Table of Content for Appendix

A Related Work about Trojan Attacks

In this paper, we mainly consider a class of Trojan attacks in which attackers fully control the training processes of a classifier. We refer to these attacks as *"full-control"* attacks. There is another less common type of Trojan attacks called *"clean-label"* attacks [37,41,55]. These attacks assume a scenario in which people want to adapt a popular pretrained classifier `C` (e.g., ResNet [14]) for their tasks by retraining the top layers of `C` with additional data collected from the web. The goal of an attacker is to craft a poisoning image $\tilde{x}$ that looks visually

Dataset	#Classes	Image size	Attack		Defense	
			#Train	#Test	#Train	#Test
MNIST	10	28×28×1	60000	10000	7000	3000
CIFAR10	10	32×32×3	50000	10000	7000	3000
GTSRB	43	32×32×3	39209	12630	8826	3804
CelebA	8	64×64×3	162770	19867	13904	5963

Table 4: Datasets used in our experiments.

indistinguishable from an image x_t of the target class t while being close to some source image x_s in the feature space by optimizing the following objective:

$$\tilde{x} = \underset{x}{\operatorname{argmin}} \ \|\mathtt{C}_{\mathrm{f}}(x) - \mathtt{C}_{\mathrm{f}}(x_s)\|_2^2 + \lambda \|x - x_t\|_2^2$$

where $\mathtt{C}_{\mathrm{f}}(\cdot)$ denotes the output of the penultimate layer of $\mathtt{C}$. The attacker then puts $\tilde{x}$ on the web so that it can be collected and labeled by victims. Since $\tilde{x}$ looks like an image of the target class t, $\tilde{x}$ will be labeled as t. When the victims retrain $\mathtt{C}$ using a dataset containing $\tilde{x}$, $\tilde{x}$ will create a *"backdoor"* in $\mathtt{C}$. The attacker can use x_s to access this *"backdoor"* and forces $\mathtt{C}$ to output t. Apart from the advantage that the attacker does not need to control the labeling process (while in fact, he can't), the clean-label attack has several drawbacks due to its impractical assumptions. For example, the victims may retrain the whole $\mathtt{C}$ instead of just the last softmax layer of $\mathtt{C}$; the victims may use their own training data to which the attacker can't access; the victims may use $\mathtt{C}$ for a completely new task that the attacker does not know; the attacker doesn't even know who are the victims. Moreover, since x_s often looks very different from images of the target class t, x_s can be easily detected by human inspection at test time.

Compared to clean-label attacks, full-control attacks are much harder to defend against because attackers have all freedom to do whatever they want with $\mathtt{C}$ before sending it to the victims. BadNet [11], Blended Injection [4] are among the earliest attacks [16,25] of this type that use only one global trigger and use image blending as an injection function. These attacks can be mitigated by well-known defenses like Neural Cleanse [47] or STRIP. Besides, their triggers also look unnatural. Therefore, subsequent attacks focus mainly on improving the robustness and stealthiness of triggers at test time. Some attacks use dynamic and/or input-specific triggers [21,32,33,38]. Others use more advanced injection functions [24,33] or GANs [31] to create hidden triggers or use physical objects as triggers [4,49].

B Experimental Setup

B.1 Datasets

We provide details of the datasets used in our experiments in Table 4. The training and test sets for defense ($\mathcal{D}_{\mathrm{val}}$ and $\mathcal{D}_{\mathrm{test}}$) are taken from the test set for attack with the #training/#test ratio of 0.7/0.3.

Encoder	Encoder
ConvBlockX(1, 16)	ConvBlockY(3, 64)
ConvBlockX(16, 32)	ConvBlockY(64, 128)
ConvBlockX(32, 64)	ConvBlockY(128, 256)
Reshape [64, 3, 3] to [576]	ConvBlockY(256, 512)
LinearBlockX(576, 256)	

Decoder	Decoder
LinearBlockX(256, 576)	DeconvBlockY(512, 512)
Reshape [576] to [64, 3, 3]	DeconvBlockY(512, 256)
DeconvBlockX(64, 32)	DeconvBlockY(256, 128)
DeconvBlockX(32, 16)	DeconvBlockY(128, 3)
DeconvBlockX(16, 1)	

MNIST CelebA

Table 5: Architectures of `F` (in AIF) for MNIST and CelebA.

B.2 Model Architectures and Training Settings for Benchmark Attacks

Model architectures Architectures of the classifier `C` in our work follow exactly those in [32,33]. Specifically, we use PreactResNet18 [15] for CIFAR10 and GTSRB, ResNet18 [14] for CelebA, and the convolutional network described in [32] for MNIST.

