DETECTING BACKDOOR ATTACKS VIA LAYER-WISE FEATURE ANALYSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Training well-performing deep neural networks (DNNs) usually requires massive training data and computational resources, which might not be affordable for some users. For this reason, users may prefer to outsource their training process to a third party or directly exploit publicly available pre-trained models. Unfortunately, doing so opens the possibility of a new dangerous training-time attack (dubbed backdoor attack) against DNNs. Currently, most of the existing backdoor detectors filter poisoned samples based on the latent feature representations generated by convolutional layers. In this paper, we first conduct a layer-wise feature analysis of poisoned and benign samples from the target class. We find out that the feature difference between benign and poisoned samples tends to reach the maximum at a critical layer, which is *not always* the one typically used in existing defenses, namely the layer before fully-connected layers. In particular, we can locate this critical layer easily based on the behaviors of benign samples. Based on this finding, we propose a simple yet effective method to filter poisoned samples by analyzing the feature differences between suspicious and benign samples at the critical layer. We conduct extensive experiments on two benchmark datasets, which confirm the effectiveness of our backdoor detection.

1 INTRODUCTION

Recent years have witnessed the successful application of deep neural networks (DNNs) in many tasks, including computer vision (He et al., 2016), natural language processing (Vaswani et al., 2017), and speech recognition (Farrús, 2018). However, training well-performing DNNs requires massive training data and computational resources, which might not be affordable for some users. Common practices to reduce training costs are to outsource the training process to a third-party (*e.g.*, a cloud service) or directly adopt publicly available pre-trained DNNs.

Unfortunately, these approaches introduce a new dangerous training-time attack, known as *backdoor attack* (BA), against DNNs (Chen et al., 2017; Gu et al., 2019; Li et al., 2022a). In general, the backdoor adversaries poison a few training samples to lead the attacked DNNs into misclassifying samples containing pre-defined trigger patterns as the adversary-specified target label. However, the attacked models behave normally on benign samples. Accordingly, these attacks are stealthy because users can hardly identify them. Since DNNs are used in many mission-critical tasks (*e.g.*, autonomous driving or facial recognition), it is urgent to design backdoor defenses.

Currently, many defense methods (Zeng et al., 2022; Huang et al., 2022; Chen et al., 2022b) have been proposed to alleviate backdoor threats. Among all backdoor defenses, backdoor detection (Chen et al., 2019; Tang et al., 2021; Hayase & Kong, 2021) is one of the most important paradigms, where defenders attempt to detect whether a suspicious object (*e.g.*, model or sample) is malicious or not. Most existing backdoor detectors were designed based on the understanding that poisoned samples should have different feature representations compared to benign ones, since the attacked DNNs have different prediction behaviors for them. We notice that most detectors analyze the features generated by the layer before the fully connected layers. This raises two intriguing questions: **1)** *Is this layer always the most critical place for backdoor detection?* **2)** *If not, how to find the critical layer for designing more effective backdoor detection?*

In this paper, we answer the first question in the negative (see Figure 1). To answer the second question, we conduct a layer-wise feature analysis of poisoned and benign samples from the target

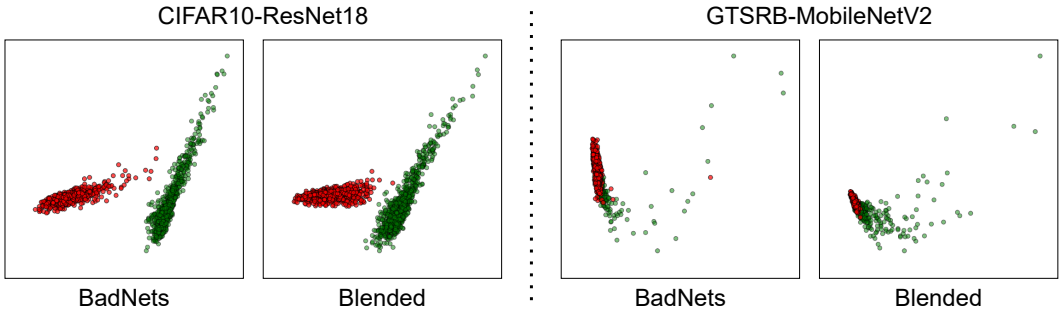


Figure 1: Features of benign (in green) and poisoned samples (in red) generated by the layer before the fully connected layers of ResNet18 and MobileNetV2 DNNs attacked by BadNets (Gu et al., 2019) and Blended (Chen et al., 2017). Visualization is based on principal component analysis (Abdi & Williams, 2010). It can be seen that the features of poisoned and benign samples are not neatly separable on the GTSRB dataset with MobileNetV2.

class. We find out that the feature difference between benign and poisoned samples tends to reach the maximum at a critical layer, which can be easily located based on the behaviors of benign samples. Specifically, this layer is the one or near the one that contributes most to assigning benign samples to their true class. Based on this observation, we propose a simple yet effective method to filter poisoned samples by analyzing the feature differences (measured by cosine similarity) between incoming suspicious samples and a few benign samples at the critical layer. Our method can serve as a 'firewall' for deployed DNNs to identify, block, and trace malicious input samples.

We next summarize our four main contributions. **1)** We demonstrate that the features of poisoned and benign samples are not always clearly separable at the layer before fully connected layers, which is the one typically used in existing defenses. **2)** We conduct a layer-wise feature analysis aimed at locating the critical layer where the separation between poisoned and benign samples is most neat. **3)** We propose a backdoor detection method to filter poisoned samples by analyzing the feature differences between suspicious and benign samples at the critical layer. **4)** We conduct extensive experiments on two benchmark datasets to assess the effectiveness of our proposed defense. We hope that our work can provide more understandings of attack mechanism, to facilitate the design of more effective and efficient backdoor defenses.

2 Related Work

In this paper, we focus on backdoor attacks and defenses in image classification. Other deep learning tasks (*e.g.*, (Xie et al., 2019; Zhai et al., 2021; Chen et al., 2022a)) are out of our current scope.

2.1 Backdoor Attacks

BadNets (Gu et al., 2017; 2019) was the first backdoor attack. It randomly selects a few benign samples and generates their poisoned versions by stamping a trigger patch onto their images and reassigning their label as the target label. Later Chen et al. (2017) noted that the poisoned image should be similar to its benign version for stealthiness; based on that, the authors proposed a blended attack by introducing trigger transparency. However, these attacks are with poisoned labels and therefore users can still identify them by examining the image-label relation. To alleviate this problem, Turner et al. (2019) proposed the clean-label attack paradigm, where the target label is consistent with the ground-truth label of poisoned samples. Specifically, in this paradigm adversarial attacks are exploited to perturb the selected benign samples before conducting the standard trigger injection process. Nguyen & Tran (2020b) adopted image warping as the backdoor trigger, which modifies the whole image while preserving its main content. Besides, Nguyen & Tran (2020a) proposed the first sample-specific attack, where the trigger varies across samples. However, its triggers are visible and the adversaries need to control the whole training process. More recently, Li et al. (2021c) introduced the first poison-only invisible sample-specific attack to address these problems. This attack was inspired by DNN-based image steganography (Tancik et al., 2020).

2.2 BACKDOOR DEFENSES

In general, existing backdoor defenses can be divided into three main categories, including input filtering, input pre-processing, and model repairing, as follows:

Input Filtering. These approaches intend to differentiate benign and poisoned samples based on their distinctive behaviors. One of the typical methods is to filter samples based on the separability of the feature representations of benign and poisoned samples (Chen et al., 2019; Hayase & Kong, 2021; Tang et al., 2021). For example, Hayase & Kong (2021) introduced a robust covariance estimation of feature representations to amplify the spectral signature of poisoned samples. Zeng et al. (2021) proposed to filter inputs based on the finding that poisoned images have some high-frequency artifacts. Gao et al. (2022) proposed to blend various images on the suspicious one, motivated by the understanding that the trigger pattern can still mislead the prediction no matter what the background contents are. Recently, Jin et al. (2022) proposed to filter poisoned samples based on existing techniques in detecting adversarial samples.

