a) (b)

Figure 5: This figure compares effect of the pruning ratio on accuracy (ACC, Fig. (a)) and attack success rate (ASR, Fig. (b)) for three widely used pruning strategies- Lipschitz pruning (Zheng et al., 2022), fine pruning (Liu et al., 2018a), and feature map pruning (Huang & Bu, 2024)- and our Hölder pruning method. Our Hölder pruning approach is the only method that effectively maintains high classification accuracy while also yielding small ASR values. Shaded regions indicate standard deviation.

Perturbation in Hidden Layer. In backdoor attacks, a small imperceptible trigger in an input results in the DNN providing an (adversary-desired) output label (Li et al., 2021c). This suggests that backdoored models are highly sensitive to specific types of small-magnitude perturbations (characterized by the trigger). Fig. 4 compares effects of perturbing inputs on output labels of a benign (Fig. 4 (a)) and a backdoored (Fig. 4 (b)) model. We observe that the backdoored model tends to classify both clean and poisoned samples to the target-class (Class 1) even for perturbations of small magnitude; in contrast, the benign model makes incorrect predictions only for larger perturbations. We leverage this observation and carry out perturbations to induce misrepresentation of samples in a higher-dimensional space, specifically within the hidden layer of the classifier. This strategy helps highlight poisoned neurons that play a significant role in identifying trigger features (additional details in Appendix A.11). Our *Fast Gradient Sign Method at Neuron level (FGSM-Neuron)* Algorithm (Algo. 1 in Appendix A.2) adapts the FGSM attack (Goodfellow et al., 2015) to target individual neurons within the hidden layer. It aims to maximize discrepancy between outputs of a neuron for a feature vector z and its perturbed variant z'. In Appendix A.2, we provide a formal proof establishing a connection between our FGSM-Neuron algorithm and the Hölder constant value, which is defined below for measuring neuron sensitivity.

Pruning Strategy based on Hölder Constant. Unlike the Lipschitz method (Zheng et al., 2022), which measures maximum change of the entire training space, we use the Hölder constant to measure differences in each local region. This is motivated by an observation that poisoned neurons may exhibit varying sensitivities for different data points. We hypothesize that each training data point and its perturbation forms a local region in Hölder space. Neurons affected by backdoor attacks are likely to display higher Hölder constant values in at least one of these local regions. The relationship between the Hölder constant and perturbation is shown in Fig. 6. Let f_{h_i} denote the function mapping the input feature vector to the i^{th} hidden layer neuron. The Hölder constant for the i^{th} neuron is

$$C_i = \max_{\bar{z}} \left(\frac{\|f_{h_i}(\bar{z}) - f_{h_i}(\bar{z}')\|}{\|\bar{z} - \bar{z}'\|^{\alpha}} \right),$$

where $\bar{z}$ is the min-max scaled feature vector z and $0 < \alpha \leq 1$. We use $\alpha = 0.5$ in our experiments.

Our pruning strategy uses a threshold-based mechanism (we give details in Appendix A.8) to remove neurons that have a high value of C_i. We present an algorithmic procedure in Algorithm 2 in Appendix A.3. We also observe in Fig. 5 that our Hölder constant-based pruning strategy is highly effective in maintaining high classification accuracy and achieving lower ASR values.

3.2 Hölder Iteration Defense

Current research is increasingly adopting self-supervised learning (SSL) methods, particularly contrastive learning, to derive clean feature extractors from contaminated datasets (Chen et al., 2022; Gao et al., 2023; Huang et al., 2022). In these paradigms, models learn by assessing similarities between different projections of the same sample. By enhancing correlations among similar samples and reducing it among dissimilar ones, models can effectively acquire meaningful features (Jaiswal et al., 2020). SSL enables a model to generate comparable feature representations for perceptually similar inputs, thus averting amplification of triggers and minimizing poisoned neurons, qualifying it as a clean feature extractor. However, absence of labels in SSL may complicate the task of distinguishing between different output labels effectively (shown in Fig. 7(a)). Importantly, our pruning method can discern between poisoned and clean neurons, even with non-discriminative features, as demonstrated in Fig. 8, resulting in a low attack success rate after pruning. We define a sample x to be *robustly clean* if $f_\theta(x) = f_{\theta p}(x)$, where $f_\theta(x)$ is the model's predicted label and $f_{\theta p}(x)$ is the output post-Hölder pruning. We compile all robust clean samples into a dataset D_{clean}, and identify the remaining potential poison dataset as $D_{poison} \coloneqq D' \backslash D_{clean}$. The feature extractor is optimized by minimizing a semi-supervised loss described below:

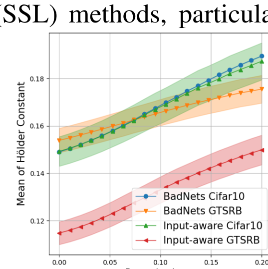
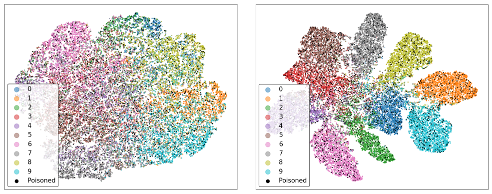


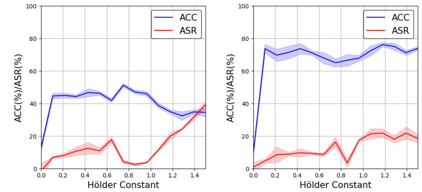
Figure 6: Hölder Constant value vs. perturbations for BadNets and InputAware attacks on CIFAR-10 and GTSRB datasets. Average Hölder Constant value increases with perturbation demonstrating high correlation between Hölder Constant and perturbations.

$$\mathcal{L} = \underbrace{\mathbb{E}_{(x,y)\sim D_{clean}}[\mathrm{H}(f_{\theta p}(x), y)]}_{\text{robust clean data}} + \lambda \underbrace{\mathbb{E}_{(u)\sim D_{poison}}[\|f_{\theta p}(u) - f_{\theta p}(u'))\|^2]}_{\text{potential poison data}},$$



(a) (b)

Figure 7: The t-SNE visualizations showcasing the feature spaces of clean and poisoned samples derived from a feature extractor trained on the CIFAR-10 dataset. Fig. (a) represents the feature space before applying Hölder iteration defense (HID). while Fig. (b) shows the feature space after applying HID.



(a) (b)

Figure 8: Performance of Hölder Pruning on feature extractors from Fig. 7(a) and Fig. 7(b). The clear separation shown in Fig. (a) results in only a few poisoned samples being selected as robust clean data for training the feature extractor in Fig. 7(a). Thus, the initial ASR in Fig. (b) is lower.

where $\mathrm{H}(p, q)$ denotes the cross-entropy between distributions p and q, u' represents the data augmentation for unlabeled data u. By cyclically segregating clean from potentially poisoned data and employing SSL, we enhance performance of the feature extractor (shown in Fig. 7(b)). We use this observation to inform development of an iterative pruning method that we term **Hölder Iteration Defense (HID)**. HID involves multiple rounds of pruning and semi-supervised learning to establish a clean feature extractor. An algorithmic procedure describing HID is presented in Appendix A.4.

4 Experiments

In this section, we conduct comprehensive experiments to evaluate effectiveness of our approach. We provide separate and detailed evaluations for **Hölder Pruning** and **Hölder Iteration Defense**.

Attacks settings. We conduct experiments involving **nine** SOTA backdoor attacks- BadNets (Gu et al., 2019), Sinusoidal signal backdoor attack (SIG) (Barni et al., 2019), Label-Consistent attack (LC) (Turner et al., 2019), Trojan (Liu et al., 2018b), Input-aware dynamic backdoor attack (Input-aware) (Nguyen & Tran, 2020), Sample-specific backdoor attack (SSBA) (Li et al., 2021d), Warping-based poisoned networks (WaNet) (Nguyen & Tran, 2021), Low frequency attack (LF) (Zeng et al., 2022), and BppAttack (Wang et al., 2022). We examine these attacks implemented by BackdoorBench (Wu et al., 2022) on CIFAR-10, CIFAR-100 (Krizhevsky & Hinton, 2009) and GTSRB (Houben et al., 2013) datasets. PreAct-ResNet18 structure (He et al., 2016) is used to train a feature extractor from scratch and select two layer-MLP with hidden layer size of 1024 for classification.

Defense settings. To evaluate our model, we divide our experiments into two parts. (a) We showcase performance of our Hölder pruning method when leveraging existing clean feature extractors, such as CLIP (Radford et al., 2021). Baseline defense methods for comparison are four SOTA pruning methods- Fine-Pruning(FP) (Liu et al., 2018a), Adversarial Neuron Pruning(ANP) (Wu & Wang, 2021), CLP (Zheng et al., 2022), and Feature map pruning(FMP) (Huang & Bu, 2024). (b) We demonstrate that Hölder Iteration Defense can learn from poisoned datasets **without any additional data**. To this end, we select four defense methods that meet this criterion- Anti-Backdoor Learning(ABL) (Li et al., 2021b), Channel Lipschitzness-based Pruning(CLP) (Zheng et al., 2022), DBD (Huang et al., 2022), and D-ST (Chen et al., 2022).

Metrics. We use the evaluation metrics ACC, ASR, and RA as described in Sec. 2.3.

4.1 Defense Performance against Backdoor attacks

Hölder Pruning – Secure defense with clean feature extractor. We compare performance of our Hölder Pruning with 4 other pruning methods using ACC, ASR, RA using the CIFAR-10 and GTSRB datasets in Table 1. We note that these experiments do not require additional clean data. Our experimental results illustrate robustness of Hölder Pruning compared to contemporary backdoor attacks. Specifically, under 9 distinct backdoor attacks, our approach consistently achieves significant reduction in ASR to below 3%, accompanied by only a marginal decrease in overall accuracy (averaging $\approx 2\%$), while maintaining a high robust accuracy (RA) of 82.5%. While FP and CLP mitigates several common attacks on MLPs, they face limitations when confronted with feature level attacks. In these instances, malicious data can be embedded within clean features, which renders FP and CLP ineffective. ANP and FMP primarily detect poisoned neurons by observing changes in the output labels produced by the model. However, in MLPs, the presence of poisoned neuron is more often reflected in numerical changes in activation function value rather than in output label flips. Results on the CIFAR-100 dataset are presented in the Appendix A.6.

Table 1: *Evaluation of Hölder Pruning (HP) against SOTA Pruning Methods*: Classification accuracy (ACC) for clean samples, attack success rate (ASR), and robust accuracy (RA) for Trojan samples for nine attacks– BadNets (Gu et al., 2019),LC (Turner et al., 2019), SIG (Barni et al., 2019), LF (Zeng et al., 2022), WaNet (Nguyen & Tran, 2021), Input-aware (Nguyen & Tran, 2020), SSBA (Li et al., 2021d), Trojan (Liu et al., 2018b), BppAttack (Wang et al., 2022)– on five defenses– ABL (Li et al., 2021b), CLP (Zheng et al., 2022), DBD (Huang et al., 2022), D-ST (Chen et al., 2022)– using transformer CLIP (Radford et al., 2021) as feature extractor with 5% poison rate (LC (Turner et al., 2019), SIG (Barni et al., 2019) used 0.5% on GTSRB). No additional clean data in use. Our HP consistently outperforms the SOTA.

