arXiv:2302.12758v1 [cs.CR] 24 Feb 2023

Defending Against Backdoor Attacks by Layer-wise Feature Analysis

Najeeb Moharram Jebreel[1], Josep Domingo-Ferrer[1], and Yiming Li[2]

[1] Universitat Rovira i Virgili
Av. Països Catalans 26, E-43007 Tarragona, Catalonia
{najeeb.jebreel,josep.domingo}@urv.cat
[2] Tsinghua University, Beijing, China
li-ym18@mails.tsinghua.edu.cn

Abstract. Training deep neural networks (DNNs) usually requires massive training data and computational resources. Users who cannot afford this may prefer to outsource training to a third party or resort to publicly available pre-trained models. Unfortunately, doing so facilitates a new training-time attack (*i.e.*, backdoor attack) against DNNs. This attack aims to induce misclassification of input samples containing adversary-specified trigger patterns. 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 be maximum at a critical layer, which is *not always* the one typically used in existing defenses, namely the layer before fully-connected layers. We also demonstrate how to locate this critical layer based on the behaviors of benign samples. We then 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 defense.

Keywords: Backdoor Detection · Backdoor Defense · Backdoor Learning · AI Security · Deep Learning.

1 Introduction

In recent years, deep neural networks (DNNs) have successfully been applied in many tasks, such as computer vision, natural language processing, and speech recognition. However, training DNNs requires massive training data and computational resources, and users who cannot afford it may opt to outsource training to a third-party (*e.g.*, a cloud service) or leverage pre-trained DNNs. Unfortunately, losing control over training facilitates *backdoor attacks* [2,5,10] against DNNs. In these attacks, the adversary poisons a few training samples to cause the DNN to misclassify samples containing pre-defined trigger patterns into an adversary-specified target class. Nevertheless, the attacked models behave normally on benign samples, which makes the attack stealthy. Since DNNs are used

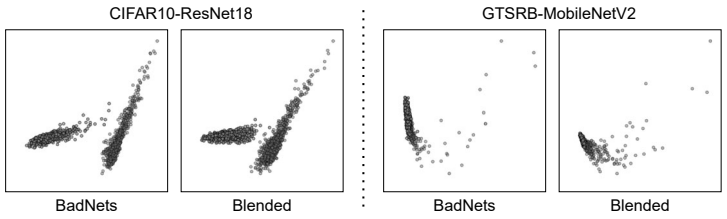


Fig. 1: PCA-based visualization of features of benign (green) and poisoned samples (red) generated by the layer before the fully connected layers of models attacked by BadNets [5] and Blended [2]. As shown in this figure, features of poisoned and benign samples are not well separable on the GTSRB benchmark.

in many mission-critical tasks (*e.g.*, autonomous driving, or facial recognition), it is urgent to design effective defenses against these attacks.

Among all backdoor defenses in the literature, backdoor detection is one of the most important defense paradigms, where defenders attempt to detect whether a suspicious object (*e.g.*, model or sample) is malicious. Currently, most existing backdoor detectors assume poisoned samples have different feature representations from benign samples, and they tend to focus on the layer before the fully connected layers [1,22,6]. Two intriguing questions arise: **(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 give a negative answer to the first question (see Figure 1). To answer the second one, we 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 can be easily located based on the behaviors of benign samples. Specifically, *the critical layer is the one or near the one that contributes most to assigning benign samples to their true class.* Based on this finding, 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 inputs. In short, our main contributions are four-fold. **(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 neatest. **(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.

2 Related Work: Backdoor Attacks and Defenses

In this paper, we focus on backdoor attacks and defenses in image classification. Other deep learning tasks are out of our current scope.

BadNets [5] was the first backdoor attack, which randomly selected a few benign samples and generated their poisoned versions by stamping a trigger patch onto their images and reassigning their label as the target label. Later [2] noted that the poisoned image should be similar to its benign version for stealthiness; these authors proposed a blended attack by introducing trigger transparency. However, these attacks are with poisoned labels and therefore users can still detect them by examining the image-label relation. To circumvent this, [23] 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 were exploited to perturb the selected benign samples before conducting the standard trigger injection process. [18] adopted image warping as the backdoor trigger, which modifies the whole image while preserving its main content. Besides, [17] proposed the first sample-specific attack, where the trigger varies across samples. However, such triggers are visible and the adversaries need to control the whole training process. More recently, [14] introduced the first poison-only invisible sample-specific attack to address these problems.