Input Pre-processing. These methods pre-process each input sample before feeding it into the deployed DNN. Their motivation is that this process can perturb potential trigger patterns and therefore prevent backdoor activation. Liu et al. (2017) proposed the first pre-processing-based defense where they adopt an encoder-decoder to modify input samples. Rosenfeld et al. (2020) adopted randomized smoothing to generate a set of input neighbors and averaged their predictions. Further, Li et al. (2021b) demonstrated that if the location or appearance of the trigger pattern is slightly different from that of the one used for training, the attack effectiveness may degrade sharply. Based on this finding, they proposed to pre-process images with spatial transformations.

Model Repairing. These defenses aim at erasing backdoors contained in the given attacked DNNs. For example, Liu et al. (2017); Zhao et al. (2020); Li et al. (2021a) revealed that users can effectively remove backdoors by fine-tuning the attacked DNNs with a few benign samples. These methods were inspired by catastrophic forgetting (Kirkpatrick et al., 2017). Liu et al. (2018); Wu & Wang (2021); Zheng et al. (2022) revealed that model pruning can also remove backdoors effectively, because backdoors are mainly encoded in specific neurons. Very recently, Zeng et al. (2022) proposed to repair compromised models with adversarial model unlearning.

In this paper, we focus on input filtering, which can serve as a 'firewall' for deployed DNNs.

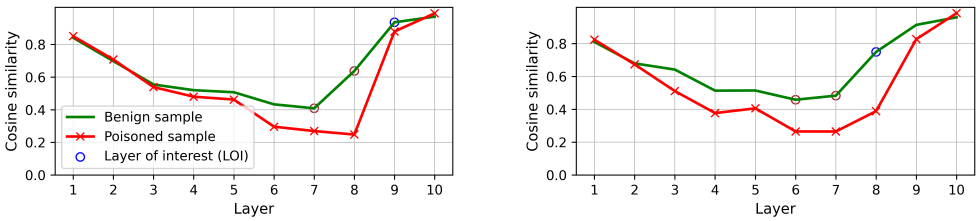
3 LAYER-WISE FEATURE ANALYSIS

3.1 PRELIMINARIES

Deep Neural Networks (DNNs). A deep neural network (DNN) is a function $f(x)$, obtained by composing L functions $f^l, l \in [1, L]$, that maps an input x to a predicted output $\hat{y}$. Each f^l is a layer that is parameterized by a weight matrix w^l, a bias vector b^l, and an activation function σ^l. f^l takes as input the output activation of the previous layer a^{l-1}. The output of f^l on an input a^{l-1} is computed as $a^l = f^l(a^{l-1}) = \sigma^l(w^l \cdot a^{l-1} + b^l)$. In a DNN-based classifier, the first layer f^1 takes x as input, whereas the last layer f^L outputs a vector $a^L \in \mathbb{R}^{\mathcal{C}}$ with $\mathcal{C}$ being the number of classes. The vector a^L is usually fed to the softmax function, which transforms it into a vector p of probabilities, called confidence scores. Moreover, a DNN-based classifier can be divided into two main parts: the *feature extractor* and the *fully connected layers*. The feature extractor takes the original input x and produces a latent representation ($i.e.$, latent features), which is passed on to the fully connected layers to make the final classification decision.

Given a training dataset $D_{train} = \{(x_i, y_i)\}_{i=1}^n$ and a loss function $\mathcal{L}$, the parameters of weights and biases of a DNN $f(\cdot)$ are learned to approximate the optimal (w^*, b^*) which minimizes $\mathcal{L}$ on D_{train}. In classification problems, $\mathcal{L}$ is usually the cross-entropy loss.

In this paper, we use DNNs as $\mathcal{C}$-class classifiers, where $y_i \in \{1, \ldots, \mathcal{C}\}$ is the ground truth of x_i, and its final predicted label $\hat{y}_i$ is the index of the highest confidence score in p_i. Also, we analyze the activations of the intermediate layers ($i.e.$, intermediate features) to detect poisoned samples.



(a) BadNets (b) ISSBA

Figure 2: Layer-wise behaviors of benign samples from the target class and poisoned samples (generated by BadNets and ISSBA attacks) on the CIFAR-10 dataset with ResNet-18

3.2 Analyzing Backdoor Attacks from a Layer-wise Perspective

We notice that the predictions of attacked DNNs for both benign samples from the target class and poisoned samples are all the target label. The attacked DNNs mainly exploit class-relevant features to predict these benign samples while they use trigger-related features for poisoned samples. We suggest that defenders could exploit this difference to design effective backdoor detection. To explore their main differences, we conduct a layer-wise analysis, as follows.

Definition 1 (Layer-wise Centroids of Target Class Features)**.** *Let f' be an attacked DNN with a target class t. Let $X_t = \{x_i\}_{i=1}^{|X_t|}$ be the set of benign samples with true class t, and let $\{a_i^1, \ldots, a_i^L\}_{i=1}^{|X_t|}$ be their intermediate features generated by f'. The centroid of t's benign features at layer l is defined as $\hat{a}_t^l = \frac{1}{|X_t|}\sum_{i=1}^{|X_t|} a_i^l$, and $\{\hat{a}_t^1, \ldots, \hat{a}_t^L\}$ is the set of layer-wise centroids of t's benign features.*

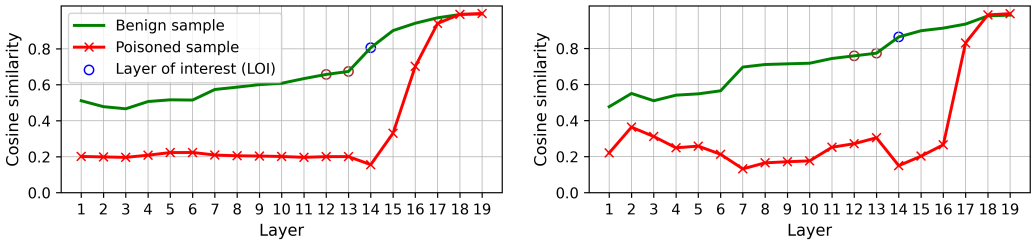
Definition 2 (Layer-wise Cosine Similarity)**.** *Let a_j^l be the features generated by layer l for an input x_j, and let cs_j^l be the cosine similarity between a_j^l and the corresponding t's centroid $\hat{a}_t^l$. The set $\{cs_j^1, \ldots, cs_j^L\}$ is said to be the layer-wise cosine similarities between x_j and t's centroids.*

Settings. We conducted six representative attacks on the following four benchmarks: CIFAR10-ResNet18, CIFAR10-MobileNetV2, GTSRB-ResNet18, and GTSRB-MobileNetV2. The six attacks were BadNets (Gu et al., 2019), the BA with blended strategy of Chen et al. (2017), the label-consistent BA (LC) of Turner et al. (2019), WaNet (Nguyen & Tran, 2020b), the invisible sample-specific BA (ISSBA) (Li et al., 2021c), and the input-aware dynamic BA (IAD) (Nguyen & Tran, 2020a). More details on the datasets, DNNs, and attack settings are presented in Section 5.1. Specifically, for each attacked DNN f' with a target class t, we estimated $\{\hat{a}_t^1, \ldots, \hat{a}_t^L\}$ using 10% of the benign test samples labeled as t. Then, for the benign and poisoned test samples classified by f' into t, we calculated the layer-wise cosine similarities between their generated features and the corresponding estimated centroids. Finally, we visualized the layer-wise means of the computed cosine similarities of the benign and poisoned samples to analyze their behaviors.

Results. Figure 2 shows the layer-wise means of cosine similarity for benign and poisoned samples with the CIFAR10-ResNet18 benchmark under the BadNets and ISSBA attacks. As we go deeper into the attacked DNN layers, the gap between the direction of benign and poisoned features gets larger until we reach a specific layer where the backdoor trigger is activated, causing poisoned samples to get closer to the target class. Figure 3 shows the same for the GTSRB-MobileNetV2 benchmark. Further, we can see that for BadNets the latent features of benign and poisoned samples are similar in the last layer of the features extractor (layer 17).