Methods→	Benign			FP			ANP			CLP			FMP			**HP(ours)**		
Attacks↓	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑
BadNets	94.5	82.9	-	91.9	3.4	89.4	55.4	99.7	10.2	85.0	6.5	84.4	64.5	86.6	16.6	**93.7**	**0.0**	**93.3**
LC	94.9	90.1	-	88.6	7.9	81.3	93.2	6.8	84.9	93.7	13.0	81.2	91.8	6.3	85.4	**93.8**	**0.0**	**87.8**
SIG	95.0	43.6	-	90.9	15.1	79.6	92.8	41.6	39.7	85.9	50.7	29.5	91.5	14.6	69.6	**93.6**	**4.3**	**90.7**
LF	94.3	87.5	-	93.8	12.5	77.8	83.8	89.4	3.4	91.5	14.3	80.1	73.7	84.8	0.1	**94.3**	**2.5**	**91.1**
WaNet	93.1	20.4	-	92.5	6.9	87.4	21.5	98.2	1.1	86.1	**0.1**	87.2	69.1	10.0	60.4	**94.5**	**0.1**	**94.2**
Input-aware	94.6	96.9	-	89.3	97.4	12.6	92.2	96.2	13.2	88.8	96.7	12.6	87.5	97.4	12.1	**93.1**	**4.5**	**88.4**
SSBA	94.5	91.7	-	88.9	4.9	83.7	75.9	99.5	10.3	93.2	28.2	69.9	84.9	94.3	13.5	**93.8**	**1.1**	**90.3**
Trojan	91.2	100.0	-	88.8	100.0	0.0	93.2	99.9	0.1	91.1	99.9	0.1	86.6	100.0	0.0	**94.4**	**4.7**	**86.9**
BppAttack	94.9	93.9	-	91.1	73.5	21.4	**94.2**	66.1	32.2	82.5	4.1	60.2	91.7	91.4	7.8	92.9	**6.7**	**77.1**
Averages	94.1	78.6	-	90.6	35.7	59.2	78.0	77.4	21.6	88.6	34.8	56.1	82.3	56.1	34.7	**93.8**↑	**2.6**↓	**88.9**↑

CIFAR-10

Methods→	Benign			FP			ANP			CLP			FMP			**HP(ours)**		
Attacks↓	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑
BadNets	87.2	90.1	-	79.6	27.7	57.8	31.1	49.5	3.2	83.9	41.0	49.3	84.1	85.2	12.8	**85.4**	**3.2**	**80.6**
LC	87.2	19.3	-	86.3	**0.0**	75.3	50.6	**0.0**	47.1	83.5	**0.0**	74.3	**86.6**	**0.0**	78.4	85.3	**0.0**	**78.9**
SIG	89.3	44.4	-	**85.4**	28.2	32.2	57.3	26.3	23.7	83.7	37.7	25.9	83.7	37.7	25.9	82.9	**0.0**	**67.1**
LF	88.5	95.0	-	81.3	51.6	38.4	60.0	27.2	42.5	83.3	47.7	44.3	84.2	97.5	2.0	**84.7**	**2.0**	**79.8**
WaNet	84.9	8.6	-	74.9	0.3	65.3	68.8	**0.0**	61.0	82.0	0.3	68.6	83.3	6.2	70.7	**83.7**	0.4	**75.1**
Input-aware	88.4	96.2	-	83.6	50.1	39.7	40.7	99.4	0.0	**83.7**	74.7	20.0	83.4	98.7	1.1	83.0	**6.6**	**68.1**
SSBA	88.3	97.0	-	**85.2**	27.3	54.5	69.5	20.3	50.7	84.3	65.7	28.7	84.1	99.0	0.0	82.8	**2.9**	**75.1**
Trojan	89.4	99.6	-	84.6	99.5	0.0	58.2	65.5	19.4	84.8	98.6	1.2	**85.5**	99.8	0.0	85.2	**4.7**	**77.9**
BppAttack	88.4	91.3	-	**85.2**	9.6	43.2	69.4	22.4	37.9	83.5	47.0	32.4	83.5	47.1	32.4	85.0	**2.4**	**77.7**
Averages	87.9	71.2	-	82.9	32.7	45.1	56.1	34.5	31.7	83.6	45.9	38.3	84.1	63.5	24.8	**84.2**↑	**2.4**↓	**75.6**↑

GTSRB

Hölder Iteration Defense –In the absence of a clean extractor. We compare performance of our Hölder Iteration Defense with four SOTA models using ACC, ASR, and RA in Table 8. Our experimental results demonstrate that our Hölder Iteration Defense consistently performs better than SOTA methods against different types of backdoor attacks. We observed that ABL includes a step like poison suppression, which performs well against clean label attacks (LC, SIG). However, the first step— Backdoor Isolation— mainly defines suppression performance and limits overall model performance, keeping ASR high and ACC low in most cases. Similar to our approach, CLP also employs pruning, but it measures the Lipschitz constant of network parameters. In CLP, poisoned neurons are dispersed throughout the network via forward propagation, and most neurons exhibit both clean and toxic features. As a result, the Lipschitz constant is adequate to only detect and prune a small number of poisoned neurons, which results in a higher ASR value. DBD and D-ST propose mitigating backdoors through SSL, using self-supervised feature extraction to obtain clean feature extractors. In classification tasks using such a feature extractor, poisoned data often exhibits higher losses compared to clean data. However, such a loss-based design performs poorly against clean label attacks and feature-level attacks. In comparison, our Hölder Iteration Defense significantly reduces ASR and increases RA, regardless of the type of trigger.

4.2 Analysis

Clean Feature Extractor Improves Pruning. We observe that models equipped with a clean feature extractor exhibit faster and more significant reduction in attack success rate (ASR) while maintaining high levels of accuracy (ACC) during pruning. Results of our experiments using the Fine-Pruning method under BadNets (Gu et al., 2019) and BppAttack (Wang et al., 2022) are shown in Fig.9.

Table 2: *Evaluation of Hölder Iteration Defense (HID) against other SOTA End-to-End Backdoor Defenses*: Classification accuracy (ACC) for clean samples, attack success rate (ASR) and robust accuracy (RA) for Trojan samples for nine different attacks–BadNets (Gu et al., 2019), LC (Turner et al., 2019), SIG (Barni et al., 2019), LF (Zeng et al., 2022), WaNet (Nguyen & Tran, 2021), Input-aware (Nguyen & Tran, 2020), SSBA (Li et al., 2021d), Trojan (Liu et al., 2018b), BppAttack (Wang et al., 2022)– on five defenses– ABL (Li et al., 2021b), CLP (Zheng et al., 2022), DBD (Huang et al., 2022), D-ST (Chen et al., 2022)– with 5% poison rate (LC (Turner et al., 2019), SIG (Barni et al., 2019) used 0.5% on GTSRB). Our HID consistently outperforms SOTA.

Methods→	Benign			ABL			CLP			DBD			D-ST			**HID(ours)**		
Attacks↓	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑
BadNets	91.9	98.7	-	83.2	2.1	89.5	90.3	38.2	59.3	87.6	2.1	87.0	88.9	3.9	88.1	**90.9**	**1.5**	**90.6**
LC	93.3	99.5	-	43.7	5.7	45.2	90.1	99.3	0.0	74.9	99.9	0.0	88.1	99.9	0.0	**90.7**	**3.9**	**88.6**
SIG	93.6	97.1	-	49.3	5.3	28.3	**89.2**	96.2	3.7	74.8	95.6	4.2	88.5	73.6	28.0	88.2	**0.1**	**89.8**
LF	93.3	98.0	-	61.5	85.7	10.4	**90.8**	96.5	3.1	83.4	98.6	29.6	87.6	83.3	34.9	88.6	**3.7**	**88.1**
WaNet	92.7	85.5	-	80.6	79.2	17.4	89.5	2.3	86.8	72.2	9.9	69.4	88.4	11.0	87.4	**92.1**	**1.3**	**91.0**
Input-aware	91.5	90.2	-	58.1	99.5	0.0	85.4	95.0	3.9	89.3	7.0	83.3	88.8	73.3	42.9	**89.4**	**4.7**	**86.1**
SSBA	93.2	94.9	-	80.5	5.6	79.7	**90.5**	44.9	49.6	72.1	99.5	0.4	88.6	84.7	12.8	89.5	**4.4**	**82.4**
Trojan	93.8	99.9	-	66.8	100.0	0.0	92.7	99.9	0.1	72.9	99.8	0.0	53.1	99.9	0.0	**91.1**	**7.5**	**83.6**
BppAttack	91.6	99.9	-	63.7	81.6	12.5	**90.1**	2.2	82.6	**90.1**	9.8	72.4	88.4	96.2	5.3	89.4	**0.6**	**89.0**
Averages	92.7	96.0	-	65.2	51.6	31.4	89.8	63.8	32.1	79.7	68.0	36.4	84.5	69.5	33.2	**89.9**↑	**3.0**↓	**87.7**↑

CIFAR-10

Methods→	Benign			ABL			CLP			DBD			D-ST			**HID(ours)**		
Attacks↓	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑
BadNets	97.3	57.9	-	96.0	**0.0**	96.6	97.5	86.4	13.4	83.4	0.0	83.8	87.7	30.4	64.7	**98.8**	**0.0**	**98.8**
LC	97.8	53.1	-	31.1	**0.0**	31.6	**98.4**	**0.0**	**98.4**	80.4	**0.0**	80.4	90.9	**0.0**	92.0	**98.4**	**0.0**	**98.4**
SIG	98.5	66.7	-	30.6	54.9	4.1	**98.0**	78.9	14.8	79.2	91.2	6.3	84.1	66.5	21.0	96.4	**6.1**	**85.5**
LF	98.4	98.6	-	19.3	10.4	5.4	**97.2**	99.2	0.0	84.1	4.0	80.2	84.7	91.5	7.4	96.2	**3.2**	**90.0**
WaNet	98.4	92.9	-	44.2	74.7	11.7	96.4	94.7	5.1	80.2	**0.0**	80.2	89.8	53.9	40.7	**98.7**	**0.0**	**98.7**
Input-aware	98.2	92.8	-	13.7	82.6	1.9	94.4	35.2	61.9	88.2	99.7	0.3	92.1	78.2	43.7	**99.3**	**0.0**	**99.1**
SSBA	98.1	99.3	-	13.3	69.8	2.7	**97.8**	97.3	2.6	81.7	99.9	0.0	88.1	97.8	1.8	95.4	5.7	**92.4**
Trojan	98.5	100.0	-	85.4	100.0	0.0	**98.2**	99.8	0.0	66.8	99.9	0.0	86.8	99.9	0.0	94.3	**8.0**	**91.4**
BppAttack	98.2	98.9	-	7.5	99.8	0.3	97.6	12.1	82.2	87.7	99.9	0.0	91.5	95.4	5.4	**97.8**	**8.4**	**89.1**
Averages	98.1	84.5	-	37.9	59.3	17.1	97.2	67.0	30.9	81.3	55.0	36.8	88.4	68.2	30.7	**97.3**↑	**3.4**↓	**93.7**↑

GTSRB

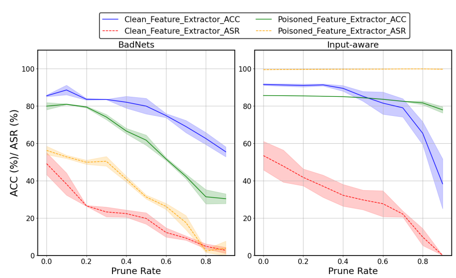


Figure 9: This figure illustrates that using a clean feature extractor improves performance of pruning techniques, indicated by lower ASR and high ACC.

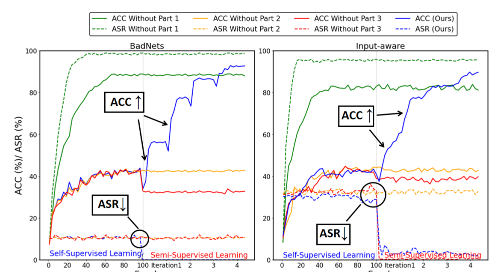


Figure 10: *Ablation*: Importance of components of our Hölder iteration defense- Clean feature extractor (Part 1), Hölder Pruning (Part 2), Iteration (Part 3).

General Plug-in Method. We have selected multiple models, including SimCLR (Chen et al., 2020), Pretrained ResNet-18 (He et al., 2016), ViT (Dosovitskiy et al., 2021), ALIGN (Jia et al., 2021), CLIP (Radford et al., 2021), and BLIP-2 (Li et al., 2023) to validate generality of our methodology. Our method seamlessly integrates with self-supervised learning, Transformer, and pretrained models. This integration is facilitated by our pruning technique, which operates exclusively within classifier without requiring alterations to feature extractors. Results are presented in Table 3.

Ablation Study. Fig. 10 presents results of an ablation study on our Hölder Iteration Defense. We assess necessity of each component through the following experiments: Clean Feature Extractor (Part 1), Hölder Pruning (Part 2), and Iteration (Part 3). Our results indicate that (a) a clean feature extractor significantly reduces ASR, (b) Hölder Pruning allows for later selection of clean data, which helps maintain lower ASR during training, (c) iterative method enhances accuracy in subsequent stages.

Runtime Comparison. We measured runtime of defense methods on the CIFAR-10 and GTSRB datasets using a PreAct-ResNet18 architecture on NVIDIA GeForce RTX 3090. Table 4 shows that our HP and HID are significantly faster than three other methods, with run-time as low as **30** seconds.