Existing backdoor defenses fall into three main categories: input filtering, input pre-processing, and model repairing. **Input filtering** intends to differentiate benign and poisoned samples based on their distinctive behaviors, like the separability of the feature representations of benign and poisoned samples. For example, [6] introduced a robust covariance estimation of feature representations to amplify the spectral signature of poisoned samples. [25] proposed to filter inputs inspired by the understanding that poisoned images tend to have some high-frequency artifacts. [4] proposed to blend various images on the suspicious one, since the trigger pattern can still mislead the prediction no matter what the background contents are. **Input pre-processing** modifies each input sample before feeding it into the deployed DNN. Its rationale is to perturb potential trigger patterns and thereby prevent backdoor activation. [16] proposed the first defense in this category where they used an encoder-decoder to modify input samples. [19] employed randomized smoothing to generate a set of input neighbors and averaged their predictions. Further, [13] demonstrated that if the location or appearance of the trigger is slightly different from that used for training, the attack effectiveness may degrade sharply. Based on this, they proposed to pre-process images with spatial transformations. **Model repairing** aims at erasing backdoors contained in the attacked DNNs. For example, [16,26,9] showed that users can effectively remove backdoors by fine-tuning the attacked DNNs with a few benign samples. [15] revealed that model pruning can also remove backdoors effectively, because backdoors are mainly encoded in specific neurons. Very recently, [24] proposed to repair compromised models with adversarial model unlearning. In this paper, *we focus on input filtering,* which is very convenient to protect deployed DNNs.

3 Layer-wise Feature Analysis

A deep neural network (DNN) $f(\boldsymbol{x})$ is composed by L layers $f^l, l \in [1, L]$. Each f^l has a weight matrix $\boldsymbol{w^l}$, a bias vector $\boldsymbol{b^l}$, and an activation function σ^l. The output of f^l is $\boldsymbol{a^l} = f^l(\boldsymbol{a^{l-1}}) = \sigma^l(\boldsymbol{w^l} \cdot \boldsymbol{a^{l-1}} + \boldsymbol{b^l})$, where f^1 takes input $\boldsymbol{x}$ and f^L outputs a vector $\boldsymbol{a^L}$ with $\mathcal{C}$ classes. The vector $\boldsymbol{a^L}$ is softmaxed to get probabilities $\boldsymbol{p}$. A DNN has a feature extractor that maps $\boldsymbol{x}$ to latent features, which are input to fully connected layers for classification.

In this paper, we use DNNs as $\mathcal{C}$-class classifiers, where y_i is the ground truth label of $\boldsymbol{x_i}$ and $\hat{y_i}$ is the index of the highest probability in $\boldsymbol{p_i}$. Also, activations of intermediate layers are analyzed for detecting poisoned samples.

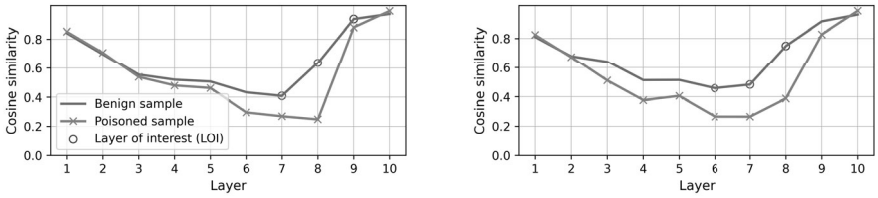
We notice that the predictions of attacked DNNs for both benign samples from the target class and poisoned samples are all the target label. The attacked DNNs mainly exploit class-relevant features to predict these benign samples while they use trigger-related features for poisoned samples. We suggest 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 $X_t = \{\boldsymbol{x_i}\}_{i=1}^{|X_t|}$ be benign samples with true class t, and let $\{\boldsymbol{a_i^1}, \ldots, \boldsymbol{a_i^L}\}_{i=1}^{|X_t|}$ be their intermediate features generated by f'. The centroid of t's benign features at layer l is defined as $\hat{\boldsymbol{a}}_{\boldsymbol{t}}^{\boldsymbol{l}} = \frac{1}{|X_t|} \sum_{i=1}^{|X_t|} \boldsymbol{a_i^l}$, and $\{\hat{\boldsymbol{a}}_{\boldsymbol{t}}^{\boldsymbol{1}}, \ldots, \hat{\boldsymbol{a}}_{\boldsymbol{t}}^{\boldsymbol{L}}\}$ is the set of layer-wise centroids of t's benign features.*

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

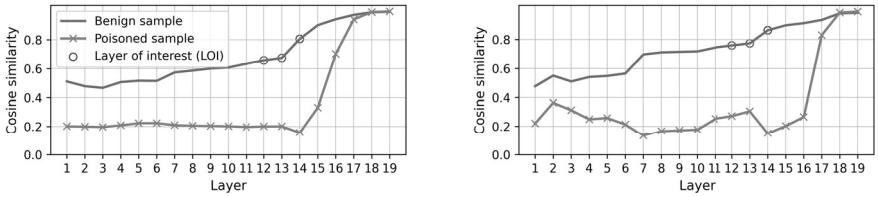
Settings. We conducted six representative attacks on four classical benchmarks: CIFAR10-ResNet18, CIFAR10-MobileNetV2, GTSRB-ResNet18, and GTSRB-MobileNetV2. The six attacks were BadNets [5], the backdoor attack with blended strategy (Blended) [2], the label-consistent attack (LC) of [23], WaNet [18], ISSBA [14], and IAD [17]. More details on the datasets, DNNs, and attack settings are presented in Section 5. Specifically, for each attacked DNN f' with a target class t, we estimated $\{\hat{\boldsymbol{a}}_{\boldsymbol{t}}^{\boldsymbol{1}}, \ldots, \hat{\boldsymbol{a}}_{\boldsymbol{t}}^{\boldsymbol{L}}\}$ 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) BadNets (b) ISSBA

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



(a) BadNets (b) ISSBA

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

a specific layer where the backdoor trigger is activated, causing poisoned samples to get closer to the target class. Figure 3 shows the same phenomenon 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 (*i.e.*, layer 17).