Training settings We train Input-Aware Attack and WaNet based on the official implementations of the two attacks[4][5] with the same settings as in the original papers [32,33]. We implement and train BadNet+, noise/image-BI+ ourselves. The settings of these attacks are given in Appdx. C.1. We set the poisoning rate of 0.1 for all the benchmark attacks.

B.3 Model Architectures and Training Settings for Our Defenses

Model architectures In AIF, `F` is a plain autoencoder. We use the two architectures in Table 5 for `F` when working on MNIST and CelebA and the architecture (C) in Table 7 when working on CIFAR10 and GTSRB. The remaining architectures in Table 7 (A, B, D) are for our ablation study in Appdx. E.3. The architecture of `G` is derived from the decoder of `F` with additional layers to handle the noise vector ϵ. ϵ has a fixed length of 128. The symbols in Tables 5, 6, 7, 8 have the following meanings: c_i is input channel, c_o is output channel, k is

[4] Input-Aware Attack: `https://github.com/VinAIResearch/input-aware-backdoor-attack-release`

[5] WaNet: `https://github.com/VinAIResearch/Warping-based_Backdoor_Attack-release`

LinearBlockX(d_i ,d_o)
Linear(d_i, d_o, b=False)
BatchNorm2d(d_o, m=0.01)
ReLU()

ConvBlockX(c_i, c_o)	DeconvBlockX(c_i, c_o)
Conv2d(c_i, c_o, k=4, s=2, p=1, b=False)	ConvTranspose2d(c_i, c_o, k=4, s=2, p=1, p_o=1, b=False)
BatchNorm2d(c_o, m=0.01)	BatchNorm2d(c_o, m=0.01)
ReLU()	ReLU()

ConvBlockY(c_i, c_o)	DeconvBlockY(c_i, c_o)
Conv2d(c_i, c_o, k=4, s=2, p=1, b=False)	ConvTranspose2d(c_i, c_o, k=4, s=2, p=1, b=False)
BatchNorm2d(c_o, m=0.01)	BatchNorm2d(c_o, m=0.01)
LeakyReLU(0.2)	ReLU()

Table 6: Linear, convolutional, and deconvolutional blocks of the architectures in Table 5.

kernel size, s is stride, p is padding, p_o is output padding, m is momentum, and b is bias.

The architectures of `F` in VIF are adapted from those in AIF by changing the middle layer between the encoder and decoder to produce the latent mean μ_z and standard deviation σ_z that characterize $q_{\mathtt{F}}(z|x)$.

Training settings If not otherwise specified, we train the generator `G`, the filter `F`, and the parameterized triggers (m_i, p_i) using an Adam optimizer [17] (learning rate = $1e^{-3}$, $\beta_1 = 0.5$, $\beta_2 = 0.9$) for 600 epochs with batch size equal to 128. For trigger synthesis (Section 3), in Eqs. 2, the norm is L2, $\delta = 0$, λ_0 varies from $1e^{-3}$ to 1 with the multiplicative step size ≈ 0.3. For VIF (Section 4.1), in Eq. 3, the norm is L2, $\lambda_1 = 1.0$ and $\lambda_2 = 0.003$. An analysis of different values of λ_2 is provided in Section 5.4.1. For AIF (Section 4.2), `F` and `G` are optimized alternately with the learning rate for `G` is $3e^{-4}$. In Eq. 8, $\delta = 0.05$, $\lambda_0 = 0.01$, $\lambda_3 = 0.3$. In Eq. 9, $\lambda_1 = 0.1$, $\lambda_4 = 0.3$, $\lambda_5 = 0.01$. To ensure that `G` and `F` are in good states before adversarial learning is conducted, we pretrain `G` and `F` for 100 epochs each. The pretraining losses for `G` and `F` are the terms $\mathcal{L}_{\text{gen}}$ in Eq. 8 and $\mathcal{L}_{\text{IF}}$ in Eq. 9, respectively. The training data in $\mathcal{D}_{\text{val}}$ are augmented with random flipping, random crop (padding size = 5), and random rotation (degree = 10).