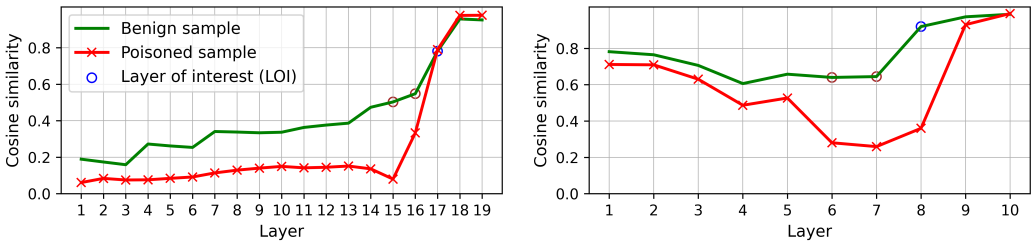
Regardless of the attack or benchmark, when we enter the latter layers of DNNs (which usually are class-specific), *benign samples start to get closer to the target class before the poisoned ones, that are still farther from the target class* because the backdoor trigger is not yet activated. This makes the difference in similarity maximum in one of those latter layers, which we call the *critical layer*. In particular, *this layer is not always the one typically used in existing defenses* (*i.e.*, the layer before fully-connected layers). Besides, we show that it is very likely to be either the layer that contributes most to assigning the benign samples to their true target class (which we name the *layer of interest or LOI*, circled in blue) or one of the two layers before the LOI (circled in brown).

Results under other attacks for these benchmarks are presented in Appendix A.



(a) BadNets (b) ISSBA

Figure 3: Layer-wise behaviors of benign samples from the target class and poisoned samples (generated by BadNets and ISSBA attacks) on the GTSRB dataset with MobileNetV2



(a) CIFAR10-MobileNetV2 (b) GTSRB-ResNet18

Figure 4: Layer-wise behavior of benign and poisoned samples under BadNets w.r.t. the target class in the CIFAR10-MobileNetV2 and GTSRB-ResNet18 benchmarks

To confirm that the above distinctive behaviors hold regardless of the datasets or models being used, we also perform the BadNet attack on the CIFAR10-MobileNetV2 and GTSRB-ResNet18 benchmarks. Figure 4 provides this confirmation.

From the analysis above, we can conclude that focusing on those circled layers can help develop a simple and robust defense against backdoor attacks.

4 Methodology

4.1 Threat Model

We consider a scenario where a user obtains a trained DNN that might have been attacked. For example, a backdoor might have been inserted during training of the DNN on a third-party's side (*e.g.*, a cloud service) or added post-training by an adversary before being downloaded by the user. We assume that the user has limited computational resources or benign samples, and therefore cannot repair the suspicious model. The user wants to defend herself by detecting at inference time whether a suspicious incoming input x_s is poisoned, given the trained model f_s.

Specifically, we consider each class in the set $\{1, \ldots, \mathcal{C}\}$ to be a potential target class. Similar to existing defenses, we assume that a small set of benign samples X_{val} is available to the user/defender. We denote the available samples that belong to a potential class t as $X_{t_{val}}$. For simplicity of notation, let $m = |X_{t_{val}}|$ denote the number of available samples labeled as t.

4.2 The Design of Our Defense

According to the lessons learned in Section 3, our method to detect poisoned samples at inference time consists of the following four main steps. **1)** Estimate the layer-wise features' centroids of class t from the middle layer upward using the class's available benign samples. **2)** Compute the

cosine similarities between the extracted features and the estimated centroids, and then compute the layer-wise means of the computed cosine similarities. **3)** Identify the layer of interest (LOI), sum up the cosine similarities in LOI and the two layers before LOI (sample-wise), and compute the mean and standard deviation of the summed cosine similarities. **4)** For any suspicious incoming input x_s classified as t by f_s, i) compute its cosine similarities to the estimated centroids in the above-mentioned three layers, and ii) consider it as a potentially poisoned input if its summed similarities fall below the obtained mean by a specific number of standard deviations.

We now describe the details of our method. For each potential target class $t \in \{1, \dots \mathcal{C}\}$, we first feed the available m benign samples to f_s and extract their intermediate features in the second half of layers to obtain the set $\{(a_i^{L/2}, \dots, a_i^L)\}_{i=1}^m$ (if L is odd, take the integer part of $L/2$ instead of $L/2$ here and in what follows). Note that we can reduce computation by focusing on the second half of layers because the LOI and the two layers before the LOI are among the latter layers of the DNN (see Section 3). After that, we compute the layer-wise centroids of the extracted features for each layer $l \in \{L/2, \dots, L\}$, as follows:

$$\hat{a}_t^l = \frac{1}{m} \sum_{i=1}^{m} a_i^l. \tag{1}$$

Then, we compute the cosine similarity between the benign features of each layer and their corresponding centroid. That is, for layer $l \in \{L/2, (L/2)+1, \dots, L-1, L\}$, we compute the cosine similarity between the features of each sample a_i^l and its centroid $\hat{a}_t^l$ as follows:

$$cs_i^l = CS(a_i^l, \hat{a}_t^l) = \frac{a_i^l \cdot \hat{a}_t^l}{||a_i^l|| \cdot ||\hat{a}_t^l||}. \tag{2}$$

Then, we aggregate the computed similarities to approximate the similarity centroid in each layer $l \in \{L/2, (L/2)+1, \dots, L-1, L\}$, as:

$$\hat{cs}_t^l = \frac{1}{m} \sum_{i=1}^{m} cs_i^l. \tag{3}$$

Next, we use $\{\hat{cs}_t^{L/2}, \hat{cs}_t^{(L/2)+1}, \dots, \hat{cs}_t^L\}$, to locate the layer of interest LOI_t that contributes most to assigning t's benign samples to their true class t. To that end, we first compute the absolute difference between the approximated similarity of each layer and its preceding one, as follows:

$$\hat{cs}_{t_{diff}} = \{|\hat{cs}_t^{(L/2)+1} - \hat{cs}_t^{L/2}|, \dots, |\hat{cs}_t^L - \hat{cs}_t^{L-1}|\}. \tag{4}$$

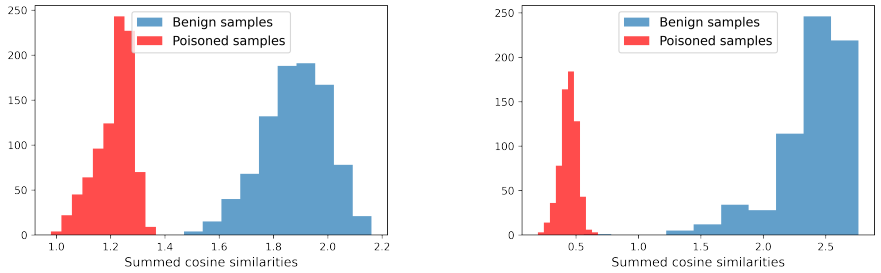
Then, we identify LOI_t as the layer for which the difference in $\hat{cs}_{t_{diff}}$ is maximum. For example, if the maximum difference is $|\hat{cs}_t^l - \hat{cs}_t^{l-1}|$, then layer l is the layer of interest. Once we locate LOI_t, we estimate the behavior of benign samples in that layer and in the two layers previous to it. For each sample $x_i \in X_{t_{val}}$, we sum up its computed cosine similarities in the three layers:

$$cs_i = cs_i^{LOI_t-2} + cs_i^{LOI_t-1} + cs_i^{LOI_t}. \tag{5}$$

After computing the summed similarities of the m samples and obtaining the set $\{cs_i\}_{i=1}^m$, we compute the mean μ_t and the standard deviation σ_t of the set.