Table 3: *Evaluation of Hölder Pruning (HP) with Different Clean Feature Extractors*: Classification accuracy (ACC) for clean samples, attack success rate (ASR) and robust accuracy (RA) for Trojan samples for eight attacks–BadNets (Gu et al., 2019), LC (Turner et al., 2019), SIG (Barni et al., 2019), LF (Zeng et al., 2022), WaNet (Nguyen & Tran, 2021), SSBA (Li et al., 2021d), Trojan (Liu et al., 2018b), BppAttack (Wang et al., 2022)– on six feature extractors– SimCLR (Chen et al., 2020), Pretrained Resnet-18 (PreRes) (He et al., 2015), Transformers: VIT (Dosovitskiy et al., 2021), ALIGN (Jia et al., 2021), CLIP (Radford et al., 2021), BLIP-2 (Li et al., 2023)– for the CIFAR-10 with 5% poison rate. Our HP is effective for all clean feature extractors.

Methods→	SimCLR			PreRes			VIT			ALIGN			CLIP			BLIP-2		
Attacks↓	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑
BadNets	90.9	1.5	90.6	93.7	0.0	94.1	90.1	0.0	89.3	88.2	0.0	88.5	93.7	0.0	93.3	98.0	0.0	97.8
LC	90.7	3.9	88.6	93.9	1.2	91.8	95.8	1.5	93.9	90.1	2.4	86.3	93.8	0.0	87.8	97.5	0.0	98.0
SIG	85.2	0.1	89.8	93.8	2.3	81.7	95.7	7.2	84.2	89.9	2.4	74.2	93.6	4.3	90.7	96.9	1.4	90.0
LF	85.6	3.7	88.1	93.8	1.9	90.9	84.9	11.9	69.1	88.0	2.6	82.9	94.3	2.5	91.1	97.0	1.9	96.4
WaNet	92.1	1.3	91.0	91.4	3.2	90.8	90.2	1.0	90.9	88.1	0.0	87.5	94.5	0.1	94.2	94.4	0.0	93.6
SSBA	88.5	4.4	82.4	92.7	9.3	79.4	91.4	0.0	91.4	88.4	3.1	84.9	93.8	1.1	90.3	98.0	4.0	91.5
Trojan	91.1	7.5	83.6	87.5	10.9	84.1	94.9	7.0	83.6	83.8	4.1	78.9	94.4	4.7	86.9	98.1	14.9	83.1
BppAttack	89.1	3.0	87.7	92.8	5.2	86.7	94.5	5.8	84.8	89.0	5.3	74.9	92.9	6.7	77.1	97.6	3.6	92.6
Averages	**93.9**	**2.3**	**89.0**	**92.5**	**4.2**	**87.5**	**92.2**	**4.3**	**85.9**	**88.2**	**2.4**	**82.3**	**93.9**	**2.4**	**88.9**	**97.2**	**3.2**	**92.9**

Table 4: *Runtime Comparison*: Average runtimes of our HP and HID and three other methods- ABL (Li et al., 2021b), DBD (Huang et al., 2022), and D-ST (Chen et al., 2022) on the CIFAR-10 and GTRSB datasets. HP is up to 1000x faster; even when clean feature extractors are unavailable, HID is up to 10x faster.

Methods(Sec)	ABL	DBD	D-ST	HID(Ours)	HP(Ours)
CIFAR-10	4680	43200	49600	5400	30
GTSRB	2280	37800	44900	3900	30

Defense Effectiveness Under Different Poisoning Rates. We conducted experiments with varying poisoning rates (from 0.1% to 5%) to explore their impact on efficacy of our Hölder Iteration Defense. Table 5 shows that the poisoning rate does not significantly affect the performance of our approach.

Table 5: *Evaluation of Hölder Iteration Defense (HID) against Different Poison Rates*: ACC, ASR, and RA for Trojan samples for eight different attacks–BadNets (Gu et al., 2019), LC (Turner et al., 2019), SIG (Barni et al., 2019), LF (Zeng et al., 2022), WaNet (Nguyen & Tran, 2021), SSBA (Li et al., 2021d), Trojan (Liu et al., 2018b), BppAttack (Wang et al., 2022) on different poison rates. (LC, SIG on GTSRB should have poison rate lower than 0.5% - indicated by 'NA' entries). Performance of our HID remains unaffected by changes in poison rate.

Dataset	CIFAR-10									GTSRB								
poison rate→	0.1%			1%			5%			0.1%			1%			5%		
Attacks↓	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑
BadNets	91.5	0.0	91.3	90.9	1.5	90.6	92.5	0.0	92.5	98.8	0.0	98.7	98.8	0.0	98.8	98.8	0.0	98.8
LC	92.5	0.0	92.5	92.4	4.5	87.3	90.7	3.9	88.6	98.4	0.0	98.4	NA	NA	NA	NA	NA	NA
SIG	91.5	0.0	91.1	90.2	0.0	90.1	85.2	0.1	89.8	98.2	0.1	90.5	NA	NA	NA	NA	NA	NA
LF	92.2	1.1	90.9	87.9	3.1	85.0	85.6	3.7	88.1	94.4	2.0	92.9	96.6	3.6	92.4	93.2	3.2	90.0
WaNet	92.5	0.0	92.3	91.9	1.0	90.1	92.1	1.3	91.0	98.5	0.0	98.5	98.5	0.0	98.5	98.7	0.0	98.7
SSBA	91.0	0.0	91.0	91.1	0.7	89.2	88.5	4.4	82.4	96.0	0.0	96.0	94.5	1.1	92.6	92.4	5.7	92.4
Trojan	91.8	0.6	92.1	88.3	2.4	87.4	91.1	7.5	83.6	92.1	1.4	90.9	94.2	7.8	84.1	91.3	8.0	91.4
BppAttack	92.5	1.0	89.1	90.7	1.5	88.9	88.4	0.6	89.0	93.9	4.5	88.9	96.0	2.5	91.2	97.8	8.4	89.1
Averages	**91.8**	**0.3**	**91.3**	**90.4**	**1.8**	**88.6**	**88.8**	**2.7**	**87.9**	**96.3**	**1.0**	**94.4**	**96.4**	**1.8**	**92.9**	**95.4**	**3.1**	**93.4**

5 CONCLUSION

We developed *Hölder Pruning*, a defense against backdoor attacks in DNNs. By leveraging clean feature extractors and the Hölder constant, our method enhanced pruning accuracy and model robustness. Extensive experiments across three datasets (CIFAR-10, CIFAR-100, GTSRB) and against nine backdoor attacks (BadNets, LC, SIG, LF, WaNet, Input-Aware, SSBA, Trojan, BppAttack) demonstrated superiority of *Hölder Pruning* over eight SOTA defenses (FP, ANP, CLP, FMP, ABL, DBD, D-ST). When a clean feature extractor was unavailable, we introduced the *Hölder Iteration Defense*, which used self-supervised and iterative semi-supervised learning to continually refine the feature space. *Hölder Pruning* and *Hölder Iteration Defense* consistently yielded high classification accuracy, high robust accuracy, and low attack success rates. Additionally, Hölder Pruning achieved up to 1000x faster runtime compared to SOTA defenses. Even when clean feature extractors were not available, our approach was up to 10x faster than comparable methods.

REFERENCES

Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 1505–1521. USENIX Association, August 2021. ISBN 978-1-939133-24-3. URL https://www.usenix.org/conference/usenixsecurity21/presentation/bagdasaryan.

Mauro Barni, Kassem Kallas, and Benedetta Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing*, 2019.

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.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.

Weixin Chen, Baoyuan Wu, and Haoqian Wang. Effective backdoor defense by exploiting sensitivity of poisoned samples. In *Advances in Neural Information Processing Systems*, 2022.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning, 2017.

Rene Y Choi, Aaron S Coyner, Jayashree Kalpathy-Cramer, Michael F Chiang, and J Peter Campbell. Introduction to machine learning, neural networks, and deep learning. *Translational vision science & technology*, 9(2):14–14, 2020.

Khoa Doan, Yingjie Lao, Weijie Zhao, and Ping Li. Lira: Learnable, imperceptible and robust backdoor attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11966–11976, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

Kuofeng Gao, Yang Bai, Jindong Gu, Yong Yang, and Shu-Tao Xia. Backdoor defense via adaptively splitting poisoned dataset, 2023.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain, 2019.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented transformer for speech recognition, 2020.

Junfeng Guo, Ang Li, and Cong Liu. Aeva: Black-box backdoor detection using adversarial extreme value analysis, 2022.

Jonathan Hayase, Weihao Kong, Raghav Somani, and Sewoong Oh. Spectre: Defending against backdoor attacks using robust statistics. In *International Conference on Machine Learning*, pp. 4129–4139. PMLR, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks, 2016.

Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, number 1288, 2013.

Dong Huang and Qingwen Bu. Adversarial feature map pruning for backdoor. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=IOEEDkla96.

Kunzhe Huang, Yiming Li, Baoyuan Wu, Zhan Qin, and Kui Ren. Backdoor defense via decoupling the training process. In *International Conference on Learning Representations*, 2022.

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information, 2018.

Shashank Mohan Jain. Hugging face. In *Introduction to transformers for NLP: With the hugging face library and models to solve problems*, pp. 51–67. Springer, 2022.

Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2020.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision, 2021.

Kaidi Jin, Tianwei Zhang, Chao Shen, Yufei Chen, Ming Fan, Chenhao Lin, and Ting Liu. Can we mitigate backdoor attack using adversarial detection methods? *IEEE Transactions on Dependable and Secure Computing*, 2022.

N.J. Kalton. Spaces of lipschitz and hölder functions and their applications. *Collectanea Mathematica*, 55(2):171–217, 2004.

Oliver Knill. Probability and stochastic processes with applications. *Havard Web-Based*, 5, 1994.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution, 2022.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world, 2017.

Daniela Lera and Ya D. Sergeyev. Lipschitz and hölder global optimization using space-filling curves. *Applied Numerical Mathematics*, 60(1-2):115–129, 2010.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023.

Linyang Li, Demin Song, Xiaonan Li, Jiehang Zeng, Ruotian Ma, and Xipeng Qiu. Backdoor attacks on pre-trained models by layerwise weight poisoning. *arXiv preprint arXiv:2108.13888*, 2021a.

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

Yiming Li, Tongqing Zhai, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor attack in the physical world, 2021c.

Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. Invisible backdoor attack with sample-specific triggers. In *IEEE International Conference on Computer Vision (ICCV)*, 2021d.

Siyuan Liang, Xingxing Wei, Siyuan Yao, and Xiaochun Cao. Efficient adversarial attacks for visual object tracking, 2020.

Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks, 2018a.

Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-221, 2018*. The Internet Society, 2018b.

Orson Mengara, Anderson Avila, and Tiago H. Falk. Backdoor attacks to deep neural networks: A survey of the literature, challenges, and future research directions. *IEEE Access*, 12:29004–29023, 2024. doi: 10.1109/ACCESS.2024.3355816.

Rui Min, Zeyu Qin, Li Shen, and Minhao Cheng. Towards stable backdoor purification through feature shift tuning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack, 2020.

Tuan Anh Nguyen and Anh Tuan Tran. Wanet - imperceptible warping-based backdoor attack. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=eEn8KTtJOx.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

Hossein Souri, Liam Fowl, Rama Chellappa, Micah Goldblum, and Tom Goldstein. Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch, 2022.

Matthew Tancik, Ben Mildenhall, and Ren Ng. Stegastamp: Invisible hyperlinks in physical photographs, 2020.

Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

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 *Proceedings of the IEEE Symposium on Security and Privacy (IEEE SP)*, San Francisco, CA, 2019.

Huandong Wang, Changzheng Gao, Yuchen Wu, Depeng Jin, Lina Yao, and Yong Li. Pategail: a privacy-preserving mobility trajectory generator with imitation learning. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. ISBN 978-1-57735-880-0. doi: 10.1609/aaai.v37i12.26700. URL https://doi.org/10.1609/aaai.v37i12.26700.

Shuo Wang, Surya Nepal, Carsten Rudolph, Marthie Grobler, Shangyu Chen, and Tianle Chen. Backdoor attacks against transfer learning with pre-trained deep learning models. *IEEE Transactions on Services Computing*, 15(3):1526–1539, 2020.

Zhenting Wang, Juan Zhai, and Shiqing Ma. Bppattack: Stealthy and efficient trojan attacks against deep neural networks via image quantization and contrastive adversarial learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15074–15084, June 2022.

Shaokui Wei, Mingda Zhang, Hongyuan Zha, and Baoyuan Wu. Shared adversarial unlearning: Backdoor mitigation by unlearning shared adversarial examples, 2023.

Baoyuan Wu, Hongrui Chen, Mingda Zhang, Zihao Zhu, Shaokui Wei, Danni Yuan, and Chao Shen. Backdoorbench: A comprehensive benchmark of backdoor learning. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

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

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.