C Additional Results of Benchmark Attacks

C.1 BadNet+ and Noise/Image-BI+

BadNet+ *BadNet+* is a variant of BadNet [11] that uses M different image patches $p_0, ..., p_{M-1}$ ($p_m \in \mathbb{I}^{c \times h_p \times w_p}$, $0 \leq m < M$) as Trojan triggers. Each

Encoder	Encoder	Encoder	Encoder
ConvBlockA(3, 32, k=5, s=1)	ConvBlockB(3, 32)	ConvBlockC(3, 32)	ConvBlockC(3, 32) $\rightarrow z_1$
ConvBlockA(32, 64, k=4, s=2)	ConvBlockB(32, 32)	MaxPool2d(2, 2)	MaxPool2d(2, 2)
ConvBlockA(64, 128, k=4, s=1)	MaxPool2d(2, 2)	ConvBlockC(32, 64)	ConvBlockC(32, 64) $\rightarrow z_2$
ConvBlockA(128, 256, k=4, s=2)	ConvBlockB(32, 64)	MaxPool2d(2, 2)	MaxPool2d(2, 2)
ConvBlockA(256, 512, k=4, s=1)	ConvBlockB(64, 64)	ConvBlockC(64, 128)	ConvBlockC(64, 128) $\rightarrow z$
ConvBlockA(512, 512, k=1, s=1)	MaxPool2d(2, 2)		
Linear(512, 256)	ConvBlockB(64, 128)		
	ConvBlockB(128, 128)		
	MaxPool2d(2, 2)		
	ConvBlockB(128, 128)		

Decoder	Decoder	Decoder	Decoder
DeconvBlockA(256, 256, k=4, s=1)	UpsamplingBilinear2d(2)	UpsamplingBilinear2d(2)	ConvTranspose2d(128, 64, k=2, s=2) $\rightarrow y_2$
DeconvBlockA(256, 128, k=4, s=2)	ConvBlockB(128, 128)	ConvBlockC(128, 64)	Concat($[y_2, z_2]$)
DeconvBlockA(128, 64, k=4, s=1)	ConvBlockB(128, 64)	UpsamplingBilinear2d(2)	ConvBlockC(128,64)
DeconvBlockA(64, 32, k=4, s=2)	UpsamplingBilinear2d(2)	ConvBlockC(64, 32)	ConvTranspose2d(64, 32, k=2, s=2) $\rightarrow y_1$
DeconvBlockA(32, 32, k=5, s=1)	ConvBlockB(64, 64)	Conv2d(32, 3, k=1)	Concat($[y_1, z_1]$)
DeconvBlockA(32, 32, k=1, s=1)	ConvBlockB(64, 32)		ConvBlockC(64,32)
ConvTranspose2d(32, 3, k=1, s=1)	UpsamplingBilinear2d(2)		Conv2d(32, 3, k=1)
	ConvBlockB(32, 32)		
	Conv2d(32, 3, k=3, p=1)		
(A)	(B)	(C)	(D)

Table 7: Architectures of `F` (in AIF) for CIFAR10 and GTSRB.

ConvBlockA(c_i, c_o, k, s)	DeconvBlockA(c_i, c_o, k, s)	ConvBlockB(c_i ,c_o)	ConvBlockC(c_i, c_o)
Conv2d(c_i, c_o, k, s)	ConvTranspose2d(c_i, c_o, k, s)	Conv2d(c_i, c_o, k=3, p=1)	Conv2d(c_i, c_o, k=3, p=1)
BatchNorm2d(c_o, m=0.1)	BatchNorm2d(c_o, m=0.1)	BatchNorm2d(c_o, m=0.05)	ReLU()
LeakyReLU()	LeakyReLU()	ReLU()	BatchNorm2d(c_o, m=0.01)
			Conv2d(c_o, c_o, k=3, p=1)
			ReLU()
			BatchNorm2d(c_o, m=0.01)

Table 8: Convolutional and deconvolutional blocks of the architectures in Table 7.

patch p_m is associated with a 2-tuple $l_m = (i, j)$ specifying the location of this patch in an input image, where $0 \leq i < h - h_p$ and $0 \leq j < w - w_p$. The pixel values and locations of the patches are generated randomly during construction. If not otherwise specified, we set $M = 20$ and set the patch size $h_p \times w_p$ to be 5×5 for MNIST, CIFAR10, GTSRB, and 8×8 for CelebA.

Blended Injection+ Blended Injection+ (BI+) is similar to BadNet+ except that it uses *full-size images* $\rho_0, ..., \rho_{M-1}$ ($\rho_m \in \mathbb{I}^{c \times h \times w}$, $0 \leq m < M$) as triggers instead of patches. Given a clean image x and a Trojan-triggering image ρ_m, the corresponding Trojan image $\tilde{x}$ is computed as follows:

$$\tilde{x} = (1 - \alpha) \cdot x + \alpha \cdot \rho_m$$

where α is the blending ratio set to 0.1 by default.

$\rho_0, ..., \rho_{M-1}$ can be either random noises or real images, resulting in two sub-versions of BI+, namely *noise-BI+* and *image-BI+*. Choosing good Trojan-triggering images for image-BI+ is non-trivial. We tried various real images (Fig. 7) and found that they lead to very different attack success rates (aka *Trojan accuracies*) (Fig. 8). The best ones often contain colorful, repetitive patterns (e.g., *"candies"* or *"crayons"* images). Besides, we also observed that training image-BI+ with $M \geq 10$ is difficult since the classifier usually needs a lot of time

Fig. 7: A list of Trojan-triggering images that we tried. The images are sorted by their training Trojan accuracy in ascending order. The last 5 images (11-15) were selected to be triggers for image-BI+ in our experiments.