To detect potentially poisoned samples, for any suspicious incoming input x_s classified as t by f_s at inference time, we extract its features in LOI_t and the two preceding layers, compute their cosine similarities to the corresponding estimated centroids $\{cs_s^{LOI_t-2}, cs_s^{LOI_t-1}, cs_s^{LOI_t}\}$, and sum them up to get cs_s. Then, we identify x_s as a potentially poisoned sample if $cs_s < \mu_t - \tau \times \sigma_t$, where τ is a threshold chosen by the defender that provides a reasonable trade-off between the true positive rate TPR and the false positive rate FPR. Figure 5 shows an example of the distributions of the summed cosine similarities of benign and poisoned features to the estimated benign centroids (in the three identified layers) under the label-consistent attack of Turner et al. (2019).

The pseudocode of our method is given in Appendix B.



(a) CIFAR10-ResNet18 (b) GTSRB-MobileNetV2

Figure 5: Distributions of the summed cosine similarities of benign and poisoned samples under the label-consistent attack on CIFAR10 with ResNet18 and GTSRB with MobileNetV2 benchmarks

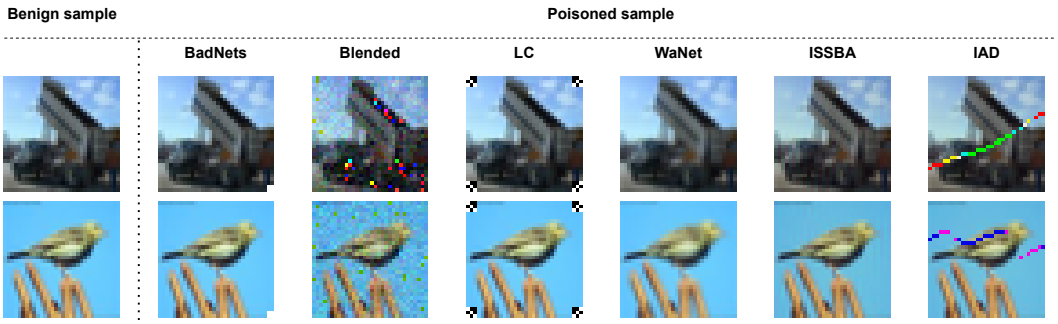


Figure 6: Example of benign samples and their poisoned versions generated by different attacks

5 Experiments

5.1 Main Settings

Datasets and DNNs. In this paper, we use two classic benchmark datasets, namely CIFAR10 (Krizhevsky et al., 2009) and GTSRB (Stallkamp et al., 2011). We use the ResNet18 (He et al., 2016) architecture on CIFAR10 and the MobileNetV2 (Sandler et al., 2018) architecture on GTSRB. More details are presented in Appendix E.2.

Attack Baselines. We evaluate each defense under the six attacks mentioned in Section 3.2: BadNets, Blended, LC, WaNet, ISSBA and IAD. They are representative of visible attacks, patch-based invisible attacks, clean-label attacks, non-patch-based invisible attacks, invisible sample-specific attacks, and visible sample-specific attacks, respectively.

Defense Baselines. We compare our defense with six representative defenses, namely randomized smoothing (RS) (Rosenfeld et al., 2020), ShrinkPad (ShPd) (Li et al., 2021b), activation clustering (AC) (Chen et al., 2019), STRIP (Gao et al., 2022), SCAn (Tang et al., 2021), and fine-pruning (FP) (Liu et al., 2018). RS and ShPd are two defenses with input pre-processing; AC, STRIP, and SCAn are three advanced input-filtering-based defenses; FP is based on model repairing.

Attack Setup. For both CIFAR10 and GTSRB, we adopt the following settings. We use a 2×2 square as the trigger pattern for BadNets, as suggested in Gu et al. (2019); Wang et al. (2019). For Blended, we adopt the random noise pattern, with a 10% blended ratio, as suggested by Chen et al. (2017). The trigger pattern adopted for the LC attack is the same used in BadNets. For WaNet, ISSBA, and IAD, we take their default settings. Besides, we set the poisoning rate to 5% for BadNets, Blended, LC, and ISSBA. For WaNet and IAD, we set the poisoning rate to 10%, following their original settings. More details on settings are given in Appendix E.3. Figure 6 shows an example of poisoned samples generated by different attacks.

Table 1: Main results (%) on the CIFAR-10 dataset. Boldfaced values are the best results among all defenses. Underlined values are the second-best results.

Attack→	BadNets		Blended		LC		WaNet		ISSBA		IAD		Avg	
Metric→ Defense↓	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
RS	9.84	8.00	7.35	5.76	9.21	7.52	98.48	10.00	8.83	8.72	13.28	6.36	24.50	7.73
ShPd	94.28	13.31	49.72	12.89	69.87	13.18	36.25	17.69	95.22	5.50	42.74	7.56	64.68	11.69
FP	96.10	17.13	96.23	16.16	94.76	17.31	96.01	18.64	98.98	19.53	97.08	22.52	96.53	18.55
AC	**99.52**	31.14	**100.00**	30.69	**100.00**	31.16	**99.18**	32.44	**99.94**	34.22	82.99	31.32	96.94	31.83
STRIP	68.70	11.70	65.20	11.70	66.00	12.80	7.90	12.30	56.20	11.40	2.10	14.00	44.35	12.32
SCAn	96.60	**0.77**	**100.00**	**0.00**	0.02	5.05	98.55	**1.06**	99.89	2.61	84.19	**0.13**	79.88	1.60
Ours	99.38	1.35	**100.00**	1.59	**100.00**	**1.20**	91.04	1.48	98.97	**1.17**	**99.12**	1.26	**98.09**	**1.34**

Table 2: Main results (%) on the GTSRB dataset. Boldfaced values are the best results among all defenses. Underlined values are the second-best results.

Attack→	BadNets		Blended		LC		WaNet		ISSBA		IAD		Avg	
Metric→ Defense↓	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
RS	13.20	22.10	10.12	20.40	9.23	19.15	10.10	17.20	8.61	16.98	17.70	17.60	11.49	18.91
ShPd	94.97	12.16	11.58	10.68	96.16	10.60	66.11	14.81	95.92	8.26	31.07	16.10	65.97	12.10
FP	89.05	18.80	30.56	**3.70**	94.71	50.02	67.12	3.24	94.22	7.05	94.37	5.75	78.34	14.76
AC	0.30	8.84	0.00	5.67	4.83	**5.42**	0.42	25.87	99.06	17.48	43.85	10.73	24.74	12.34
STRIP	32.00	9.00	80.40	10.80	7.40	11.00	34.20	11.40	13.00	13.60	6.60	10.60	28.93	11.07
SCAn	46.05	**2.57**	46.02	4.03	30.45	11.39	54.07	**1.88**	96.85	**0.17**	0.09	19.41	45.59	6.58
Ours	**99.99**	6.23	**100.00**	6.72	**100.00**	5.95	**100.00**	6.49	**100.00**	5.43	**100.00**	**4.67**	**100.00**	**5.92**

Defense Setup. For RS, ShPd and STRIP, we adopt the settings suggested in Rosenfeld et al. (2020); Li et al. (2021b); Gao et al. (2022). For FP, we prune 95% of the dormant neurons in the last convolution layer and fine-tune the pruned model using 5% of the training set. We adjust RS, ShPd, and FP to be used as detectors for poisoned samples by comparing the prediction change before and after applying them to an incoming input. For AC, STRIP, SCAn, and our defense, we randomly select 10% from each benign test set as the available benign samples. For SCAn, we identify classes with scores larger than e as potentially target classes, as suggested in Tang et al. (2021). For our defense, we use a threshold $\tau = 2.5$, which gives a reasonable trade-off between TPR and FPR for both benchmarks. More details are presented in Appendix E.4.