Yi Zeng, Won Park, Z. Morley Mao, and Ruoxi Jia. Rethinking the backdoor attacks' triggers: A frequency perspective, 2022.

Zaixi Zhang, Qi Liu, Zhicai Wang, Zepu Lu, and Qingyong Hu. Backdoor defense via deconfounded representation learning, 2023.

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

Runkai Zheng, Rongjun Tang, Jianze Li, and Li Liu. Data-free backdoor removal based on channel lipschitzness. *arXiv preprint arXiv:2208.03111*, 2022.

Mingli Zhu, Shaokui Wei, Li Shen, Yanbo Fan, and Baoyuan Wu. Enhancing fine-tuning based backdoor defense with sharpness-aware minimization, 2023a.

Mingli Zhu, Shaokui Wei, Hongyuan Zha, and Baoyuan Wu. Neural polarizer: A lightweight and effective backdoor defense via purifying poisoned features, 2023b.

APPENDIX

This appendix provides the formal proof establishing the connection between FGSM-Neuron algorithm and Hölder constant value for measuring neuron sensitivity. Additionally, we detail algorithmic steps of the ***Hölder Pruning*** defense and ***Hölder Iteration Defense*** against backdoor attacks and presents the results of our experiments on the CIFAR-100 dataset. We also provide a brief summary of related work, and descriptions of the backdoor attacks and defense methodologies that we evaluate our ***Hölder Pruning*** defense and ***Hölder Iteration Defense***.

This Appendix is organized into the following sections:

A.1 BROADER IMPACT

Deep neural networks (DNNs) have demonstrated exceptional performance in numerous applications such as computer vision, speech recognition, and recommendation systems. However, training deep learning models typically requires large amounts of data and computational resources, often involving third-party data or servers. This dependency raises significant security concerns, particularly when using large public datasets that may contain malicious or poisoned data samples. Attackers can inject poisoned data into normal samples, embedding backdoors in the model during training, thus posing a substantial threat to the model's deployment. To mitigate these risks, defenders must remove potential backdoors from the model before actual deployment to ensure its security and reliability. In this work, we propose a lightweight, plug-and-play defense strategy that can be applied in real-world scenarios to reduce risk of model poisoning. We aim to draw the community's attention to practical defense strategies to enhance the security of machine learning models.

A.2 FGSM-NEURON ALGORITHM AND HÖLDER CONSTANT VALUE FOR MEASURING NEURON SENSITIVITY

In this section, we first present the FGSM-Neuron algorithm, which generates perturbed input feature vectors for each neuron in the hidden layer. The objective is to maximize the discrepancy between the outputs of a neuron for an original feature vector and its perturbed variant. The FGSM-Neuron algorithm adapts the FGSM attack (Goodfellow et al., 2015) to specifically target individual neurons within the hidden layer. Subsequently, we formally establish the connection between the FGSM-Neuron algorithm and the Hölder constant value, which is used to measure neuron sensitivity.

Next, we review the necessary notations to demonstrate the connection between the FGSM-Neuron algorithm, as outlined in Algorithm 1, and the Hölder constant, which is used for measuring neuron sensitivity.

Algorithm 1 FGSM-Neuron

1: **Input:** f_{h_i}: i^{th} neuron in the hidden layer of classifier, $\bar{z}$: min-max scaled feature vector z of input sample x, ϵ: perturbation size.
2: Initialize: $\bar{z}' = \bar{z} +$ random noise
3: Calculate the loss: loss $= \|f_{h_i}(\bar{z}) - f_{h_i}(\bar{z}')\|^2$
4: Calculate the gradient: grad $= \frac{\partial \text{loss}}{\partial \bar{z}}$
5: Update: $\bar{z}' = \bar{z} + \epsilon \cdot \text{sign}(\text{grad})$
6: Apply feature vector bounds: $\bar{z}' = \text{clamp}(\bar{z}', \min = 0, \max = 1)$
7: **Output:** $\bar{z}'$

Let $Z = [z_j]_{j=1}^n$ be the set of features extracted by the clean feature extractor for a given input set $[x_j]_{j=1}^n$, where $x_j \in D'$ and D' denotes the training dataset that includes both clean and poisoned input samples. Let $[\bar{z}_j]_{j=1}^n$ denote the min-max scaled features of the feature set $[z_j]_{j=1}^n$. Let $[f_{h_i}]_{i=1}^l$ denote the outputs of each neuron in the hidden layer of a classifier. For a Hölder exponent $0 < \alpha \leq 1$, we define the Hölder constant C_i for the i^{th} neuron as:

$$C_i = \max_{\bar{z}} \left(\frac{\|f_{h_i}(\bar{z}) - f_{h_i}(\bar{z}')\|}{\|\bar{z} - \bar{z}'\|^\alpha} \right),$$

Recall that the Hölder constant C_i is used to measure the sensitivity of a neuron to perturbed inputs and that high sensitivity indicates that the corresponding neuron is a poison neuron enabling backdoor attacks (Fig. 4).

The following proposition establishes the connection between the FGSM-Neuron algorithm and Hölder constant values:

Proposition 1. *Assuming that the function $f_{h_i}(\cdot)$, which models the input-output relation of the i^{th} hidden layer neuron, is continuously differentiable, the FGSM-Neuron algorithm, as presented in Algorithm 1, identifies the perturbed sample $\bar{z}'$ that maximizes the term $\frac{\|f_{h_i}(\bar{z}) - f_{h_i}(\bar{z}')\|}{\|\bar{z} - \bar{z}'\|^\alpha}$ for a given scaled feature $\bar{z}$.*

Proof. Recall from Line 5 of Algorithm 1,

$$\bar{z}' = \bar{z} + \epsilon \cdot \text{sign}\left(\frac{\partial \|f_{h_i}(\bar{z}) - f_{h_i}(\bar{z}')\|}{\partial \bar{z}} \right).$$

This yields:

$$\|\bar{z}' - \bar{z}\|^\alpha = \|\epsilon \cdot \text{sign}\left(\frac{\partial \|f_{h_i}(\bar{z}) - f_{h_i}(\bar{z}')\|}{\partial \bar{z}} \right)\|^\alpha = \epsilon^\alpha.$$

Then we can write:

$$\frac{\|f_{h_i}(\bar{z}) - f_{h_i}(\bar{z}')\|}{\|\bar{z} - \bar{z}'\|^\alpha} = \frac{\|f_{h_i}(\bar{z}) - f_{h_i}(\bar{z}')\|}{\epsilon^\alpha}.$$

Finally, noting that Line 5 of Algorithm 1 updates in the ascent direction of $\|f_{h_i}(\bar{z}) - f_{h_i}(\bar{z}')\|$ using a stochastic sign gradient approach proves that the perturbed sample $\bar{z}'$ computed by the FGSM-Neuron algorithm maximizes the term $\frac{\|f_{h_i}(\bar{z}) - f_{h_i}(\bar{z}')\|}{\|\bar{z} - \bar{z}'\|^\alpha}$. □

Proposition 1 demonstrates that the FGSM-Neuron algorithm, finds the perturbed sample $\bar{z}'$ that maximizes the term $\frac{\|f_{h_i}(\bar{z}) - f_{h_i}(\bar{z}')\|}{\|\bar{z} - \bar{z}'\|^\alpha}$ for a given min-max scaled feature sample $\bar{z}$. The assumption that the function $f_{h_i}(\cdot)$, which models the input-output relation of the i^{th} hidden layer neuron, is continuously differentiable is a standard one for DNNs. This property has been demonstrated to hold for most DNN architectures, including CNNs and MLPs, as discussed in (Goodfellow et al., 2015; Choi et al., 2020). Identifying C_i values for each scaled feature sample $\bar{z}$ and obtaining the maximum C_i value among them provides a metric to measure the maximum sensitivity to any given feature of an input in the training dataset, which can be used to identify the poisoned neurons.

A.3 *Hölder Pruning* ALGORITHM

Algorithm 2 below presents the detailed algorithmic procedure of the Hölder pruning defense outlined in Section 3. The algorithm processes the outputs of each neuron in the hidden layer of a multi-layer perceptron (MLP), denoted by $[f_{h_i}]_{i=1}^{l}$, against the features $Z = [z_j]_{j=1}^{n}$ extracted by a clean feature extractor for corresponding inputs $[x_j]_{j=1}^{n}$, the maximum perturbation size ϵ, the Hölder exponent $0 < \alpha \leq 1$, and the number of neurons $p < l$ that need to be pruned. In *line 3*, the features $Z = [z_j]_{j=1}^{n}$ are scaled using a min-max scaler to obtain the scaled feature set $[\bar{z}_j]_{j=1}^{n}$. *Line 7* of the algorithm computes the perturbed feature $\bar{z}'_j$ corresponding to each scaled feature sample $\bar{z}_j$ using the FGSM-Neuron algorithm presented in Algorithm 1. *Lines 8 and 9* compute the Hölder constant for the i^{th} hidden layer neuron f_{h_i} with respect to the scaled feature vector $\bar{z}_j$ and accumulate each Hölder constant value corresponding to $\bar{z}_j$, for $j = 1, 2, \ldots, n$, into the set $\bar{C}$. *Line 11* finds the maximum Hölder constant value observed for the i^{th} neuron across the input scaled feature samples $[\bar{z}_j]_{j=1}^{n}$ and appends it to the Hölder constant set C. *Line 13* sorts the Hölder constant values in the set C in descending order, and *Line 14* outputs the indices of the first $p < l$ neurons from the ordered set C to be pruned.

Algorithm 2 Hölder Pruning

1: **Input:** $[f_{h_i}]_{i=1}^{l}$: l neurons in the hidden layer of MLP, $Z = [z_j]_{j=1}^{n}$: n feature vectors of input samples $[x_j]_{j=1}^{n}$ extracted from the clean feature extractor, ϵ: maximum perturbation size, α: Hölder exponent, $p < l$: number of neurons that need to be pruned.
2: Initialize Hölder constant set $C := \{\emptyset\}$;
3: Min-Max scaling of features $[\bar{z}_j]_{j=1}^{n} = \left[\frac{z_j - \min(Z)}{\max(Z) - \min(Z)}\right]_{j=1}^{n}$, where $\min(Z)$ and $\max(Z)$ are the minimum and maximum values of each feature dimension across all n feature vectors;
4: **for** $i = 1, 2, \ldots, l$ **do**
5:     $\bar{C} := \{\emptyset\}$
6:     **for** $j = 1, 2, \ldots, n$ **do**
7:         $\bar{z}'_j$ = FGSM-Neuron$(f_{h_i}, \bar{z}_j, \epsilon)$
8:         Compute Hölder constant for i^{th} hidden layer neuron f_{h_i} w.r.t to scaled feature vector $\bar{z}_j$:
9:         $\bar{C} \leftarrow \bar{C} \cup \frac{|f_{h_i}(\bar{z}_j) - f_{h_i}(\bar{z}'_j)|}{||\bar{z}_j - \bar{z}'_j||^{\alpha}}$
10:     **end for**
11:     $C \leftarrow C \cup \max(\bar{C})$
12: **end for**
13: Sort the set C in descending order
14: **Output:** Indices of first $p < l$ neurons from the set C

A.4 *Hölder Iteration Defense* ALGORITHM

In below we present the Hölder Iteration Defense (HID) algorithm. HID can obtain a clean end-to-end DNN model in the absence of a clean feature extractor.

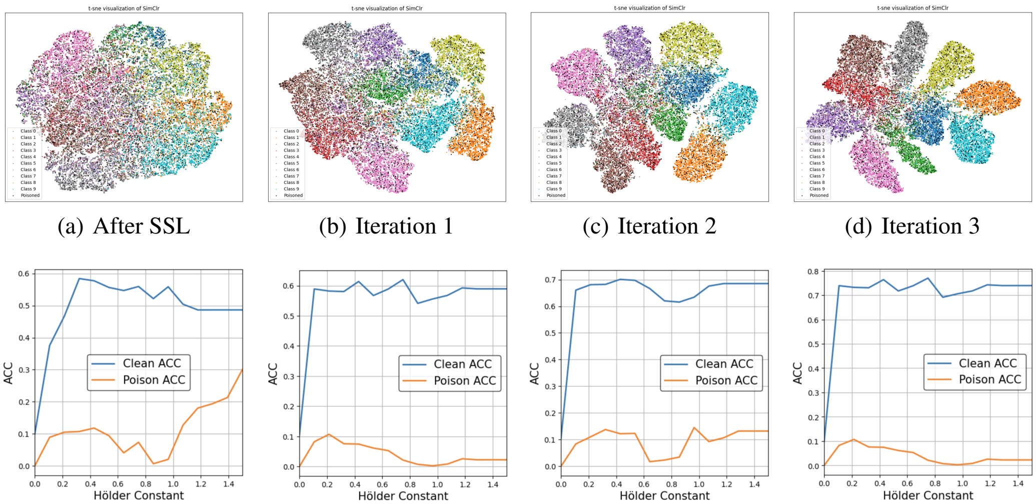
Self-supervised learning: The model undergoes contrastive learning using unlabeled training data, where it extracts and ensures consistent features from two different perspectives of the same image.