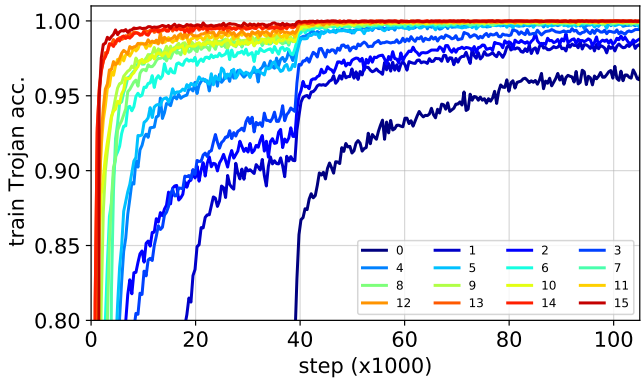


Fig. 8: Training Trojan accuracy curves of BI on CIFAR10 w.r.t. different triggers in Fig. 7

to remember real images. Therefore, we selected only 5 images with the highest training Trojan accuracies (images 11-15 in Fig. 7) to be used as triggers for image-BI+ in our experiments.

noise-BI+, by contrast, achieves almost perfect Trojan accuracies even when M is big ($M \approx 100$). We think the main reason behind this phenomenon is that a random noise image usually have much more distinct patterns than a real image. Although blending with clean input images may destroy some patterns in the noise image, many other patterns are still unaffected and can successfully cause the classifier to output the target class.

C.2 Results of Benchmark Attacks on $\mathcal{D}_{\text{test}}$

For completeness, we provide results of the benchmark attacks on $\mathcal{D}_{\text{test}}$ in Table 9 (for single-target mode) and in Table 10 (for all-target mode). The results in Table 9 are quite similar to the results on $\mathcal{D}_{\text{val}} \cup \mathcal{D}_{\text{test}}$ in Table 1. Since we were

Dataset	Benign	BadNet+		noise-BI+		image-BI+		InpAwAtk		WaNet	
	Clean	Clean	Trojan	Clean	Trojan	Clean	Trojan	Clean	Trojan	Clean	Trojan
MNIST	99.57	99.47	99.97	99.35	100.0	99.37	100.0	99.33	99.30	99.50	99.07
CIFAR10	94.71	94.83	100.0	94.57	100.0	95.13	99.90	94.57	99.40	94.27	99.67
GTSRB	99.63	99.42	100.0	99.45	100.0	99.34	100.0	98.97	99.66	99.16	99.42
CelebA	78.82	79.57	100.0	78.32	100.0	78.82	100.0	78.17	100.0	78.18	99.97

Table 9: Clean and Trojan accuracies of *single-target* attacks on the defense test set ($\mathcal{D}_{\text{test}}$) of different datasets.

Dataset	BadNet+		noise-BI+		image-BI+		InpAwAtk	
	Clean	Trojan	Clean	Trojan	Clean	Trojan	Clean	Trojan
MNIST	99.37	98.54	99.57	99.38	99.53	99.25	99.23	97.64
CIFAR10	94.63	94.30	94.32	93.77	94.70	94.06	94.53	94.10
GTSRB	99.63	99.08	99.58	99.06	99.37	99.08	99.16	99.29

Table 10: Clean and Trojan accuracies of different *all-target* attacks on the defense test set ($\mathcal{D}_{\text{test}}$) of different datasets.

not successful in training the all-target version of WaNet, we exclude this attack from the results in Table 10.

D Additional Results of Baseline Defenses

D.1 Network Pruning

We provide the Trojan accuracies of Network Pruning (NP) [22] at 1%, 5%, and 10% decrease in clean accuracy in Table 11 and the corresponding pruning curves in Fig. 9. It is clear that NP is a very ineffective Trojan mitigation method since the classifier pruned by NP still achieves nearly 100% Trojan accuracies on CIFAR10, GTSRB, and CelebA even when experiencing about 10% decrease in clean accuracy.