Evaluation Metrics. We use the main accuracy (MA) and the attack success rate (ASR) to measure attack performance. Specifically, MA is the number of correctly classified benign samples divided by the total number of benign samples, and ASR is the number of poisoned samples classified as the target class divided by the total number of poisoned samples. We adopt TPR and FPR to evaluate the performance of all defenses, where TPR is computed as the number of detected poisoned inputs divided by the total number of poisoned inputs, while FPR is the number of benign inputs falsely detected as poisoned divided by the total number of benign inputs.

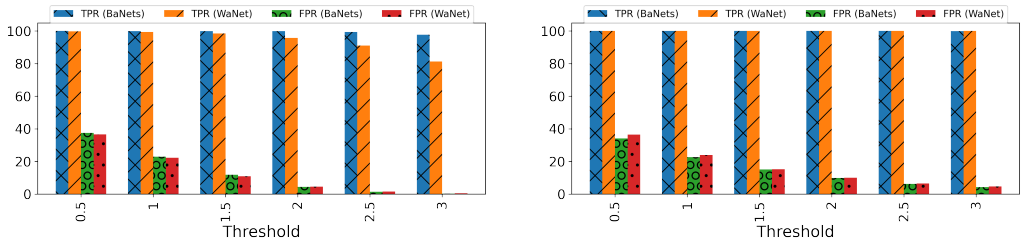
5.2 Main Results

For each attack, we run each defense five times for a fair comparison. Due to space limitations, we present the average TPR and FPR in this section. The detailed results are in Appendix C.

As shown in Table 1-2, existing defenses fail with low TPR or high FPR in many cases, especially on the GTSRB dataset. For example, AC fails in most cases on GTSRB, although it has promising performance on CIFAR-10. In contrast, our method has good performance in detecting all attacks on both datasets. There are only a few cases (4 over 28) where our approach is not optimal or sub-optimal. In these cases, our detection is still on par with state-of-the-art methods, and another indicator ($i.e.$, TPR or FPR) is significantly better than them. For example, when defending against the blended attack on the GTSRB dataset, the TPR of our method is 69.44% larger than that of the FP, which has the smallest FPR in this case. These results verify the effectiveness of our detection.

5.3 Discussion

Effect of the Detection Threshold. Figure 7 shows the TPRs and FPRs of our defense with threshold $\tau \in \{0.5, 1, 1.5, 2, 2.5, 3\}$ for BadNets and WaNet. It shows that setting the threshold as 2.5 is reasonable, as it offers a high TPR while keeping a low FPR. We notice that there is a trade-off between TPR and FPR, according to the threshold ($i.e.$, the larger the threshold, the smaller the TPR and FPR). Users should specify its value based on their specific needs.



(a) CIFAR10-ResNet18 (b) GTSRB-MobileNetV2

Figure 7: The impact of detection thresholds on TPR (%) and FPR (%).

Table 3: The impact of poisoning rates.

Poisoning Rate↓, Metric→	MA (%)	ASR (%)	TPR (%)	FPR (%)
1%	91.52	94.15	99.64	1.25
3%	92.28	96.31	99.32	1.32
5%	91.45	97.20	99.36	1.35
10%	91.45	97.56	99.83	1.62

Table 4: The effectiveness of defenses with different features. Latent features denote those generated by the feature extractor that is typically used in existing defenses. Critical features are those extracted by our method from the identified layers.

Metric→	TPR (%)		FPR (%)	
Defense↓, Features→	Latent Features	Critical Features	Latent Features	Critical Features
AC	0.3	**96.32**	8.84	**7.67**
SCAn	46.05	**86.19**	2.57	**1.96**
Ours	1.31	**99.99**	**4.93**	6.23

Effect of Poisoning Rates. We launch BadNets on CIFAR10-ResNet18 using different poisoning rates $\in \{1\%, 3\%, 5\%, 10\%\}$ to study the impact of poisoning rates on our defense. Table 3 shows that the attack success rate (ASR) increases with the increase of the poisoning rate. However, the poisoning rate has minor effects on our TPR and FPR. These results verify our effectiveness again.

Effectiveness of Our Layer Selection. In this section, we verify that the layer of interest (LOI) identified by our method is useful for detecting poisoned samples. Specifically, we compare the performance of AC, SCAn, and our method of detecting BadNets on the GTSRB-MobileNetV2 benchmark using latent features and critical features. We generate latent features based on the feature extractor ($i.e.$, the layer before fully-connected layers) that is typically adopted in existing defenses. The critical features are extracted by LOI used in our method. As shown in Table 4, using our features leads to significantly better performance in almost all cases. In other words, existing detection methods can also benefit from our layer selection. These results verify the effectiveness of our layer selection and partly explain our promising performance.

Effectiveness of Cosine Similarity. We compare cosine similarity with the Euclidean distance as a metric to differentiate between benign and poisoned samples. As shown in Appendix D.2, the cosine similarity gives a better differentiation than the Euclidean distance. It is mostly because the direction of features is more important for detection than their magnitude.

6 Conclusion

In this paper, we conducted a layer-wise feature analysis of the behavior of benign and poisoned samples generated by attacked DNNs. We revealed that the feature difference between benign and poisoned samples tends to reach the maximum at a critical layer, which can be easily located based on the behaviors of benign samples. Motivated by these findings, we proposed a simple yet effective backdoor detection to determine whether a given suspicious testing sample is poisoned by analyzing the differences between its features and those of a few local benign samples. We conducted extensive experiments on benchmark datasets to verify the effectiveness and efficiency of our detection.

ETHICS STATEMENT

DNNs are widely adopted in many mission-critical areas (*e.g.*, autonomous driving); therefore, their security is of great significance. The vulnerability of DNNs to backdoor attacks raises serious concerns about using third-party training resources. In this paper, we propose a general method to detect poisoned samples at inference time. This work has no particular ethical issues because our method is purely defensive and does not reveal any new vulnerabilities of DNNs.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of datasets, models, training, and attack settings in Appendix E. We also describe the adopted computing facilities and provide the anonymized open-source codes of our main experiments in the same appendix.

REFERENCES

Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010.

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. In *AAAI Workshop*, 2019.

Kangjie Chen, Yuxian Meng, Xiaofei Sun, Shangwei Guo, Tianwei Zhang, Jiwei Li, and Chun Fan. Badpre: Task-agnostic backdoor attacks to pre-trained nlp foundation models. In *ICLR*, 2022a.

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

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *ICML*, 2019.

Mireia Farrús. Voice disguise in automatic speaker recognition. *ACM Computing Surveys (CSUR)*, 51(4):1–22, 2018.

Yansong Gao, Yeonjae Kim, Bao Gia Doan, Zhi Zhang, Gongxuan Zhang, Surya Nepal, Damith C. Ranasinghe, and Hyoungshick Kim. Design and evaluation of a multi-domain Trojan detection method on deep neural networks. *IEEE Transactions on Dependable and Secure Computing*, 19 (4):2349–2364, 2022.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

Jonathan Hayase and Weihao Kong. Spectre: Defending against backdoor attacks using robust covariance estimation. In *ICML*, 2021.

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

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

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.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

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

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. In *ICLR*, 2021a.

Yiming Li, Tongqing Zhai, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor attack in the physical world. In *ICLR Workshop*, 2021b.

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

Yiming Li, Mengxi Ya, Yang Bai, Yong Jiang, and Shu-Tao Xia. BackdoorBox: A python toolbox for backdoor learning. 2022b. URL `https://github.com/THUYimingLi/BackdoorBox`.

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

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

Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural Trojans. In *ICCD*, 2017.

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

Tuan Anh Nguyen and Anh Tuan Tran. Wanet-imperceptible warping-based backdoor attack. In *International Conference on Learning Representations*, 2020b.

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

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

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

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

Di Tang, XiaoFeng Wang, Haixu Tang, and Kehuan Zhang. Demon in the variant: Statistical analysis of $dnns$ for robust backdoor contamination detection. In *USENIX Security*, 2021.

Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

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 *IEEE S&P*, 2019.

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

Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *ICLR*, 2019.