Regardless of the attack or benchmark, when we enter the second half of DNN layers (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 B. In those materials, we also provide confirmation that the above distinctive behaviors hold regardless of the datasets or models being used. 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 The Proposed Defense

Threat Model. Consider a user that obtains a suspicious trained f_s that might contain hidden backdoors. We assume that the user has limited computational

Algorithm 1 Identify layer of interest (LOI).

Input: Cosine similarities $\{\hat{cs}_t^{\lfloor L/2 \rfloor}, \ldots, \hat{cs}_t^{L}\}$ for potential target class t

$max_{diff} \leftarrow \hat{cs}_t^{\lfloor L/2 \rfloor+1} - \hat{cs}_t^{\lfloor L/2 \rfloor}$; $LOI_t \leftarrow \lfloor L/2 \rfloor + 1$;
for $l \in \{\lfloor L/2 \rfloor + 2, \ldots, L\}$ **do**
  $l_{diff} \leftarrow \hat{cs}_t^{l} - \hat{cs}_t^{l-1}$;
  if $l_{diff} > max_{diff}$ **then**
    $max_{diff} \leftarrow l_{diff}$; $LOI_t \leftarrow l$;
return LOI_t.

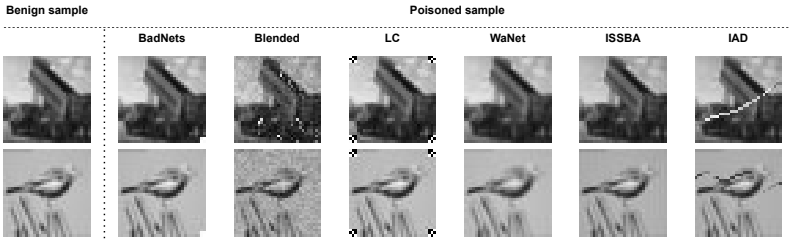


Fig. 4: The example of benign samples and their poisoned versions generated by six representative backdoor attacks.

resources or benign samples, and therefore cannot repair f_s. The user wants to defend by detecting at inference time whether a suspicious incoming input $\boldsymbol{x_s}$ is poisoned, given f_s. 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 $X_{t_{val}}$. Let $m = |X_{t_{val}}|$ denote the number of available samples labeled as t.

Method Design. Based on the lessons learned in Section 3, our method to detect poisoned samples at inference time consists of four steps. **1)** Estimate the layer-wise features' centroids of class t for each of layers $\lfloor L/2 \rfloor$ to L 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) as per Algorithm 2, 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 $\boldsymbol{x_s}$ classified as t by f_s, **4.1)** compute its cosine similarities to the estimated centroids in the above-mentioned three layers, and **4.2)** consider it as a potentially poisoned input if its summed similarities fall below the obtained mean by a specific number τ of standard deviations (called threshold in what follows). A detailed pseudocode can be found in Appendix A.

5 Experiments

5.1 Main Settings

Datasets and DNNs. In this paper, we use two classic benchmark datasets, namely CIFAR10 [8] and GTSRB [21]. We use the ResNet18 [7] on CIFAR10 and

the MobileNetV2 [20] on GTSRB. More details are presented in Appendix D.1. The source code, pre-trained models, and poisoned test sets of our defense are available at `https://github.com/NajeebJebreel/DBALFA`.

Attack Baselines. We evaluated each defense under the six attacks mentioned in Section 3: 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 compared our defense with six representative defenses, namely randomized smoothing (RS) [19], ShrinkPad (ShPd) [13], activation clustering (AC) [1], STRIP [4], SCAn [22], and fine-pruning (FP) [15]. 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 took the following settings. We used a 2×2 square as the trigger pattern for BadNets (as suggested in [5]). We adopted the random noise pattern, with a 10% blend ratio, for Blended (as suggested in [2]). The trigger pattern adopted for the LC attack was the same used in BadNets. For WaNet, ISSBA, and IAD, we took 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%. We implement baseline attacks based on the codes in `BackdoorBox` [12]. More details on settings are given in Appendix D.3. Figure 4 shows an example of poisoned samples generated by different attacks.

Defense Setup. For RS, ShPd and STRIP, we took the settings suggested in [19,13,4]. 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 prediction change 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. For SCAn, we identified classes with scores larger than e as potential target classes, as suggested in [22]. For our defense, we used a threshold $\tau = 2.5$, which gives a reasonable trade-off between TPR and FPR for both benchmarks.

Evaluation Metrics. We used 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 adopted 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, whereas FPR is the number of benign inputs falsely detected as poisoned divided by the total number of benign inputs.