D.2 STRIP

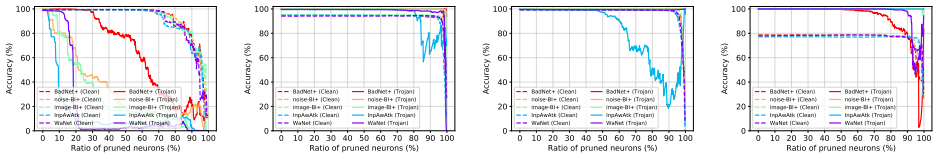
In Table 12, we report the false negative rates (FNRs) of STRIP at 1%, 5%, and 10% false positive rate (FPR). We also provide the AUCs and the entropy histograms of STRIP in Figs. 10 and 11, respectively. Note that AUCs are only suitable for experimental purpose not practical use since in real-world scenarios, we still have to compute thresholds based on FPRs on clean data. STRIP achieves very high FNRs on MNIST, CIFAR10, and GTSRB when defending against InpAwAtk and WaNet (Table 12) which corresponds to low AUCs (Fig. 10).

D.3 Neural Cleanse

After classifying `C` as Trojan-infected, Neural Cleanse (NC) mitigates Trojans via pruning `C` or via input checking. We refer to these two methods as *Neural Cleanse*

Dataset	BadNet+			noise-BI+			image-BI+			InpAwAtk			WaNet		
	1%	5%	10%	1%	5%	10%	1%	5%	10%	1%	5%	10%	1%	5%	10%
MNIST	37.38	22.73	14.98	14.21	11.00	6.75	14.69	6.31	4.72	3.32	3.32	3.32	1.18	1.18	1.18
CIFAR10	100.0	100.0	100.0	99.33	87.26	87.26	99.74	99.74	99.74	56.52	56.52	56.52	96.85	96.59	96.59
GTSRB	100.0	100.0	100.0	100.0	100.0	100.0	99.87	99.87	99.87	18.38	18.38	18.38	98.86	98.86	98.86
CelebA	84.08	63.35	50.64	100.0	99.97	99.97	99.87	99.87	99.80	100.0	100.0	99.97	98.22	88.48	53.10

Table 11: Trojan accuracies of `C` pruned by Network Pruning at 1%, 5%, and 10% decrease in clean accuracy. *Smaller values are better*. Results are computed on $\mathcal{D}'_{\text{test}}$.



(a) MNIST (b) CIFAR10 (c) GTSRB (d) CelebA

Fig. 9: Clean accuracy (dashed) and Trojan accuracy (solid) curves of `C` pruned by Network Pruning for different attacks and datasets which correspond to the results in Table 11.

Pruning (NCP) and *Neural Cleanse Input Checking (NCIC)*. Both methods build a set of synthetic Trojan images by blending all clean images in $\mathcal{D}_{\text{val}}$ with the synthetic trigger corresponding to the detected target class. NCP ranks neurons in the second last layer of `C` according to their average activation gaps computed on the synthetic Trojan images and the corresponding clean images in $\mathcal{D}_{\text{val}}$ in descending order. It gradually prunes the neurons with the highest ranks first until certain decrease in clean accuracy is met. NCIC, on the other hand, picks the top 1% of the neurons in the second last layer of `C` with largest average activations on the synthetic

Trojan images to form a characteristic group of Trojan neurons. Given an input image x, NCIC considers the mean activations of the neurons in the group w.r.t. x as a score for detecting whether x contains Trojan triggers or not. If the score is greater than a threshold, x is considered as a Trojan image, otherwise, a clean image. The threshold is chosen based on the scores of all clean images in $\mathcal{D}_{\text{val}}$. We provide the results of NCP in Table 13, Fig. 12 and the results of NCIC in Table 14. At 5% decrease in clean accuracy, NCP reduces the Trojan accuracies of all the attacks except WaNet to almost 0% on MNIST and CIFAR10. However, NCP is ineffective against these attacks especially image-BI+ and InpAwAtk on GTSRB and CelebA. At 10% FPR, NCIC achieves nearly perfect FNRs against BadNet+, noise-BI+, and image-BI+ on all datasets but is also ineffective against InpAwAtk on GTSRB and CelebA. Note that both NCP and NCIC have almost no effect against WaNet on MNIST, CIFAR10, and CelebA since NC misclassifies the Trojan classifiers w.r.t. this attack as benign.

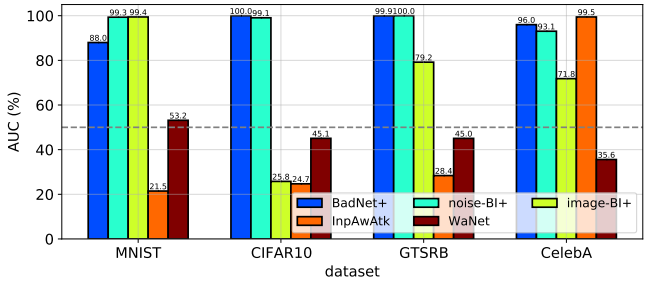


Fig. 10: AUCs of STRIP against different attacks on different datasets. *Larger values are better.*