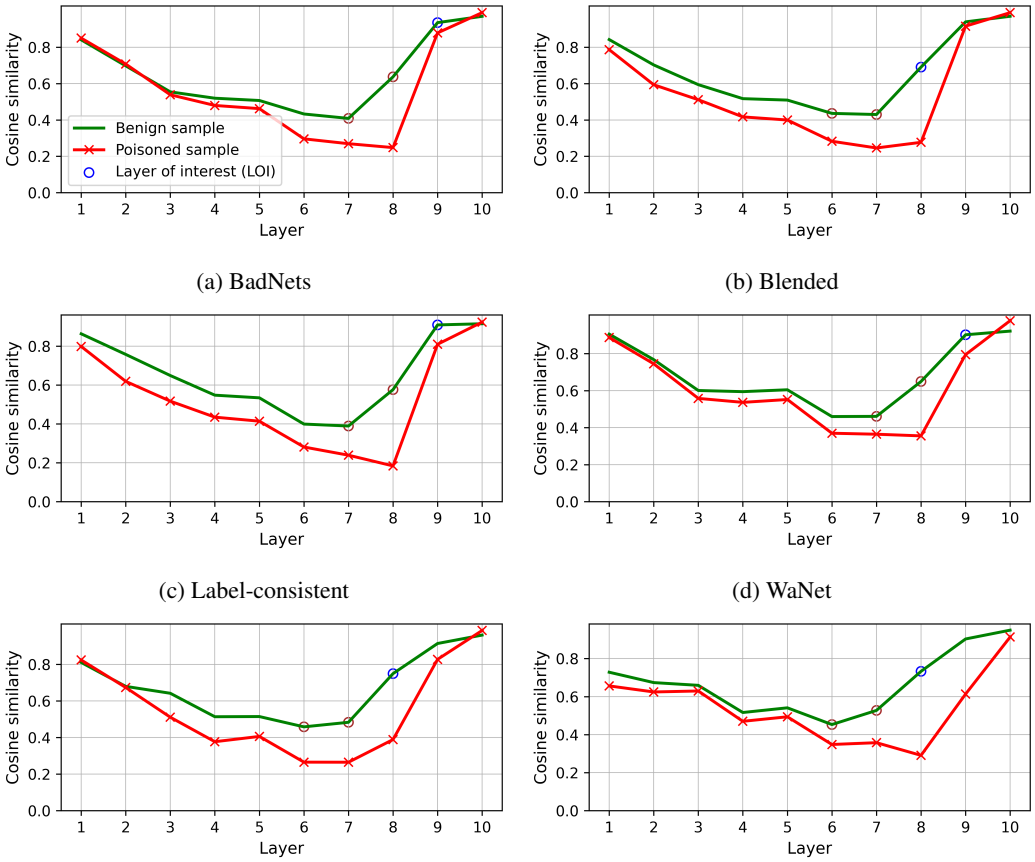
Yi Zeng, Won Park, Z Morley Mao, and Ruoxi Jia. Rethinking the backdoor attacks' triggers: A frequency perspective. In *ICCV*, 2021.

Yi Zeng, Si Chen, Won Park, Z Morley Mao, Ming Jin, and Ruoxi Jia. Adversarial unlearning of backdoors via implicit hypergradient. In *ICLR*, 2022.

Tongqing Zhai, Yiming Li, Ziqi Zhang, Baoyuan Wu, Yong Jiang, and Shu-Tao Xia. Backdoor attack against speaker verification. In *ICASSP*, 2021.

Pu Zhao, Pin-Yu Chen, Payel Das, Karthikeyan Natesan Ramamurthy, and Xue Lin. Bridging mode connectivity in loss landscapes and adversarial robustness. In *ICLR*, 2020.

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



(a) BadNets (b) Blended

(c) Label-consistent (d) WaNet

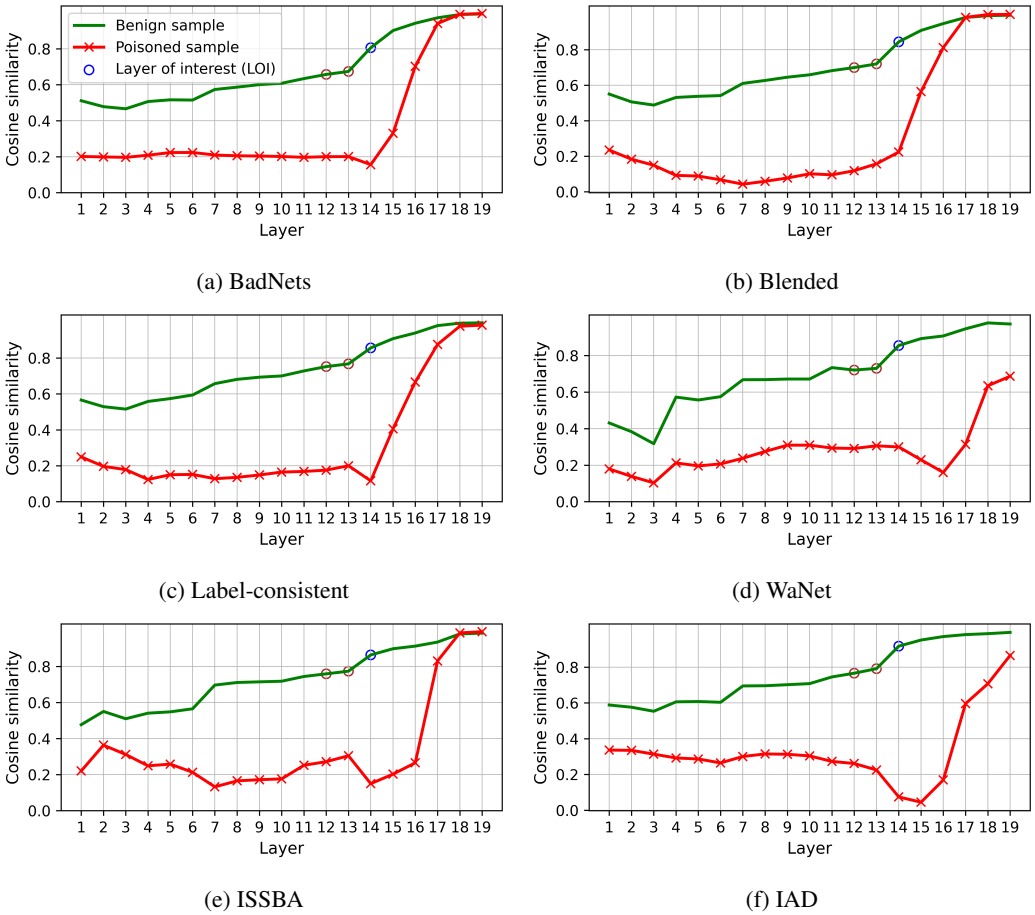
(e) ISSBA (f) IAD

Figure 8: Layer-wise behavior of benign and poisoned samples w.r.t. the target class in the CIFAR10-ResNet18 benchmark

A Additional Results on Layer-wise Feature Analysis

Figure 8 shows the layer-wise behavior of benign and poisoned features w.r.t. the target class on the CIFAR10-ResNet18 benchmark under all the used attacks. Figure 9 shows the same on the GTSRB-MobileNetV2 benchmark.

It can be seen that the layer with the maximum difference in cosine similarity is likely to be one of the three circled layers (the LOI and the two preceding layers). This happens in all cases, except for WaNet on GTSRB-MobileNetV2. We can also notice that the layer-wise gaps are smaller for WaNet, which is stealthier than the other attacks. Nevertheless, no matter how stealthy the attack is, the difference is always evident in one of the circled layers.



(a) BadNets (b) Blended

(c) Label-consistent (d) WaNet

(e) ISSBA (f) IAD

Figure 9: Layer-wise behavior of benign and poisoned samples w.r.t. the target class in the GTSRB-MobileNetV2 benchmark

B PSEUDOCODE OF THE METHOD

Algorithm 1 summarizes our defense.