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**

5.2 Main Results

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

As shown in Tables 1 and 2, existing defenses failed to detect attacks with low TPR or high FPR in many cases, especially on the GTSRB dataset. For example, AC failed in most cases on GTSRB, although it had promising performance on CIFAR-10. In contrast, our method had good performance in detecting all attacks on both datasets. There were only a few cases (4 over 28) where our approach was neither optimal nor close to optimal. In these cases, our detection was still on par with state-of-the-art methods, and another indicator (*i.e.*, TPR or FPR) was significantly better than them. For example, when defending against the blended attack on the GTSRB dataset, the TPR of our method was 69.44% larger than that of FP, which had the smallest FPR in this case. These results confirm the effectiveness of our detection.

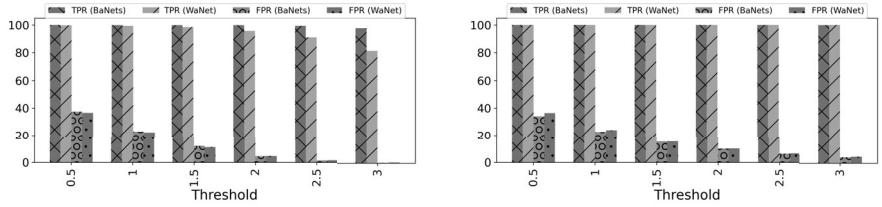
5.3 Discussions

Performance of Attacks. Table 3 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.

Effects of the Detection Threshold. Figure 5 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.

Table 3: MA% and ASR% under the selected backdoor attacks 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



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

Fig. 5: Impact of detection thresholds on TPR (%) and FPR (%)

Table 4: 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 5: 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 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

It can be seen that a threshold 2.5 is reasonable, as it offers a high TPR while keeping a low FPR. Note that the larger the threshold, the smaller the TPR and FPR. Users should choose the threshold based on their specific needs.

Effects of the Poisoning Rate. We launched BadNets on CIFAR10-ResNet18 using different poisoning rates $\in \{1\%, 3\%, 5\%, 10\%\}$ to study the impact of poisoning rates on our defense. Table 4 shows the attack success rate (ASR) increases with the poisoning rate. However, the poisoning rate has minor effects on our TPR and FPR. These results confirm again the effectiveness of our method.

Effectiveness of Our Layer Selection. We compared the performance of AC, SCAn, and our method at detecting BadNets on the GTSRB-MobileNetV2

Table 6: Performance of features from individual layers compared to identified layers by our defense. The LOI of WaNet and IAD are 9 and 8, respectively.

Layer		1	2	3	4	5	6	7	8	9	10	Ours
WaNet	TPR (%)	0.00	0.10	0.05	0.00	0.01	0.00	68.82	98.08	59.82	0.00	91.04
	FPR (%)	0.09	0.82	0.24	0.20	0.21	0.04	2.06	1.52	2.06	0.65	1.48
IAD	TPR (%)	19.32	34.03	6.44	30.49	61.09	78.65	88.81	99.65	99.10	2.36	99.12
	FPR (%)	1.65	1.38	1.44	1.60	2.27	1.70	1.29	1.13	1.09	1.24	1.26

benchmark using latent features and critical features. We generated 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 were extracted by the layer of interest (LOI) used in our method. Table 5 shows that using our features led to significantly better performance in almost all cases. In other words, existing detection methods can also benefit from our LOI selection. Also, we compared the performance of our method on CIFAR10-ResNet18 under WaNet and IAD when using the features of every individual layer, and when using LOI and the two layers before LOI. Table 6 shows that as we approach the critical layer, which was just before LOI with WaNet and at LOI with IAD, the detection performance gets better. Since our method included the critical layer, it also was effective. These results confirm the effectiveness of our layer selection and partly explain our method's good performance.

Effectiveness of Cosine Similarity. We compared the cosine similarity with the Euclidean distance as a metric to differentiate between benign and poisoned samples. In Appendix C.2, we show the cosine similarity gives a better differentiation than the Euclidean distance. This is mostly because the direction of features is more important for detection than their magnitude.

Resistance to Adaptive Attacks. The adversary may adapt his attack to bypass our defense by optimizing the model's original loss $\mathcal{L}_{org}$ and minimizing the layer-wise angular deviation between the features of the poisoned samples and the features' centroids of the target class's benign samples. We studied the impact of this strategy by introducing the *cosine distance* between the features of poisoned samples and the target class centroids as a secondary loss function $\mathcal{L}_{cd}$ in the training objective function. Also, we introduced a penalty parameter β, which yielded a modified objective function $(1-\beta)\mathcal{L}_{org}+\beta\mathcal{L}_{cd}$. The role of β is to control the trade-off between the angular deviation and the main accuracy loss. We then launched BadNets on CIFAR10-ResNet18 under the modified objective function. Table 7 (top subtable) shows MA and ASR with different penalty factors. We can see that values of $\beta < 0.9$ slightly increased the main accuracy because the second loss acted as a regularizer to the model's parameters, which reduced over-fitting. Also, ASR stayed similar to the non-adaptive ASR (when $\beta = 0$). However, the main accuracy degraded with greater β values, because the original loss function was dominated by the angular deviation loss.