NT-Xent Loss Given a mini-batch consisting of N unique samples, SimCLR applies two distinct data augmentations to each sample, resulting in $2N$ augmented samples. The loss for a positive pair of samples (i, j) can be defined as:

$$\mathcal{L}_{i,j} = -\log \frac{\exp\left(\left(\frac{z_i \cdot z_j}{||z_i|| \cdot ||z_j||}\right)/\tau\right)}{\sum_{k=1}^{2N} \mathbb{I}_{[k \neq i]} \cdot \exp\left(\left(\frac{z_i \cdot z_j}{||z_i|| \cdot ||z_j||}\right)/\tau\right)}, \tag{1}$$

where z_i and z_j are the representations of the augmented samples i and j respectively, τ is the temperature parameter, $(\mathbb{I}_{[k \neq i]})$ is an indicator function that is 1 if $k \neq i$, and 0 otherwise. The NT-Xent Loss is computed across all $2N$ positive pairs in this mini-batch.

Semi-supervised learning Loss:



(a) After SSL (b) Iteration 1 (c) Iteration 2 (d) Iteration 3

(e) After SSL (f) Iteration 1 (g) Iteration 2 (h) Iteration 3

Figure 11: The t-SNE visualization of the feature space generated by the feature extractor trained during the Hölder Iteration Defense is shown in Fig (a) to Fig (d). Hölder Pruning results (accuracy of clean and poisoned samples) under those feature extractors are shown in Fig (e) to Fig (h). We observe that performance of the feature extractor improves with each iteration; specifically, boundaries between classes become more distinct.

Let a sample x from training data D to be *robustly clean* if $f_\theta(x) = f_{\theta p}(x)$, where $f_\theta(x)$ is the model's predicted label and $f_{\theta p}(x)$ is the output post-Hölder pruning. We compile all robust clean samples into a dataset D_{clean}, and identify the remaining potential poison dataset as $D_{\text{poison}} := D' \setminus D_{\text{clean}}$. The feature extractor is optimized by minimizing a semi-supervised loss described as follows:

$$\mathcal{L} = \underbrace{\mathbb{E}_{(x,y)\sim D_{\text{clean}}}[\mathrm{H}(f_{\theta p}(x), y)]}_{\text{robust clean data}} + \lambda \underbrace{\mathbb{E}_{(u)\sim D_{\text{poison}}}[\| f_{\theta p}(u) - f_{\theta p}(u'))\|^2]}_{\text{potential poison data}},$$

where $\mathrm{H}(p, q)$ denotes the cross-entropy between distributions p and q, and u' represents the data augmentation for unlabeled data u.

Algorithm: We summarize the HID in Algorithm 3. We have also shown results for SSL and three iterations on the CIFAR-10 dataset under a BadNets attack (Fig. 11).

Algorithm 3 Hölder Iteration Defense

Input: D the poisoned training set, e_{ssl} number of training epochs for self-supervised learning, e_{mix} number of training epochs for semi-self-supervised learning in each iteration, G_θ: randomly initialized model, f_θ: randomly initialized classifier, I: number of iteration.
Train G_θ by using NT-Xent Loss on training set D for e_{ssl} epoch.
for $i = 1, 2, \ldots, I$ **do**
  Obtained feature vectors $Z = [z_j]_{j=1}^n$ by using G_θ.
  Min-Max scaling of features $[\bar{z}_j]_{j=1}^n = \left[\frac{z_j - \min(Z)}{\max(Z) - \min(Z)}\right]_{j=1}^n$, where $\min(Z)$ and $\max(Z)$ are the minimum and maximum values of each feature dimension across all n feature vectors;
  Random initialize classifier f_θ
  Train classifier f_θ on $[\bar{z}_j]_{j=1}^n$
  Get a clean classifier $f_{\theta p}$ by applying Hölder Pruning on classifier f_θ
  Obtain clean robust data set D_{clean} by x, where $x \in D$ and $f_\theta(x) = f_{\theta p}(x)$
  Obtain potential poison data D_{poison} is $D_{poison} := D' \setminus D_{clean}$
  Train G_θ by using Semi-supervised learning Loss mentioned at Appendix 5 by using D_{clean} and D_{poison} for e_{mix} epoches.
end for
Output: A high performance model G_θ

A.5 Related Work

We present a summary of related work in this section. Many of the works listed below have been described at appropriate sections in the main paper. Our objective in this section is to categorize related work into (a) works introducing backdoor attacks and (b) those which focus on design of defense strategies against backdoor attacks.

Backdoor attacks aim to mislead a DNN to exhibit abnormal behavior on samples with a trigger while behaving normally on other samples Min et al. (2023). An adversary carrying out a backdoor attack modifies a small fraction of training samples and assigns these samples to an (adversary-desired) target label Bagdasaryan & Shmatikov (2021); Doan et al. (2021); Gu et al. (2019); Li et al. (2021c); Souri et al. (2022). Backdoor attacks can be (i) patch-based Chen et al. (2017); Gu et al. (2019); Turner et al. (2019), where the inserted trigger takes a form of a patch (e.g., white square in BadNets Gu et al. (2019)), or (ii) non-patch-based, where the trigger exploits properties of the input such as image-size Zhao et al. (2020) and uses techniques such as image warping Nguyen & Tran (2021) or steganography Tancik et al. (2020) to imperceptibly deform the image.

Backdoor defense mechanisms can be categorized broadly into (i) In-training or (ii) Post-training methods. In-training defenses assume that the defender has access to a subset of poisoned data for model training, and subsequently leverage differences in observed behaviors (e.g., magnitudes of loss functions) associated with poisoned and clean samples to mitigate effects of backdoor attacks Gao et al. (2023); Li et al. (2021b); Zhang et al. (2023). Post-training defenses, on the other hand, assume access only to a possibly backdoored DNN model, and generally requires access to additional clean samples to mitigate a backdoor attack Zhu et al. (2023b). Examples of post-training defense methods includes pruning Zhu et al. (2023a), fine-tuning Kumar et al. (2022), and toxin suppression Zeng et al. (2021). We explore efficacy of our Hölder Pruning approach, which is applicable during both the in-training and post-training phases. We compare our method with state-of-the-art defenses in both in-training and post-processing stages to demonstrate its versatility and effectiveness.

A.6 Experimental Results on the CIFAR-100 dataset

In this section, we provide additional experimental results on CIFAR-100 dataset to explore the potential influence of the dataset. Specifically, CIFAR-100 contains only 500 images per class with a large number of categories, a common scenario in real-world classification tasks. To ensure that the number of backdoor instances does not exceed the number of images per class, resulting in unbalanced data, we set the poison rate to 1%.

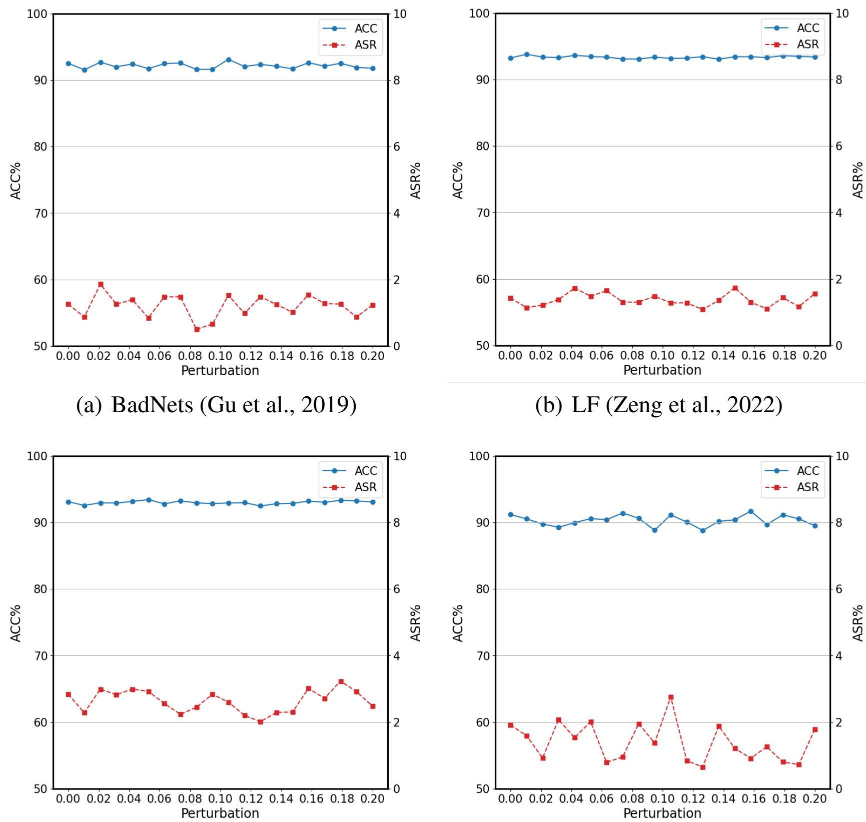
It can be observed that due to the decrease in poison rate, all defense methods have shown improvement. ABL exhibits a greater ability to isolate poisoned samples in their initial isolation step, leading to a reduction of ASR to 0 across all five types of attacks. However, the nature of pruning in CLP, determined by its pruning rate selection, allows the model to maintain a relatively high ASR. The classification-based methods DBD and D-ST also show improvement based on loss, yet they still struggle to address clean label attacks. In this setting, our defense method still performs the best, achieving a 1.5% ASR on HP and remarkable 0% ASR on HID. Additionally, our ACC and RA remain at high levels, with RA significantly surpassing other defense methods.

A.7 Performance Under Different Perturbation Magnitudes

We further investigated the effectiveness of Hölder Pruning under different ϵ values, as shown in Figure 12. It can be observed that the effectiveness of Hölder Pruning remains consistently satisfactory across various ϵ values. As the ϵ value increases, there is a slight decrease in accuracy, highlighting the resilience of Hölder Pruning under different ϵ values. Within the epsilon perturbation range of 0.05 to 0.2, the ASR consistently remains at a low level, while the ACC stays at a high level. This provides valuable guidance for selecting the appropriate perturbation size.

A.8 Hölder Constant Selection

To determine the appropriate value of Hölder constants, we conducted extensive experiments, which revealed that Hölder constants of neurons inside classifier typically exhibit a specific distribution. In Fig. 13, most Hölder constants are close to 0, showing a very high peak, followed by a rapid



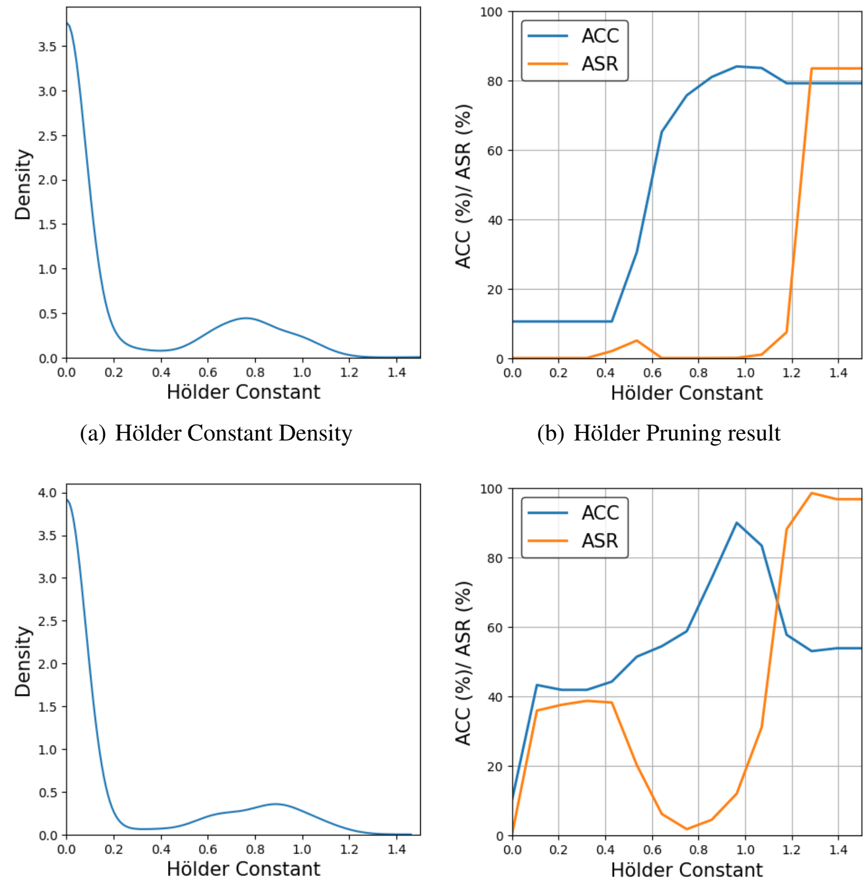
(a) BadNets (Gu et al., 2019)

(b) LF (Zeng et al., 2022)

(c) Input-aware (Nguyen & Tran, 2020)

(d) BppAttack (Wang et al., 2022)

Figure 12: The performance of our proposed Hölder Pruning for different perturbation magnitudes in BadNets (Gu et al., 2019), LF (Zeng et al., 2022), Input-aware (Nguyen & Tran, 2020) and BppAttack (Wang et al., 2022) attacks on the CIFAR-10 dataset. We observe that the ACC and ASR values when using our Hölder Pruning defense are not effected by perturbation magnitude.