Table 5: MA% and ASR% under the selected BAs on the CIFAR10-ResNet18 and the GTSRB-MobileNetV2 benchmarks. Best scores are in bold.

Benchmark↓	Metric↓,Attack→	BadNets	Blended	LC	WaNet	ISSBA	IAD
CIFAR10-ResNet18	MA%	91.45	92.19	91.98	91.13	**94.74**	94.42
	ASR%	97.20	**100.0**	99.96	99.04	**100.0**	99.66
GTSRB-MobileNetV2	MA%	97.00	97.27	97.45	96.09	98.43	**98.81**
	ASR%	95.49	**100.0**	**100.0**	91.82	**100.0**	99.63

Algorithm 1 Detecting BAs via layer-wise feature analysis

Input: Suspicious trained DNN f_s; Validation samples X_{val}; Threshold τ; Suspicious input x_s

Output: Boolean value (True/False) tells if x_s is poisoned.

for `each potential target class` $t \in \{1, \dots, \mathcal{C}\}$ **do** ▷ An offline-line loop conducted for one time only
  $X_{t_{val}} \leftarrow$Split t's benign samples from X_{val}
  $m \leftarrow |X_{t_{val}}|$
  $\{a_i^{(L/2)}, \dots, a_i^L\}_{i=1}^m \leftarrow$Layers' features generated by f_s for $\{x_i \in X_{t_{val}}\}$
  $\hat{a}_t^l \leftarrow \frac{1}{m}\sum_{i=1}^m a_i^l$ ▷ Estimate t's centroid at layer $l \in \{L/2, \dots, L\}$
  $cs_i^l \leftarrow \text{CosineSimilarity}(a_i^l, \hat{a}_t^l)$ ▷ Similarity of a_i^l to its centroid
  $\hat{cs}_t^l \leftarrow \frac{1}{m}\sum_{i=1}^m cs_i^l$ ▷ Aggregate computed benign similarities at layer l
  $LOI_t \leftarrow \text{IdentifyLayerOfInterest}(\{\hat{cs}_t^{L/2}, \dots, \hat{cs}_t^{L/2}\})$
  $cs_i \leftarrow cs_i^{LOI_t-2} + cs_i^{LOI_t-1} + cs_i^{LOI_t}$
  $\mu_t, \sigma_t \leftarrow \text{MEAN}(\{cs_i\}_{i=1}^m), \text{STD}(\{cs_i\}_{i=1}^m)$
$IsPoisoned \leftarrow False$
$\hat{y}_s \leftarrow f_s(x_s)$ ▷ $\hat{y}_s$ is the predicted class by f_s for x_s
for `each potential target class` $t \in \{1, \dots, \mathcal{C}\}$ **do**
  if $\hat{y}_s = t$ **then**
    $\{cs_s^{LOI_t-2}, cs_j^{LOI_t-1}, cs_j^{LOI_t}\} \leftarrow \{\text{CosineSimilarity}(a_s^l, \hat{a}_t^l)\}_{l=LOI_t-2}^{LOI_t}$
    $cs_s \leftarrow cs_s^{LOI_t-2} + cs_s^{LOI_t-1} + cs_s^{LOI_t}$
    if $cs_s < (\mu_t - \tau \times \sigma_t)$ **then**
      $IsPoisoned \leftarrow True$
return $IsPoisoned$
procedure IdentifyLayerOfInterest($\{\hat{cs}^{L/2}, \dots, \hat{cs}^{L/2}\}$)
  $max_{diff} \leftarrow |\hat{cs}^{(L/2)+1} - \hat{cs}^{L/2}|$
  $LOI \leftarrow (L/2) + 1$
  for $l \in \{(L/2) + 2, \dots, L\}$ **do**
    $l_{diff} \leftarrow |\hat{cs}^l - \hat{cs}^{l-1}|$
    if $l_{diff} > max_{diff}$ **then**
      $max_{diff} \leftarrow l_{diff}$
      $LOI \leftarrow l$
  return LOI

C Performance of Attacks

Table 5 shows the performance of the selected attacks on the CIFAR10-ResNet18 and the GTSRB-MobileNetV2 benchmarks. It can be seen that sample-specific attacks (*e.g.*, ISSBA and IAD), performed better than other attacks in terms of MA and ASR.

Table 6: Stability on the CIFAR10-ResNet18 benchmark (±: standard deviation). Lowest standard deviations are in bold.

Attack→	BadNets		Blended		LC		WaNet		ISSBA		IAD		Avg	
Metric→ Defense↓	TPR%	FPR%	TPR%	FPR%	TPR%	FPR%	TPR%	FPR%	TPR%	FPR%	TPR%	FPR%	TPR%	FPR%
AC	99.52 (±0.25)	31.14 (±12.54)	100.00 (±**0.00**)	30.69 (±13.59)	100.00 (±**0.00**)	31.16 (±12.43)	99.18 (±0.23)	32.44 (±13.80)	99.94 (±0.12)	34.22 (±13.91)	82.99 (±14.49)	31.32 (±13.12)	96.94 (±2.52)	31.83 (±13.23)
SCAn	96.60 (±0.23)	0.77 (±1.54)	100.00 (±**0.00**)	0.00 (±**0.00**)	0.02 (±0.02)	5.05 (±0.19)	98.55 (±**0.04**)	1.06 (±1.21)	99.89 (±**0.08**)	2.61 (±3.38)	84.19 (±30.54)	0.13 (±0.24)	79.88 (±5.15)	1.60 (±1.09)
FP	96.10 (±**0.12**)	17.13 (±0.33)	96.23 (±2.92)	16.16 (±0.22)	94.76 (±1.11)	17.31 (±0.60)	96.01 (±0.86)	18.64 (±0.69)	98.98 (±0.84)	19.53 (±0.70)	97.08 (±1.01)	22.52 (±0.97)	96.53 (±1.14)	18.55 (±0.59)
Ours	99.38 (±0.34)	1.35 (±**0.24**)	100.00 (±**0.00**)	1.59 (±0.39)	100.00 (±**0.00**)	1.20 (±**0.20**)	91.04 (±1.19)	1.48 (±**0.16**)	98.97 (±0.51)	1.17 (±**0.09**)	99.12 (±**0.45**)	1.26 (±**0.13**)	98.09 (±**0.42**)	1.34 (±**0.20**)

Table 7: Stability on the GTSRB-MobileNetV2 benchmark (±: standard deviation). Lowest standard deviations are in bold.

Attack→	BadNets		Blended		LC		WaNet		ISSBA		IAD		Avg	
Metric→ Defense↓	TPR%	FPR%	TPR%	FPR%	TPR%	FPR%	TPR%	FPR%	TPR%	FPR%	TPR%	FPR%	TPR%	FPR%
AC	0.30 (±0.20)	8.84 (±12.58)	0.00 (±**0.00**)	5.67 (±9.48)	4.83 (±1.36)	5.42 (±9.05)	0.42 (±0.84)	25.87 (±13.51)	99.06 (±0.34)	17.48 (±13.78)	43.85 (±17.09)	10.73 (±13.49)	24.74 (±3.31)	12.34 (±11.98)
SCAn	46.05 (±3.76)	2.57 (±2.06)	46.02 (±5.14)	4.03 (±2.29)	30.45 (±8.24)	11.39 (±5.10)	54.07 (±12.92)	1.88 (±1.55)	96.85 (±3.32)	0.17 (±**0.32**)	0.09 (±0.04)	19.41 (±4.09)	45.59 (±5.57)	6.58 (±2.57)
FP	89.05 (±4.56)	18.80 (±17.73)	30.56 (±22.79)	3.70 (±**0.17**)	94.71 (±0.10)	50.02 (±0.70)	67.12 (±5.69)	3.24 (±**0.13**)	94.22 (±0.20)	7.05 (±1.20)	94.37 (±0.33)	5.75 (±**0.59**)	78.34 (±5.61)	14.76 (±3.42)
Ours	99.99 (±**0.01**)	6.23 (±**0.46**)	100.00 (±**0.00**)	6.72 (±0.53)	100.00 (±**0.00**)	5.95 (±**0.20**)	100.00 (±**0.00**)	6.49 (±0.47)	100.00 (±**0.00**)	5.43 (±0.48)	100.00 (±**0.00**)	4.67 (±0.99)	100.00 (±**0.00**)	5.92 (±**0.52**)

Table 8: Comparison between Euclidean distance and cosine similarity as metrics to differentiate between benign and poisoned samples (±: standard deviation). Best scores are in bold.