Table 7 (bottom subtable) shows the TPRs and FPRs of AC, SCAn, and our defense with different penalty factors. As β increased (up to $\beta = 0.9$), the TPR of

Table 7: Adaptive attack. Top, impact of penalty factor β on MA and ASR. Bottom, impact of penalty factor β on TPR and FPR.

β	0	0.5	0.6	0.7	0.8	0.9	0.91	0.92	0.95
MA (%)	91.45	92.96	92.06	92.65	92.63	90.33	79.97	69.13	10
ASR (%)	97.20	96.72	96.93	96.63	96.29	96.88	96.41	97.36	100

Defense↓	$\beta \rightarrow$ Metric (%)↓	0	0.5	0.6	0.7	0.8	0.9	0.91	0.92	0.95
AC	TPR	99.52	99.20	99.16	45.69	26.26	26.22	23.81	13.38	0.00
	FPR	31.14	29.46	28.85	8.21	7.72	6.21	0.25	7.80	0.00
SCAn	TPR	96.60	96.55	96.60	72.80	56.19	0.00	0.00	0.00	0.00
	FPR	0.77	1.38	4.60	1.14	0.10	0.00	0.00	0.00	0.00
Ours	TPR	99.38	99.41	98.18	97.43	97.52	94.20	24.20	0.00	0.00
	FPR	1.35	1.96	1.44	1.15	0.53	1.40	4.17	0.00	0.00

our defense decreased from 99.38% to 94.20% while FPR was almost unaffected. This shows that the adversary gained a small advantage with $\beta = 0.9$. On the other hand, the other defenses achieved limited or poor robustness compared to ours with the same β values. With $\beta \geq 0.91$, AC, SCAn, and our method defense failed to counter the attack. However, looking at Table 7 (top subtable) we can see the main accuracy degraded with these high β values, which made it easy to reject the model due its low performance.

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 found 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. Based on this finding, 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. Our extensive experiments on benchmark datasets confirmed the effectiveness of our detection. We hope our work can provide a deeper understanding of attack mechanisms, to facilitate the design of more effective and efficient backdoor defenses and more secure DNNs.

Acknowledgments

This research was funded by the European Commission (projects H2020-871042 "SoBigData++" and H2020-101006879 "MobiDataLab"), the Government of Catalonia (ICREA Acadèmia Prize to J.Domingo-Ferrer, grant no. 2021 SGR 00115, and FI_B00760 grant to N. Jebreel), and MCIN/AEI/ 10.13039/501100011033 and "ERDF A way of making Europe" under grant PID2021-123637NB-I00 "CURLING". The authors are with the UNESCO Chair in Data Privacy, but the views in this paper are their own and are not necessarily shared by UNESCO.

References

1. Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I., Srivastava, B.: Detecting backdoor attacks on deep neural networks by activation clustering. In: AAAI Workshop (2019)
2. Chen, X., Liu, C., Li, B., Lu, K., Song, D.: Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526 (2017)
3. Cohen, J., Rosenfeld, E., Kolter, Z.: Certified adversarial robustness via randomized smoothing. In: ICML (2019)
4. Gao, Y., Kim, Y., Doan, B.G., Zhang, Z., Zhang, G., Nepal, S., Ranasinghe, D.C., Kim, H.: 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)
5. Gu, T., Liu, K., Dolan-Gavitt, B., Garg, S.: BadNets: Evaluating backdooring attacks on deep neural networks. IEEE Access **7**, 47230–47244 (2019)
6. Hayase, J., Kong, W.: Spectre: Defending against backdoor attacks using robust covariance estimation. In: ICML (2021)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
8. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
9. Li, Y., Lyu, X., Koren, N., Lyu, L., Li, B., Ma, X.: Neural attention distillation: Erasing backdoor triggers from deep neural networks. In: ICLR (2021)
10. Li, Y., Jiang, Y., Li, Z., Xia, S.T.: Backdoor learning: A survey. IEEE Transactions on Neural Networks and Learning Systems (2022)
11. Li, Y., Ya, M., Bai, Y., Jiang, Y., Xia, S.T.: BackdoorBox: A python toolbox for backdoor learning (2022), `https://github.com/THUYimingLi/BackdoorBox`
12. Li, Y., Ya, M., Bai, Y., Jiang, Y., Xia, S.T.: Backdoorbox: A python toolbox for backdoor learning. arXiv preprint arXiv:2302.01762 (2023)
13. Li, Y., Zhai, T., Jiang, Y., Li, Z., Xia, S.T.: Backdoor attack in the physical world. In: ICLR Workshop (2021)
14. Li, Y., Li, Y., Wu, B., Li, L., He, R., Lyu, S.: Invisible backdoor attack with sample-specific triggers. In: ICCV (2021)
15. Liu, K., Dolan-Gavitt, B., Garg, S.: Fine-pruning: Defending against backdooring attacks on deep neural networks. In: RAID (2018)
16. Liu, Y., Xie, Y., Srivastava, A.: Neural Trojans. In: ICCD (2017)
17. Nguyen, T.A., Tran, A.: Input-aware dynamic backdoor attack. In: NeurIPS (2020)
18. Nguyen, T.A., Tran, A.T.: Wanet-imperceptible warping-based backdoor attack. In: International Conference on Learning Representations (2020)
19. Rosenfeld, E., Winston, E., Ravikumar, P., Kolter, Z.: Certified robustness to label-flipping attacks via randomized smoothing. In: ICML (2020)
20. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: CVPR (2018)
21. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: The German traffic sign recognition benchmark: a multi-class classification competition. In: IJCNN (2011)
22. Tang, D., Wang, X., Tang, H., Zhang, K.: Demon in the variant: Statistical analysis of *dnns* for robust backdoor contamination detection. In: USENIX Security (2021)
23. Turner, A., Tsipras, D., Madry, A.: Label-consistent backdoor attacks. arXiv preprint arXiv:1912.02771 (2019)