Dataset	BadNet+			noise-BI+			image-BI+			InpAwAtk			WaNet		
	1%	5%	10%	1%	5%	10%	1%	5%	10%	1%	5%	10%	1%	5%	10%
MNIST	88.45	64.40	36.25	13.95	0.00	0.00	16.40	0.05	0.00	99.95	99.80	99.15	99.85	97.55	92.40
CIFAR10	0.00	0.00	0.00	10.35	2.35	0.95	99.45	97.30	95.75	99.55	97.90	96.20	100.0	99.25	96.90
GTSRB	1.05	0.20	0.00	0.00	0.00	0.00	79.20	58.30	44.75	99.85	98.80	97.15	99.50	96.00	91.65
CelebA	17.50	11.20	7.70	33.80	19.80	14.70	75.75	62.35	54.45	1.90	1.20	1.00	99.80	98.55	96.20

Table 12: False negative rates (FNRs) of STRIP at 1%, 5%, and 10% false positive rate (FPR) for different attacks and datasets. *Smaller values are better.*

D.4 Februus

We reimplement Februus based on the official code provided by the authors[6]. Since there is no script for training the inpainting GAN in the authors' code, we use the "inpaint" function from OpenCV instead. Februus has 2 main hyperparameters that need to be tuned which are: i) the convolutional layer of `C` at which GradCAM computes heatmaps (*"heatmap layer"* for short) and ii) the threshold for converting GradCAM heatmaps into binary masks (*"binary threshold"* for short). As shown in Fig. 14, the performance of Februus greatly depends on these hyperparameters. Increasing the binary threshold means smaller areas are masked and inpainted, which usually leads to smaller decreases in clean accuracy (smaller ↓Cs) yet higher Trojan accuracies (higher Ts). Meanwhile, choosing top layers of `C` to compute heatmap (e.g., layer4) usually causes bigger ↓Cs yet lower Ts since the selected regions are often broader (Fig. 13). For simplicity, we choose the (layer, threshold) setting that gives the smallest decrease in recovery accuracy (↓R) of Februus when defending against BadNet+ and apply this setting to all other attacks.

D.5 Neural Attention Distillation

We reimplement Neural Attention Distillation (NAD) based on the official code provided by the authors[7]. We finetune the original Trojan classifier `C` to obtain

[6] Februus: `https://github.com/AdelaideAuto-IDLab/Februus`
[7] NAD: `https://github.com/bboylyg/NAD`

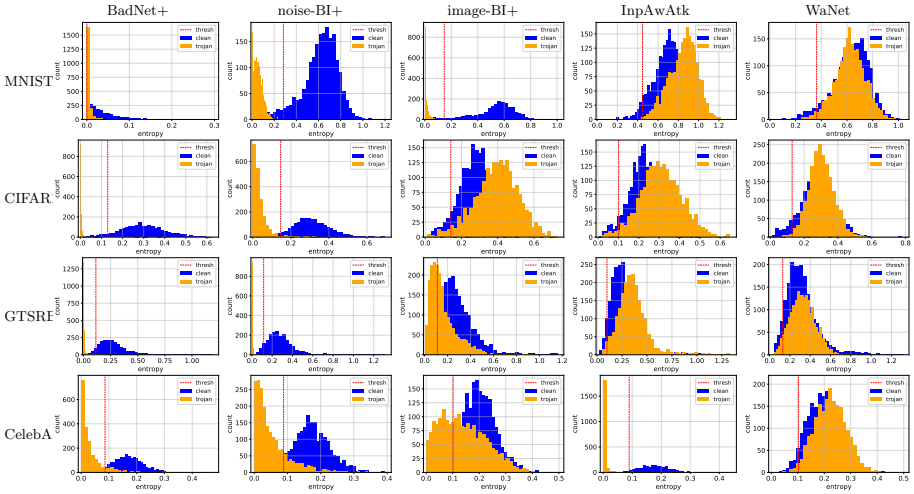
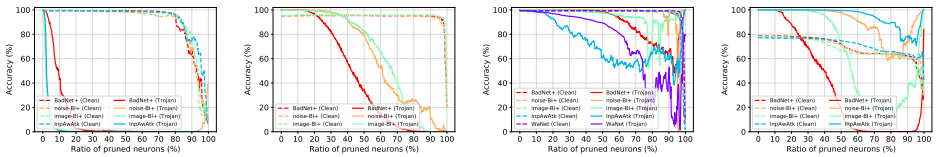


Fig. 11: Histograms of entropies computed by STRIP for different attacks and datasets. The vertical red dashed line in each plot indicates the threshold at 5% false positive rate.