(a) Hölder Constant Density

(b) Hölder Pruning result

(c) Hölder Constant Density

(d) Hölder Pruning result

Figure 13: This figure presents the probability distribution functions (PDFs) of Hölder Constant values and the performance of our Hölder Pruning in Hölder Iteration Defense against BadNets (Gu et al., 2019) (Fig. (a) and Fig. (b)) and SSBA (Li et al., 2021d) (Fig. (c) and Fig. (d)) attacks on the CIFAR-10 dataset. The PDFs in Fig. (a) and Fig. (c) illustrate that the majority of neurons in the hidden layer of the classifier exhibit low Hölder Constant values, indicating minimal sensitivity to perturbations, indicative of clean neurons for BadNets and SSBA attacks. Conversely, neurons with larger Hölder Constant values exhibit high sensitivity to perturbations, indicative of backdoored neurons for both BadNets and SSBA attacks. The neurons with high Hölder constant values are targeted for removal by the Hölder Pruning method proposed in this paper. Figures (b) and Fig. (d) demonstrate that removing neurons with high Hölder Constant values results in a low Attack Success Rate (ASR) while maintaining high Classification Accuracy (ACC).

Table 6: *Evaluation of Hölder Pruning (HP) against other SOTA Pruning Methods*: The classification accuracy (ACC) for clean samples, attack success rate (ASR) and robust accuracy (RA) for Trojan samples for nine different attacks–BadNets (Gu et al., 2019), LC (Turner et al., 2019), SIG (Barni et al., 2019), LF (Zeng et al., 2022), WaNet (Nguyen & Tran, 2021), Input-aware(Nguyen & Tran, 2020), SSBA(Li et al., 2021d), Trojan (Liu et al., 2018b), BppAttack (Wang et al., 2022) on five different defense methods using transformer CLIP (Radford et al., 2021) as feature extractor with 1% poison rate (Note: LC (Turner et al., 2019), SIG (Barni et al., 2019) used 0.5%). No additional clean data in use.

Methods→	Benign			FP			ANP			CLP			FMP			**HP(ours)**		
Attacks↓	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑
BadNets	79.1	46.2	-	52.0	20.9	40.7	37.1	60.6	13.2	**77.1**	13.2	68.4	72.1	34.5	48.5	74.1	**1.7**	**72.8**
LC	78.2	12.9	-	59.8	2.1	52.4	60.9	1.5	54.6	**77.1**	4.8	69.2	69.8	**0.0**	61.9	76.4	**0.0**	**73.7**
SIG	79.0	39.6	-	56.6	15.7	17.1	57.8	3.5	23.4	77.3	29.1	26.7	74.2	11.7	25.8	**77.7**	**0.3**	**50.7**
LF	78.5	89.1	-	47.1	40.4	31.9	16.2	98.8	1.1	**77.1**	1.2	**71.9**	72.4	15.7	59.5	75.2	**0.0**	71.8
WaNet	78.5	0.5	-	59.6	**0.0**	47.5	63.1	40.1	0.3	73.2	**0.0**	60.1	65.1	**0.0**	44.2	**76.1**	**0.0**	**72.1**
Input-aware	79.1	80.1	-	55.7	73.3	16.9	54.0	94.8	3.7	78.0	58.9	32.0	71.5	85.2	11.5	**78.9**	**4.4**	**68.1**
SSBA	79.0	68.4	-	52.5	12.2	34.6	58.0	2.9	40.6	77.7	51.6	36.6	72.3	83.3	12.5	**78.9**	**0.0**	**66.9**
Trojan	78.9	95.5	-	56.4	93.9	4.3	57.9	46.2	24.1	**77.6**	94.3	4.9	71.5	93.8	4.8	76.6	**2.5**	**64.2**
BppAttack	78.3	96.1	-	59.1	77.7	9.2	35.5	100.0	0.0	**77.7**	86.3	10.8	69.9	97.8	1.2	76.4	**4.6**	**50.7**
Averages	78.7	58.7	-	55.4	37.3	28.2	48.9	49.8	17.8	**76.9**	37.7	42.2	70.9	46.8	29.9	76.7	**1.5**	**65.7**

CIFAR-100

Table 7: *Evaluation of Hölder Iteration Defense (HID) against other SOTA End-to-End Backdoor Defenses*: The classification accuracy (ACC) for clean samples, attack success rate (ASR) and robust accuracy (RA) for Trojan samples for nine different attacks–BadNets (Gu et al., 2019), LC (Turner et al., 2019), SIG (Barni et al., 2019), LF (Zeng et al., 2022), WaNet (Nguyen & Tran, 2021), Input-aware (Nguyen & Tran, 2020), SSBA (Li et al., 2021d), Trojan (Liu et al., 2018b), BppAttack (Wang et al., 2022) on five different defense (AB L(Li et al., 2021b), CLP (Zheng et al., 2022), DBD (Huang et al., 2022), D-ST (Chen et al., 2022)) method with 1% poison rate (Note: LC (Turner et al., 2019), SIG (Barni et al., 2019) used 0.5%).

Methods→	Benign			ABL			CLP			DBD			D-ST			**HID(ours)**		
Attacks↓	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑
BadNets	65.3	99.1	-	61.3	**0.0**	62.0	54.4	36.4	39.4	62.0	**0.0**	62.3	53.9	**0.0**	53.1	**63.1**	**0.0**	**62.6**
LC	66.3	99.8	-	60.6	**0.0**	58.5	**65.3**	40.7	40.1	64.8	92.2	6.4	55.1	16.1	45.8	60.3	**0.0**	**59.9**
SIG	66.5	98.9	-	57.8	**0.0**	15.6	**63.1**	44.1	18.8	62.3	85.6	10.4	54.8	65.1	7.8	59.8	**0.0**	**58.1**
LF	66.1	99.2	-	55.8	18.3	41.1	**65.1**	44.6	38.7	58.1	1.3	54.5	52.3	2.7	49.7	63.3	**0.0**	**61.5**
WaNet	65.6	98.6	-	64.0	2.7	55.7	61.2	35.4	41.3	61.7	**0.0**	58.9	54.4	0.1	53.5	**64.1**	**0.0**	**60.5**
Input-aware	65.1	99.3	-	56.3	39.9	30.3	64.1	15.9	50.0	62.4	**0.0**	59.5	54.9	46.8	31.9	**64.2**	**0.0**	**60.3**
SSBA	66.1	99.9	-	59.7	**0.0**	53.8	49.1	95.3	3.4	63.1	**0.0**	48.0	52.7	68.9	23.1	**63.2**	**0.0**	**63.0**
Trojan	65.6	99.9	-	57.3	**0.0**	42.5	33.3	76.4	6.0	**64.6**	**0.0**	59.9	53.6	95.0	3.9	64.1	**0.0**	**63.2**
BppAttack	65.6	97.3	-	**63.5**	13.5	52.5	57.6	**0.0**	50.0	63.4	**0.0**	58.2	53.9	0.1	53.1	62.2	**0.0**	**58.6**
Averages	65.8	99.1	-	59.5	8.2	45.7	57.0	43.2	31.9	62.4	19.9	46.4	53.9	32.7	35.8	**62.7**	**0.0**	**60.9**

CIFAR-100

decline in frequency. Subsequently, there exists a range with relatively high frequency, after which the frequency significantly decreases.

We carried out experiments which indicated that poisoned neurons are typically linked to larger Hölder constants. Therefore, we opted to prune neurons with Hölder constants exceeding a certain threshold value. This threshold is a tunable parameter, whose value is selected by analyzing the density distribution of Hölder constants. We observe that within this high-frequency range, the Attack Success Rate (ASR) significantly decreases, while the Accuracy (ACC) remains high. If the Hölder constant exceeds this range, the ASR remains high; if it falls below this range, the model's ACC decreases.

This information is crucial for selecting the optimal values of Hölder constants to achieve the best pruning results. Consequently, we propose setting the Hölder constant within this high-frequency range to ensure high ACC and low ASR.

A.9 Comparison with other Baselines

We compared our defense method with four other model repair methods, namely Neural Cleanse (NC) (Wang et al., 2019), Adversarial unlearning of backdoors via implicit hypergradient (i-BAU) (Zeng et al., 2021), D-BR (Chen et al., 2022), and Shared Adversarial Unlearning: (SAU) (Wei et al., 2023).

Notably, these methods require the defender to possess an extra benign dataset. To ensure a fair comparison, we provided these methods with an additional 5% of clean data. As shown in Tables 1-2, as expected, NC, i-BAU, and SAU were able to maintain high accuracy due to the supplementary information from the benign local dataset. However, these model repair methods share a common issue: during the removal of malicious information, it is challenging to entirely separate clean from poisoned features within the neurons, resulting in their recovery accuracy (RA) being significantly lower than that of our method. In contrast, our method achieved the lowest attack success rate (ASR) and the highest RA in nearly all cases, while its accuracy (ACC) was either the highest or the second highest. These results further validate the effectiveness and advantages of our approach, which does not require additional clean datasets and yet achieves comparable accuracy, the lowest ASR, and the highest RA.

Table 8: *Evaluation of Hölder Iteration Defense (HID) against other SOTA End-to-End Backdoor Defenses*: Classification accuracy (ACC) for clean samples, attack success rate (ASR) and robust accuracy (RA) for Trojan samples for nine different attacks–BadNets (Gu et al., 2019), LC (Turner et al., 2019), SIG (Barni et al., 2019), LF (Zeng et al., 2022), WaNet (Nguyen & Tran, 2021), Input-aware (Nguyen & Tran, 2020), SSBA (Li et al., 2021d), Trojan (Liu et al., 2018b), BppAttack (Wang et al., 2022)– on five defenses– NC (Wang et al., 2019), i-BAU (Zeng et al., 2021), D-BR (Chen et al., 2022), SAU (Wei et al., 2023)– with 5% poison rate (LC (Turner et al., 2019), SIG (Barni et al., 2019) used 0.5% on GTSRB). Our HID consistently outperforms the SOTA.