24. Zeng, Y., Chen, S., Park, W., Mao, Z.M., Jin, M., Jia, R.: Adversarial unlearning of backdoors via implicit hypergradient. In: ICLR (2022)
25. Zeng, Y., Park, W., Mao, Z.M., Jia, R.: Rethinking the backdoor attacks' triggers: A frequency perspective. In: ICCV (2021)
26. Zhao, P., Chen, P.Y., Das, P., Ramamurthy, K.N., Lin, X.: Bridging mode connectivity in loss landscapes and adversarial robustness. In: ICLR (2020)

A Detailed method

Algorithm 2 summarizes our defense.

Algorithm 2 Detecting backdoor attacks via layer-wise feature analysis

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

Output: Boolean value (True/False) tells if $\boldsymbol{x_s}$ is poisoned.

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

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 $\{(\boldsymbol{a_i^{L/2}}, \dots, \boldsymbol{a_i^L})\}_{i=1}^m$ (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. After that, we compute the layer-wise centroids of the extracted features for each layer $l \in \{L/2, \dots, L\}$ (Line 5). Then, we compute the cosine similarity between the benign features of each layer and their corresponding centroid (Line 6).

Then, we aggregate the computed similarities to approximate the similarity centroid in each layer $l \in \{L/2, (L/2)+1, \dots, L-1, L\}$ (Line 7). 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 (Lines 20-28). We compute the difference between the approximated similarity of each layer and its preceding one, and we identify the layer with the maximum difference as LOI_t. For example, if the maximum difference is $|\hat{cs}_t^{l} - \hat{cs}_t^{l-1}|$, then layer l is the layer of interest.

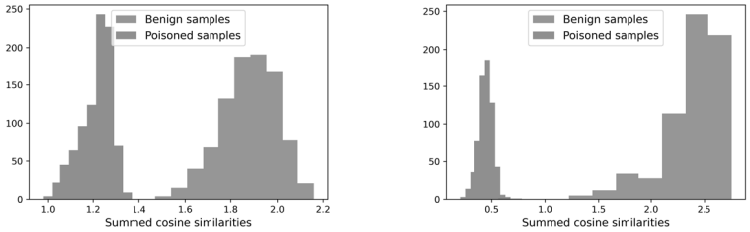
Once we locate LOI_t, we estimate the behavior of benign samples in that layer and in the two layers previous to it. For each sample $\boldsymbol{x_i} \in X_{t_{val}}$, we sum up its computed cosine similarities in the three layers (Line 9). After computing the summed similarities of the m samples and 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 $\boldsymbol{x_s}$ 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 $\boldsymbol{x_s}$ as a potentially poisoned sample if $cs_s < \mu_t - \tau \times \sigma_t$, where τ is an input threshold chosen by the defender that provides a reasonable trade-off between the true positive rate TPR and the false positive rate FPR. Figure 6 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 [23].