Dataset	BadNet+			noise-BI+			image-BI+			InpAwAtk			WaNet		
	1%	5%	10%	1%	5%	10%	1%	5%	10%	1%	5%	10%	1%	5%	10%
MNIST	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-	-	-
CIFAR10	0.00	0.00	0.00	7.37	1.11	0.70	0.00	0.00	0.00	-	-	-	-	-	-
GTSRB	61.94	48.23	48.23	78.71	1.03	1.03	51.74	51.74	51.74	49.02	38.33	38.33	2.75	1.58	1.58
CelebA	75.97	26.73	7.63	99.95	98.75	95.30	99.75	96.21	83.62	99.87	98.42	93.31	-	-	-

Table 13: Trojan accuracies of the Trojan classifier `C` pruned by Neural Cleanse Pruning at 1%, 5%, and 10% decrease in clean accuracy for different attacks and datasets. *Smaller values are better*. Some results for InpAwAtk and WaNet are not available because Neural Cleanse fails to classify `C` as Trojan-infected in these cases (Fig. 4a).

the teacher `T` in 10 epochs. After that, we distill knowledge from `T` to `C` in 20 more epochs. In both cases, the batch size is 64 and the optimizer is Adam with an learning rate of 0.0001 divided by 10 after 10 epochs. We note that in the original paper, the authors reported that they used an initial learning rate of 0.1 and divided it by 10 after every 2 epochs during knowledge distillation. However, in our experiment, we found that such an initial learning rate is too large for distillation and can cause a significant drop in the clean accuracy of `C` which is hard to be recovered even if the learning rate is decayed later.



(a) MNIST (b) CIFAR10 (c) GTSRB (d) CelebA

Fig. 12: Clean accuracy (dashed) and Trojan accuracy (solid) curves of the Trojan classifier `C` pruned by Neural Cleanse Pruning for different attacks and datasets which correspond to the results in Table 13.

Dataset	BadNet+			noise-BI+			image-BI+			InpAwAtk			WaNet		
	1%	5%	10%	1%	5%	10%	1%	5%	10%	1%	5%	10%	1%	5%	10%
MNIST	9.75	4.75	3.25	0.00	0.00	0.00	0.00	0.00	0.00	0.80	0.30	0.15	-	-	-
CIFAR10	99.50	78.90	0.65	90.55	39.30	0.05	95.50	56.90	0.40	-	-	-	-	-	-
GTSRB	100.0	0.40	0.00	0.00	0.00	0.00	0.03	0.00	0.00	99.48	92.17	86.65	3.02	0.40	0.32
CelebA	2.25	0.00	0.00	17.70	0.65	0.00	50.20	11.60	2.35	88.25	42.15	13.60	-	-	-

Table 14: False negative rates (FNRs) computed by Neural Cleanse Input Checking at 1%, 5%, and 10% false positive rate (FPR) for different attacks and datasets. *Smaller values are better.*

E Additional Results and Ablation Studies of Our Defenses

E.1 Results of Our Defenses against All-target Attacks

In Tables 15 and 16, we show the results our filtering and FtC defenses against different *all-target* attacks. On CIFAR10 and GTSRB, VIF/VIFtC and AIF/AIFtC are comparable. However, on MNIST, AIF/AIFtC is clearly better than VIF/VIFtC.

E.2 Comparison between Input Filtering and Input Processing [25]

In Fig. 17, we compare the performances of our Input Filtering (IF) and Input Processing (IP) against all the benchmark attacks on all the datasets. It is clear that IF outperforms IP in terms of Trojan accuracy in most cases, especially under InputAwAtk and WaNet. For example, IP achieves very high (poor) Trojan accuracies of 34.85%, 62.33%, and 76.49% against WaNet on CIFAR10, GTSRB, and CelebA, respectively while our IF achieves only 4.82%, 9.83%, and 15.21%. These results empirically verify the importance of the term $-\log p_{\mathtt{C}}(y|x^{\circ})$ in the loss of IF. This term ensures that the filtered output x°computed by `F` cannot cause harm to `C` even when it look very similar to the original input x.