Threshold→	0.5		1		1.5		2		2.5		3	
Evaluation metric→ Similarity metric↓	TPR%	FPR%	TPR%	FPR%	TPR%	FPR%	TPR%	FPR%	TPR%	FPR%	TPR%	FPR%
Euclidean distance	99.98 (±0.01)	**24.77** (±**0.64**)	97.12 (±3.13)	**13.64** (±0.90)	95.73 (±2.41)	**8.67** (±**0.36**)	70.84 (±10.00)	4.69 (±0.36)	53.13 (±12.94)	3.21 (±1.78)	15.71 (±10.61)	1.56 (±**0.06**)
Cosine similarity	**100.00** (±**0.00**)	33.65 (±1.11)	**99.99** (±**0.00**)	18.74 (±**0.73**)	**99.91** (±**0.05**)	8.71 (±0.88)	**99.76** (±**0.04**)	**3.89** (±**0.32**)	**99.12** (±**0.45**)	**0.17** (±**0.13**)	**95.99** (±**3.04**)	**0.40** (±0.18)

D Additional Discussion

D.1 Stability Comparison

We compare the stability of our defense with that of AC, SCAn, and FP on the CIFAR10-ResNet18 and GTSRB-MobileNetV2 benchmarks. We ran each defense five times and report the average TPR and FPR with their standard deviations. Tables 6 and 7 show that our defense, in general, is more stable than the others.

D.2 Effectiveness of Cosine Similarity

We also tried the Euclidean distance as a metric to differentiate between benign and poisoned samples, as we did with cosine similarity. The only difference was considering any suspicious input with a summed distance greater than the mean of benign samples with τ standard deviation as potentially poisoned. Table 8 shows the detection performance of our defense with each of the two metrics in the CIFAR10-ResNet18 benchmark under the IAD BA with different thresholds. It can be seen that cosine similarity provides a better differentiation between benign and poisoned samples. A possible explanation is that the direction of features is more important for detection than their magnitude.

D.3 Runtime Comparison

We compared the average CPU runtime (in seconds) of our defense with that of AC and SCAn on the whole benign and poisoned test sets. Figure 10 shows that our defense had the shortest runtime on CIFAR10-ResNet18 and the second shortest on GTSRB-MobileNetV2. It had a runtime slightly longer than that of AC on GTSRB-MobileNetV2 because MobileNetV2 contains a larger number of intermediate layers, which increases the time required to analyze them.

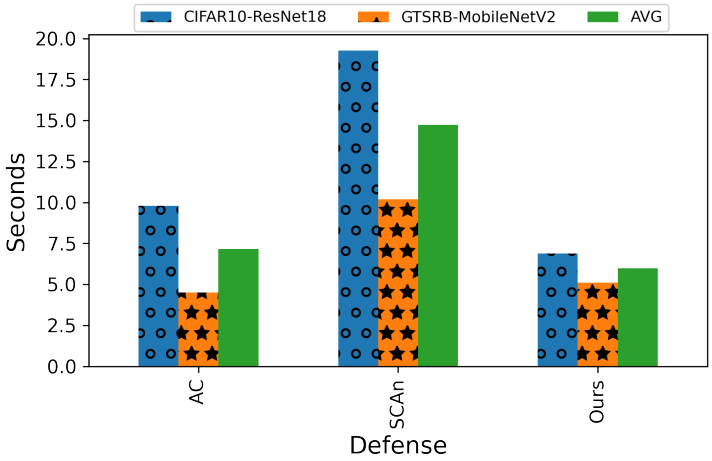


Figure 10: Average CPU running time in seconds.

Table 9: Statistics of the used datasets and DNNs

Dataset	Input size	# Classes	# Training samples	#Test samples	# Available samples	DNN model	# Layers
CIFAR-10	3x32x32	10	50,000	10,000	1,000	ResNet18	10
GTSRB	3x32x32	43	39,209	12,630	1,263	MobileNetV2	19

E Detailed Settings for Experiments

We used the PyTorch framework to implement the experiments on an AMD Ryzen 5 3600 6-core CPU with 32 GB RAM, an NVIDIA GTX 1660 GPU, and Windows 10 OS. In addition, we used the BackdoorBox (Li et al., 2022b) open toolbox for conducting all attacks and re-implemented the other defenses used in our work. The source code, pre-trained models, and poisoned test sets of our defense are available at `https://github.com/anonymized1/DBALFA`.

E.1 Datasets and DNNs

Table 9 summarizes the statistics of the used datasets and DNNs and the number of benign samples available to the defender. Note that, for ease of computation, we consider as a layer each convolutional block other than the first convolutional layer and the last fully connected layer.

E.2 Training Setting

We used the cross-entropy loss and the SGD optimizer with a momentum 0.9 and weight decay 5×10^{-4} on all benchmarks. We used initial learning rates 0.1 for ResNet18 and 0.01 for MobileNetV2, and trained models for 200 epochs. The learning rates were decreased by a factor of 10 at epochs 100 and 150, respectively. We set the batch size to 128 and trained all models until they converged.

E.3 Attack Setting

The target class on all datasets was 1 for BadNets (Gu et al., 2019), the BA with blended strategy (Chen et al., 2017) (Blended), the invisible sample-specific BA (Li et al., 2021c) (ISSBA), and the input-aware dynamic BA (Nguyen & Tran, 2020a) (IAD). The target classes for the label-consistent BA (Turner et al., 2019) and WaNet (Nguyen & Tran, 2020b) were 2 and 0, respectively, on all datasets. The trigger patterns of attacks were the same as those presented in Section 5.1. In particular, we set the blended ratio to $\lambda = 0.1$ for the blended attack on all datasets. We used the label-consistent BA with maximum perturbation size 16. For WaNet, we set the noise rate to $\rho_n = 0.2$, the control grid size to $k = 4$, and the warping strength to $s = 0.5$ on all datasets, as suggested in the WaNet paper (Nguyen & Tran, 2020b). For IAD (Nguyen & Tran, 2020a), we trained the classifier and the trigger generator concurrently. We attached the dynamic trigger to the samples from other classes and relabeled them as the target label.

E.4 DEFENSE SETTING

For RS, we generated 100 neighbors of each input with a mean = 0 and a standard deviation = 0.1, as suggested in Cohen et al. (2019). We set the shrinking rate to 10% for ShPd and padded shrinked images with 0-pixels to expand them to their original size, as suggested in Li et al. (2021b). For FP, we pruned 95% of the dormant neurons in the last convolution layer and fine-tuned the pruned model using 5% of the training set. We adjusted RS, ShPd, and FP to be used as detectors for poisoned samples by comparing the change in prediction before and after applying them to an incoming input. For AC, STRIP, SCAn, and our defense, we randomly selected 10% from each benign test set as the available benign samples. Then, for AC, we used the available benign samples, from each class, for normalizing benign and poisoned test samples and identifying potential poisoned clusters. For STRIP, we blended each input with 100 random inputs from the available benign samples using a blending value $\alpha = 0.5$, as suggested in Gao et al. (2022). Then, we identified inputs with entropy below the 10-th percentile of the entropies of benign samples as potentially poisoned samples. For SCAn, we identified classes with scores larger than e as potential target classes, as suggested in Tang et al. (2021), and identified the cluster that did not contain the available benign samples as a poisoned cluster. For our defense, we used a threshold $\tau = 2.5$, which gave us a reasonable trade-off between TPR and FPR on both benchmarks.