Methods→	Benign			NC			i-BAU			D-BR			SAU			**HID(ours)**		
Attacks↓	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑
BadNets	91.9	98.7	-	**91.5**	3.2	89.5	88.4	**1.3**	87.7	10.0	1.5	10.7	91.1	**1.3**	90.4	90.9	1.5	**90.6**
LC	93.3	99.5	-	**92.2**	93.4	6.6	88.8	6.6	84.2	11.1	**0.0**	12.8	91.0	7.9	81.2	90.7	3.9	**88.6**
SIG	93.6	97.1	-	**93.4**	96.4	3.5	88.5	0.8	48.3	10.5	**0.0**	11.3	91.5	**0.0**	50.3	88.2	0.1	**89.8**
LF	93.3	98.0	-	**92.0**	62.1	33.9	87.5	15.3	60.4	10.5	82.3	2.8	90.8	**2.7**	83.7	88.6	3.7	**88.1**
WaNet	92.7	85.5	-	91.0	95.1	4.7	89.7	16.1	70.7	66.0	74.4	13.7	91.1	4.5	85.1	**92.1**	**1.3**	**91.0**
Input-aware	91.5	90.2	-	**92.8**	67.8	31.1	88.2	3.4	79.1	81.2	81.3	16.9	91.8	**0.9**	85.2	89.4	4.7	**86.1**
SSBA	93.2	94.9	-	**93.7**	99.8	0.0	90.3	6.6	58.4	15.1	63.8	5.7	90.9	11.0	57.8	89.5	**4.4**	**82.4**
Trojan	93.8	99.9	-	**93.3**	**0.0**	80.4	89.0	4.9	68.6	11.7	**0.0**	13.0	91.3	0.9	**85.7**	91.1	7.5	83.6
BppAttack	91.6	99.9	-	**92.9**	3.1	85.5	91.1	9.3	55.5	81.9	93.0	6.1	91.6	2.3	85.5	89.4	**0.6**	**89.0**
Averages	92.7	96.0	-	**92.5**	57.8	37.2	89.0	7.1	68.1	33.1	44.0	10.3	91.2	3.6	78.3	89.9↑	**3.0↓**	**87.7↑**

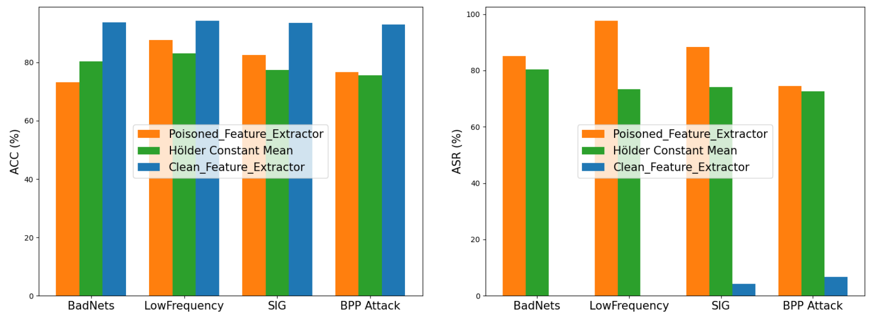
CIFAR-10

Methods→	Benign			NC			i-BAU			D-BR			SAU			**HID(ours)**		
Attacks↓	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑	ACC↑	ASR↓	RA↑
BadNets 97.3	57.9	-	94.4	**0.0**	94.4	96.8	**0.0**	96.8	11.2	**0.0**	11.3	97.8	**0.0**	97.7	**98.8**	**0.0**	**98.8**	
LC	97.8	53.1	-	98.3	**0.0**	98.2	96.3	**0.0**	96.5	10.6	**0.0**	10.8	96.9	**0.0**	97.0	**98.4**	**0.0**	**98.4**
SIG	98.5	66.7	-	**98.3**	74.4	19.5	96.7	8.6	34.3	19.5	**0.0**	5.8	97.8	0.8	27.9	96.4	6.1	**85.5**
LF	98.4	98.6	-	92.2	1.7	32.4	94.5	15.5	24.2	4.4	78.8	1.9	96.0	**0.2**	9.4	**96.2**	3.2	**90.0**
WaNet	98.4	92.9	-	96.4	**0.0**	95.7	98.1	**0.0**	97.5	73.0	81.2	14.5	97.5	0.9	96.7	**98.7**	**0.0**	**98.7**
Input-aware	98.2	92.8	-	92.6	92.3	7.4	97.0	0.4	95.4	5.3	32.0	5.3	98.7	**0.0**	93.9	**99.3**	**0.0**	**99.1**
SSBA	98.1	99.3	-	**98.0**	92.3	7.4	94.3	7.1	78.4	10.8	**0.0**	10.6	81.1	90.9	8.0	95.4	5.7	**92.4**
Trojan	98.5	100.0	-	93.3	**0.0**	80.4	**95.6**	0.1	78.8	21.8	**0.0**	14.9	85.7	**0.0**	12.2	94.3	8.0	**91.4**
BppAttack	98.2	98.9	-	96.5	0.5	82.2	97.5	1.9	90.1	41.8	77.2	7.8	**98.0**	**0.0**	**97.7**	97.8	8.4	89.1
Averages	98.1	84.5	-	95.5	29.0	57.5	96.3	3.7	76.8	22.0	29.9	9.2	94.3	10.3	60.1	**97.3↑**	**3.4↓**	**93.7↑**

GTSRB

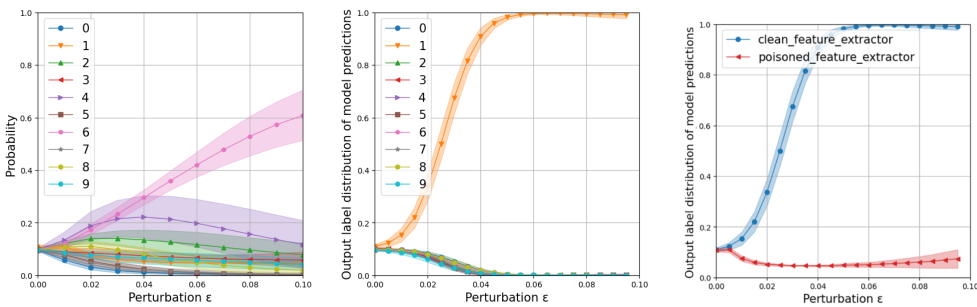
A.10 Additional ablation study for Hölder Pruning

Figure 14 presents the results of an ablation study on our Hölder Pruning defense method. We evaluate the necessity of each component through the following experiments: Clean Feature Extractor (Part 1) and comparing the use of the maximum Hölder value versus the average Hölder value across all training data for measuring the sensitivity of neurons (Part 2). Our results indicate that (a) the Clean Feature Extractor significantly reduces the Attack Success Rate (ASR). For the backdoored model with a poisoned feature extractor, the model's output under perturbation is quite interesting. As shown in Figure 15, even when the target class is 1, the backdoored model tends to predict label 6, which implies that the poisoned neurons are less sensitive to perturbation than the clean neurons. (b) The average Hölder value is not an effective metric for identifying poisoned neurons. We consider that this is due to the presence of latent poisoned neurons that are generally insensitive to perturbations but activate under specific conditions. This characteristic aligns with the facts of backdoored data, where only a small number of poison samples are dispersed throughout the entire training set.



(a) ACC (b) ASR

Figure 14: This figure shows higher ACC values and lower ASR values of our Hölder Pruning under different conditions (blue color bars). Poisoned_Feature_Extractor means we use a poisoned feature extractor with Hölder Pruning, Hölder Constant Mean indicates we use the mean Hölder Constants for each neurons to evaluate the sensitivity. Clean_Feature_Extractor is the approach used in our Hölder Pruning.



(a) (b) (c)

Figure 15: This figure compares effects of perturbing inputs on output labels of backdoored models without a clean feature extractor ((a)) and with a clean feature extractor ((b)). We observe that models without a clean feature extractor are more likely to output label 6 in the presence of perturbations, whereas label 1 is the target label. This indicates that in models without a clean feature extractor, clean neurons are more sensitive than toxic neurons. Fig. (c) compares probabilities of predicting the target class between (a) and (b). Models with a clean feature extractor significantly increase sensitivity to perturbations, demonstrating that compressing toxins into the classifier layer enhances the features of toxic neurons.

A.11 Perturbation for Backdoored model

For a backdoored model without a clean feature extractor, Fig. 15 show that they are less sensitive to perturbations compared to models with a clean feature extractor. Additionally, we observe that backdoored models without a clean feature extractor tend to output label 6 under perturbation, whereas the target label is 1. This indicates that the toxic neurons in such models are less sensitive to perturbations than the clean neurons, making it more challenging to identify the toxic neurons. In contrast, models with a clean feature extractor exhibit significantly enhanced sensitivity of toxic neurons to perturbations, greatly surpassing that of models without a clean feature extractor.

A.12 Dataset Information

We conduct experiments on three benchmark datasets, CIFAR-10, CIFAR-100, and GTSRB. CIFAR-10 contains 60,000 images divided into 10 classes, with 6,000 images per class. CIFAR-100 contains 60,000 images divided into 100 classes, with 600 images per class. GTSRB contains 51,839 images divided into 43 classes, with varying numbers of images per class.

A.13 Training Settings

The machine used for these experiments was an NVIDIA GeForce RTX 3090. We conducted experiments using PreActResNet18 as the base model.

- For the proposed Hölder Pruning method, we trained the classifier for 100 epochs using the training matrix extracted by the clean feature extractor. The hidden layer size of the classifier was set to 1024. We used Cross Entropy Loss and the Adam optimizer, with a learning rate of 0.001.
- For the proposed Hölder Iteration Defense, we trained the feature extractor using NT-Xent Loss for 100 epochs. Inside the self-supervised learning, we used the SGD optimizer to optimize the feature extractor, with a learning rate of 0.4, a weight decay of 0.0001, and momentum set to 0.9. We set the number of iterations to 4 for the semi-supervised learning, and the optimizer is setting as Adam with learning rate of 0.002. During each iteration, we trained the classifier for 60 epochs using the training matrix extracted by the generated feature extractor. The training epochs for semi-supervised learning were set to 20. The Hölder Pruning settings are the same as above. Since the performance of the feature extractor will also affect the results of Hölder Pruning, the performance of the feature extractor improves with each iteration. We suggest iteratively minimizing the pruning size; specifically, set the pruning size to 512 initially, and decrease it by 128 during each iteration.

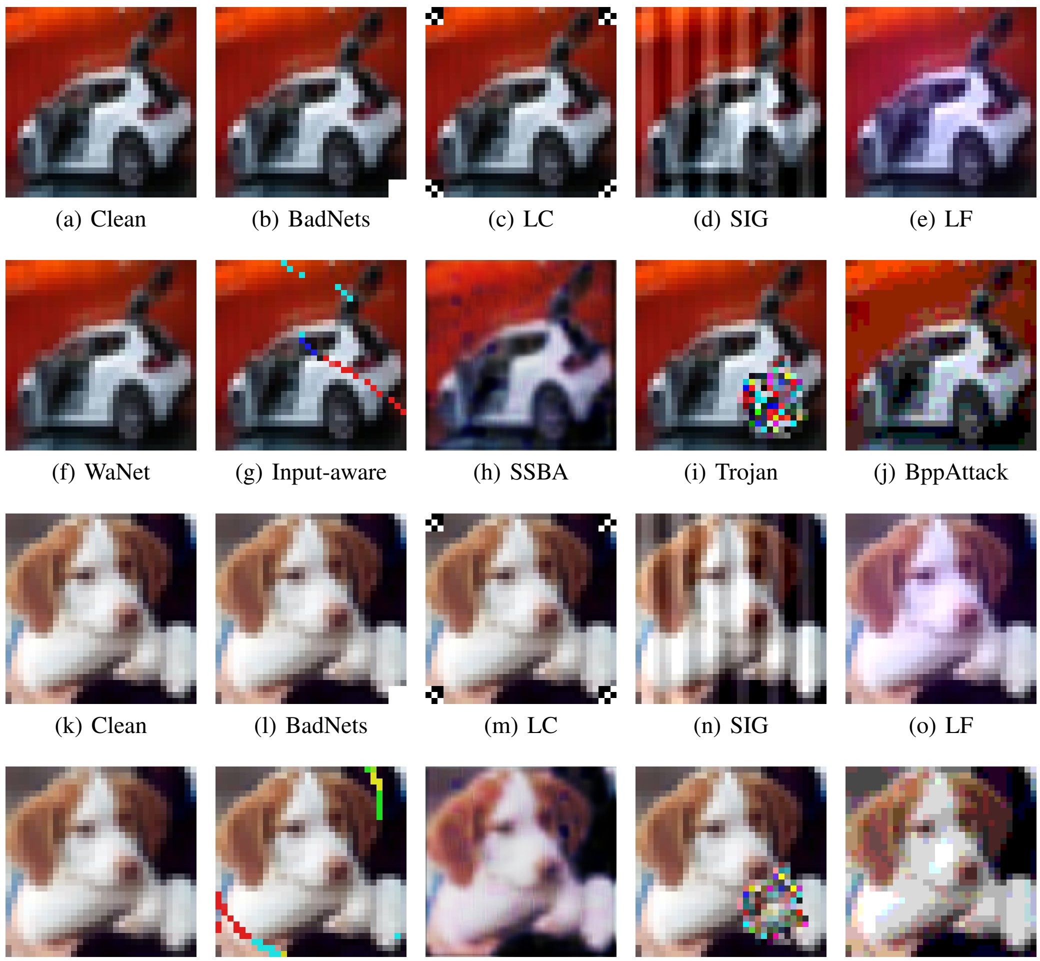
A.14 Attack Settings

In this part, we provide additional implementation details on nine SOTA backdoor attacks.