B Additional Results on Layer-wise Feature Analysis

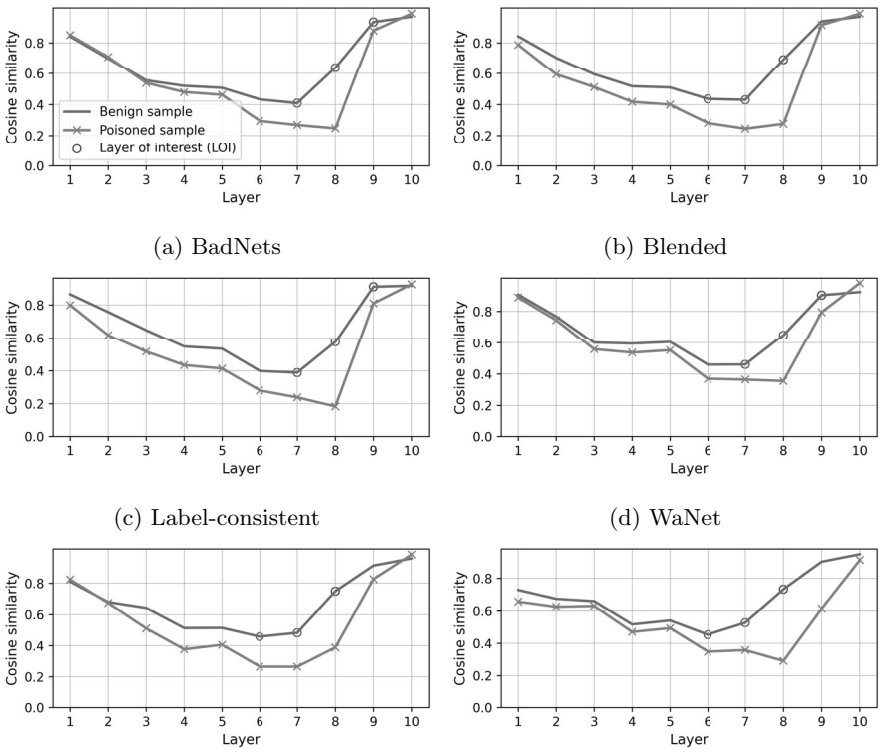
Figure 7 shows the layer-wise behavior of benign and poisoned features w.r.t. the target class on the CIFAR10-ResNet18 benchmark under all the used attacks. Figure 8 shows the same on the GTSRB-MobileNetV2 benchmark.

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) CIFAR10-ResNet18 (b) GTSRB-MobileNetV2

Fig. 6: 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

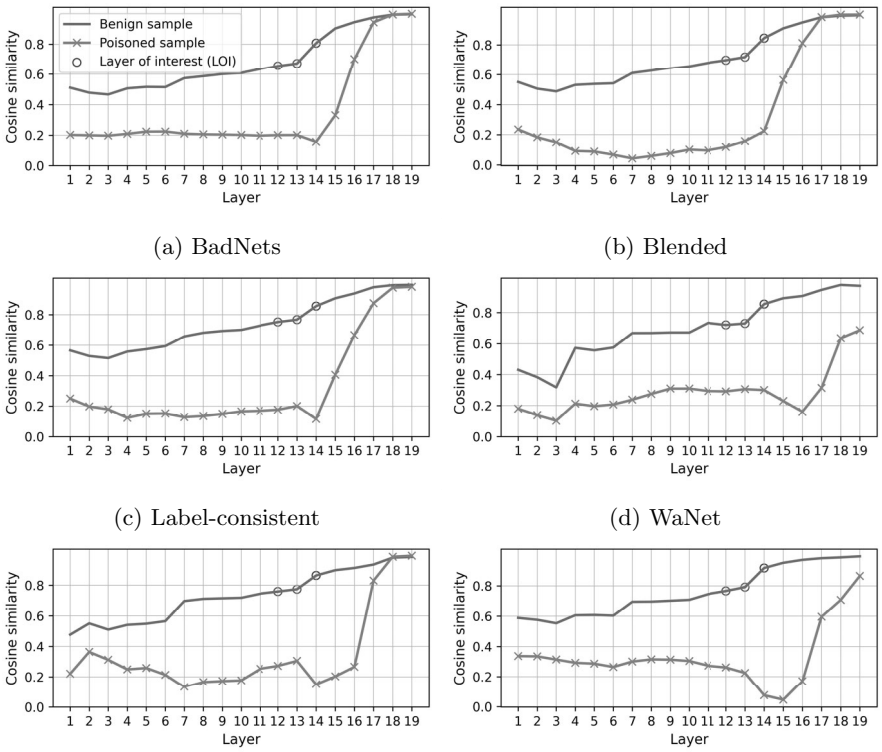


(a) BadNets (b) Blended

(c) Label-consistent (d) WaNet

(e) ISSBA (f) IAD

Fig. 7: Layer-wise behavior of benign and poisoned samples w.r.t. the target class in the CIFAR10-ResNet18 benchmark, under all implemented attacks



(a) BadNets (b) Blended

(c) Label-consistent (d) WaNet

(e) ISSBA (f) IAD

Fig. 8: Layer-wise behavior of benign and poisoned samples w.r.t. the target class in the GTSRB-MobileNetV2 benchmark, under all implemented attacks