Table 9: Criteria of nine attacks–BadNets (Gu et al., 2019), LC (Turner et al., 2019), SIG (Barni et al., 2019), LF (Zeng et al., 2022), WaNet (Nguyen & Tran, 2021), SSBA (Li et al., 2021d), Trojan (Liu et al., 2018b), BppAttack (Wang et al., 2022)

Criterion →	Size		Visibility		Variability		Label-consistency		Coverage	
Attack ↓	Patch	Blend	Visible	Invisible	Agnostic	Specific	Dirty	Clean	Local	Global
BadNets	✓		✓		✓		✓		✓	
LC	✓		✓		✓			✓	✓	
SIG		✓	✓		✓			✓		✓
LF		✓		✓		✓	✓			✓
WaNet		✓		✓	✓		✓			✓
Input-aware	✓		✓			✓	✓		✓	
SSBA		✓		✓		✓	✓			✓
Trojan	✓		✓		✓		✓		✓	
BppAttack		✓		✓		✓	✓			✓

- **BadNets (Gu et al., 2019)**: The trigger is a 3×3 white square at the bottom right corner of images as shown in Figure. We attach the trigger to a portion of training samples from other classes and change the label to the target label. We achieve the attack success rate (ASR) of 98.7% and the natural accuracy on clean data (ACC) of 91.9% on CIFAR-10.
- **LC (Turner et al., 2019)**: The trigger is a 3×3 checkerboard at the four corners of images as shown in Figure. To make the backdoored model rely more on the trigger pattern rather than the salient features from the source class, we apply adversarial perturbations to render these poisoned samples harder to classify. We use Projected Gradient Descent (PGD) to generate adversarial perturbations with a maximum perturbation size (ϵ) of 16, each pixel of the image can be altered by up to 16 units in any direction. We achieve an ASR of 99.5% and ACC of 93.3% on CIFAR-10.
- **SIG (Barni et al., 2019)**: Sig attack injects a sinusoidal signal as the trigger over the images. The trigger is embedded to a portion of training samples from other classes and the label is changed to the target label. We achieve an ASR of 97.1% and ACC of 93.6% on CIFAR-10.
- **LF (Zeng et al., 2022)**: Low Frequency Attack manipulates the frequency components of an image by applying a low-pass filter, which reduces high-frequency information while preserving the low-frequency content, thereby embedding a backdoor trigger. The low-pass



(a) Clean (b) BadNets (c) LC (d) SIG (e) LF

(f) WaNet (g) Input-aware (h) SSBA (i) Trojan (j) BppAttack

(k) Clean (l) BadNets (m) LC (n) SIG (o) LF

(p) WaNet (q) Input-aware (r) SSBA (s) Trojan (t) BppAttack

Figure 16: Examples of poisoned CIFAR-10 images including Beign, BadNets (Gu et al., 2019),LC (Turner et al., 2019), SIG (Barni et al., 2019), LF (Zeng et al., 2022), WaNet (Nguyen & Tran, 2021), Input-aware (Nguyen & Tran, 2020), SSBA (Li et al., 2021d), Trojan (Liu et al., 2018b), BppAttack (Wang et al., 2022)

filter reduces details and noise in the image, preserving the primary visual and semantic information. We achieve the attack success rate (ASR) of 98.0% and the natural accuracy on clean data(ACC) of 93.3% on CIFAR-10.

- **WaNet (Nguyen & Tran, 2021)**: We first define a specific warping pattern that creates visually subtle distortions. The parameters include the perturbation strength ($s = 0.5$), the noise grid size ($k = 4$), and the grid rescale factor ($grid\_rescale = 1$), which are used to control the spatial transformation perturbations added to the input images. Then we warp a portion of images from the training set according to the defined warping pattern. We achieve an ASR of 85.5% and ACC of 92.7% on CIFAR-10.
- **Input-aware (Nguyen & Tran, 2020)**: InputAware attack involves analyzing the input data to identify the most suitable locations and forms for trigger embedding, and then generating a trigger that blends seamlessly with the input to ensure it manipulates the model effectively while being invisible to the human eye. The generator is optimized through adversarial training with 500 epoches. We achieve an ASR of 90.2% and ACC of 91.5% on CIFAR-10.
- **SSBA (Li et al., 2021d)**: The SSBA attack utilizes an encoder-decoder network to embed attacker-specified strings into normal training images. This method is grounded in principles of image steganography, where the network learns how to covertly insert information into images without being detected visually. Through this method, each training image is individually modified to contain a unique trigger that only the model can recognize. We achieve an ASR of 93.2% and ACC of 94.9% on CIFAR-10.
- **Trojan (Liu et al., 2018b)**: The TrojanNN attack involves selecting influential neurons. We identify the two most influential neurons in the model's linear layer by analyzing the magnitude of the weights. A trigger is then generated using an encoder-decoder structure, and this trigger is adjusted until the activation values of the two neurons reach 100 or until 1000 epochs are completed. We achieve an ASR of 93.8% and ACC of 99.9% on CIFAR-10.
- **BppAttack (Wang et al., 2022)**: Bits Per Pixel attack is a type of backdoor attack targeting image compression models. This attack subtly modifies the encoder parameters during the image compression process, embedding triggers without significantly degrading image quality. These triggers are injected in the Discrete Cosine Transform (DCT) domain and invisible to the human eye. We achieve the ASR of 99.9% and ACC of 91.6% on CIFAR-10.

A.15 Defense Settings

In this part, we provide additional details on the twelve backdoor defense methodologies that we consider.

Pruning Defense on classifier:

- **Fine-Pruning (Liu et al., 2018a)**: Forward propagation is performed on all of the training data D, and the activation values of neurons in the classifier are recorded and accumulated. Neurons are then sorted based on these activation values, and 1% of neurons or weights with the lowest activation values are pruned each time. The pruned model is tested to ensure that the overall accuracy does not drop more than 10%. If the accuracy drop exceeds 10%, pruning is stopped. The steps of sorting, pruning, testing, and accuracy checking are repeated until the pruning target or the upper limit of accuracy drop is reached.
- **CLP (Zheng et al., 2022)**: In the CLP pruning method, evaluate the Lipschitz Constant on each neruons within a classifier. The pruning process involves calculating the mean and std of each Lipschitz Constant inside the neruons, pruning the neurons which has the Lipschitz Constant larger than mean + std.
- **ANP (Wu & Wang, 2021)**: In the ANP defense process, random perturbations epsilon are first generated in the range of [-0.4, 0.4] and applied to the weights and biases of neurons. These perturbations are optimized using Projected Gradient Descent (PGD) to maximize the classification loss, repeating this process once. Then, the mask values are updated by calculating a loss function that includes the losses of clean and adversarially perturbed data, with a weight of 0.2. The mask values indicate which neurons are most sensitive to adversarial perturbations. The training progress is recorded every 500 iterations until 2000 iterations are completed. Finally, neurons sensitive to adversarial perturbations are pruned

step-by-step using a threshold of 0.90 and a step size of 0.05, reducing the success rate of backdoor attacks while maintaining accuracy on clean data.

- **FMP (Huang & Bu, 2024)**: The FMP (Adversarial Feature Map Pruning) defense process involves several steps: first, generating potential poisoned samples through Feature Reverse Generation (FRG), which includes initializing the input with random noise, calculating the loss based on the classifier hidden layer output, updating the input according to the gradient of the loss, and ensuring the perturbed input remains within valid bounds. Then, feeding these generated poisoned samples into the model to observe the inference accuracy of each classifier hidden layer and identifying backdoor-related feature maps through significant changes in accuracy. Next, pruning these identified backdoor neurons inside the classifier by setting their weights to zero during the forward pass.

Data free defense:

- **ABL (Li et al., 2021b)**: In the ABL backdoor defense process, the model is first pre-trained on the backdoored dataset for 20 epochs using a Flooding loss function (flooding: 0.5). Based on these loss values, 1% of the suspicious data (isolation_ratio: 0.01) is isolated, and the model is further trained on the remaining clean data for 60 epochs. Finally, the model undergoes 20 epochs of unlearning (unlearning_epochs: 20) to forget the backdoor patterns learned from the isolated malicious data.
- **CLP (Zheng et al., 2022)**: In the CLP pruning method, the Channel Lipschitz Constant (CLC) is used to assess and adjust the sensitivity of each channel within a neural network. The pruning process involves calculating the CLC for each channel, and applying pruning if the CLC exceeds the mean plus a multiple (u) of the standard deviation. The intensity of pruning is controlled by the parameter u, which is set to 3. Additionally, the configuration parameters u_{min} : 0, u_{max} : 10, and u_{num} : 20 allow for testing 21 different values of u evenly distributed from 0 to 10. This helps in evaluating the impact of different pruning intensities on model performance to optimize defense effectiveness while maintaining clean model accuracy.
- **DBD (Huang et al., 2022)**: In the DBD defense strategy, self-supervised learning for 100 epochs is the initial phase where the model learns intrinsic features of the data to create feature extractors without relying on labels, and the 'temperature=1' parameter controls the sensitivity of learning. The warmup period for 10 epochs prepares the model for more complex learning tasks. The subsequent semi-supervised learning phase trains the model using both labeled and unlabeled data, employing a confidence threshold set at 0.5 to distinguish between reliable and unreliable data, further optimizing model performance.
- **D-ST (Chen et al., 2022)**: First, train a backdoored model from scratch using a poisoned dataset without any data augmentation. Then, fine-tune the backdoored model with intra-class loss L_intra with the same poisoned dataset. Next, compute the Feature Consistency towards Transformations (FCT) metric for all training samples, calculate thresholds based on the FCT values, and separate the samples into clean (bottom 20%), poisoned (top 5%), and uncertain samples. After that, train a feature extractor using the semi-supervised contrastive learning (SS-CTL) method with 200 epochs using the clean samples and train a classifier by minimizing the mixed cross-entropy loss with 10 epochs using all samples. Finally, combine the model with the feature extractor and classifier.

Additional defense:

- **i-BAU (Zeng et al., 2021)**: In the I-BAU defense process, the model parameters are initialized using the configured random seed and optimizer, and a clean train dataset is loaded. 5% of the clean data is selected for training. In the inner loop, the trigger is updated using gradient ascent to maximize the loss, with up to 5 fixed-point iterations per update. In the outer loop, the model parameters are updated using implicit hypergradient to minimize the loss. This process is repeated for 5 rounds, with the model's performance on clean and backdoor samples evaluated after each round. Finally, the updated model parameters and defense results are saved.
- **NC (Wang et al., 2019)**: The Neural Cleanse defense process begins by initializing parameters and datasets, including using 5% of the training data to train the reverse trigger and

mask. Poisoned classification model is first loaded. The mask and trigger are then initialized and trained to minimize the combined classification loss and mask regularization loss for 80 epochs. After each iteration, the L1 norms of all labels are calculated to detect potential backdoor attacks, and the regularization cost is adjusted based on a patience parameter of 5. If the attack success rate exceeds the threshold of 98.0%, a backdoor attack is flagged. The data is then split, selecting a cleaning subset using 5% of the data (‘cleaning_ratio: 0.05‘). Adversarial samples are generated through reverse learning to fine-tune the model, reducing its sensitivity to backdoor triggers. Finally, the fine-tuned model is evaluated, and the final model and defense results are saved.

- **D-BR (Chen et al., 2022)**: First, train a backdoored model from scratch using a poisoned dataset without any data augmentation. Then, fine-tune the backdoored model with intra-class loss L_{intra} with the same poisoned dataset. Next, compute the Feature Consistency towards Transformations (FCT) metric for all training samples, calculate thresholds based on the FCT values, and separate the samples into clean (bottom 20%), poisoned (top 5%), and uncertain samples. In the two-stage Secure Training (BR) phase, the unlearning process uses gradient ascent on poisoned samples to make the model forget the poisoned features, with a learning rate of 0.0001, batch size of 128, and 20 epochs. The relearning process retrains the model on clean samples using the same parameters. The model's performance is evaluated on clean and poisoned test datasets to measure attack success rate (ASR), accuracy (ACC), and robust accuracy (RC).
- **SAU (Wei et al., 2023)**: defense process involves using a small portion of the clean dataset, specifically 5%, for adversarial training. This training is conducted over 100 rounds, during which PGD attacks are used to generate shared adversarial examples. These examples are constrained by the L_{inf} norm, limiting the perturbation to a range of 0.2, with a step size of 0.2 and iterating 5 steps. During adversarial training, the model parameters are updated by combining adversarial loss weighted at 0.01, shared loss weighted at 1, clean data classification loss weighted at 1, and shared adversarial risk loss weighted at 1. Each round includes one outer optimization step. This process aims to minimize the total loss, thereby effectively mitigating the impact of backdoor attacks while maintaining high accuracy and robustness of the model.