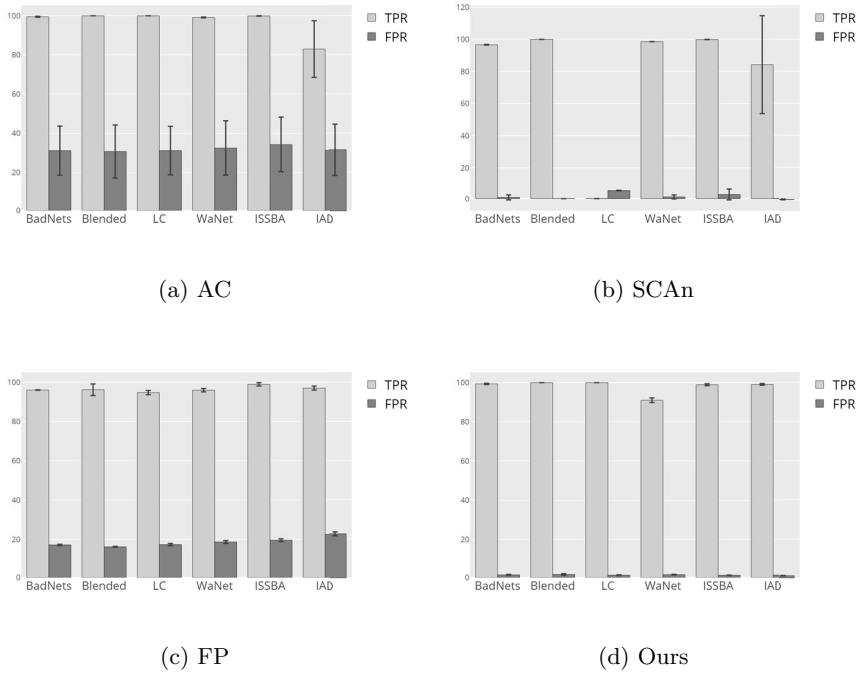
C Additional Discussion

C.1 Stability Comparison

We compared 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 we report the average TPR and FPR with their standard deviations. Error bars in Figure 9 and Figure 10 show that our defense, in general, is more stable than the others.

C.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



(a) AC (b) SCAn

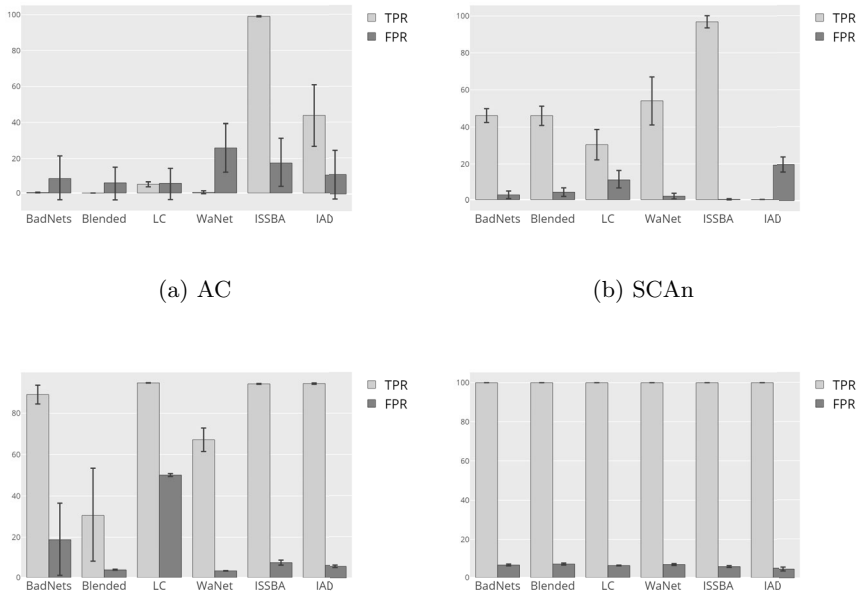
(c) FP (d) Ours

Fig. 9: Stability on the CIFAR10-ResNet18 benchmark

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)

of benign samples with τ standard deviations 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 backdoor attack 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.



(a) AC (b) SCAn

(c) FP (d) Ours

Fig. 10: Stability on the GTSRB-MobileNetV2 benchmark

C.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 11 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.

D 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 [11] open toolbox for conducting all attacks and re-implemented the other defenses used in our work.

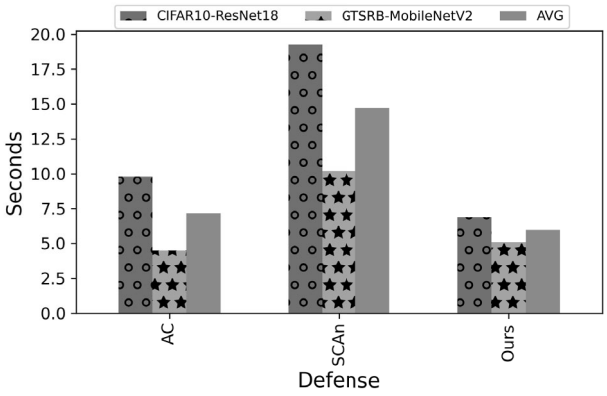


Fig. 11: Average CPU running time in seconds.

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

D.1 Datasets and DNN Architectures

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.

D.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.

D.3 Attack Setting

The target class on all datasets was 1 for BadNets [5], the backdoor attack with blended strategy [2] (Blended), the invisible sample-specific attack [14] (ISSBA), and the input-aware dynamic attack [17] (IAD). The target classes for the label-consistent attack [23] and WaNet [18] were 2 and 0, respectively, on all datasets. The trigger patterns of attacks were the same as those presented in the main

paper. In particular, we set the blended ratio to $\lambda = 0.1$ for the blended attack on all datasets. We used the label-consistent backdoor attack 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 [18]. For IAD [17], 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.

D.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 [3]. 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 [13]. 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 [4]. 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 [22], 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.