E.3 Different architectures of the filter `F`

A major factor that affects the performance of `F` is its architecture. We consider an architecture of `F` to be more complex than others if `F` achieves smaller re-

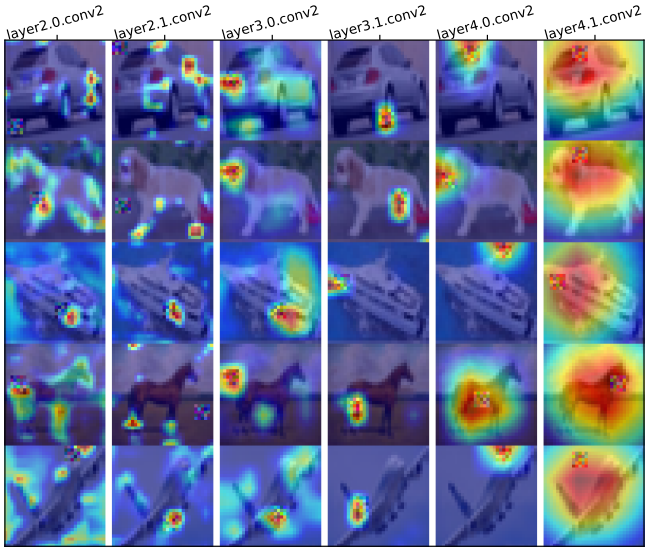
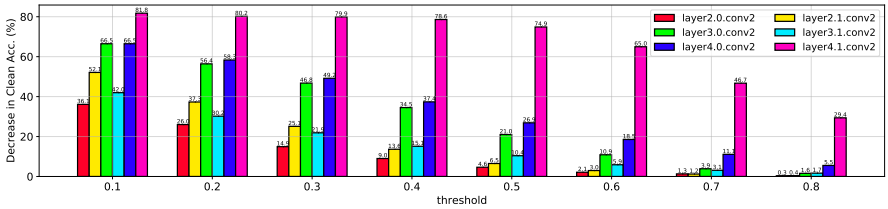


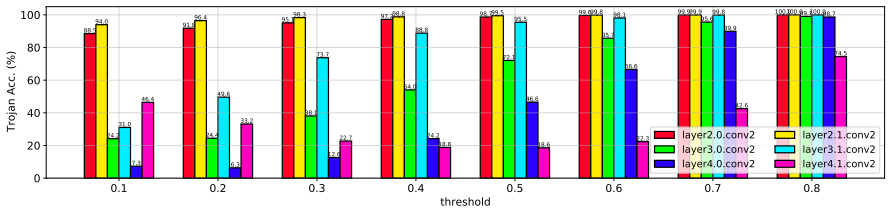
Fig. 13: Examples of heatmaps computed by Februus's GradCAM at different layers of `C` ordered from bottom (layer2.0.conv2) to top (layer4.1.conv2). The dataset is CIFAR10 and the classifier is a PreactResNet18 [15]. The Trojan attack is BadNet+. It is clear that only some layers give reasonably good results.

construction loss on $\mathcal{D}_{\text{val}}$ with this architecture. In Section 4, we argued that an optimal filter should be neither too simple nor too complex. Here, we empirically verify this intuition by examining 4 different architectures of `F` for CIFAR10 marked as A, B, C, D (Table 7). Their complexities are greater in alphabetical order as shown in Figs. 15a, 15b. The architecture D has skip-connections between its encoder and decoder while A, B, C do not.

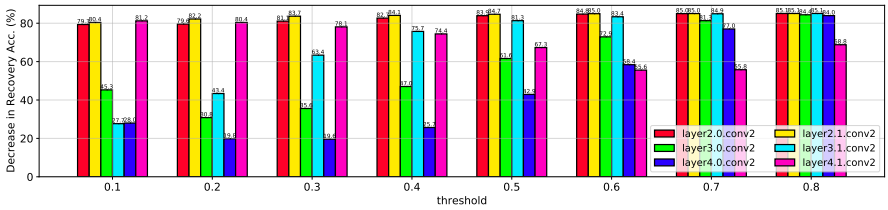
Denote `F` with the architectures A, B, C, D as $\texttt{F}_\text{A}$, $\texttt{F}_\text{B}$, $\texttt{F}_\text{C}$, $\texttt{F}_\text{D}$ respectively. As shown in Table 18, $\texttt{F}_\text{C}$ is the best in terms of recovery accuracy although C is neither the simplest (A) nor the most complex architecture (D). $\texttt{F}_\text{A}$ achieves reasonably low Trojan accuracies comparable to those of $\texttt{F}_\text{B}$ and $\texttt{F}_\text{C}$ but the worst clean accuracies. $\texttt{F}_\text{D}$, by contrast, experiences almost no decrease in clean accuracy but achieves very high Trojan accuracies. The reason is that $\texttt{F}_\text{D}$ simply copies all information from an input to the output via its skip-connections rather than learning compressed latent representations of input images. One can verify this by observing that the D_{KL} (Eq. 3) of $\texttt{F}_\text{D}$ is almost 0 (Fig. 16c). In this case, changing λ_2 has no effect on the Trojan accuracy of $\texttt{F}_\text{D}$ as shown in Fig. 17. However, it is still possible to lower the Trojan accuracy of $\texttt{F}_\text{D}$ without removing the skip-connections in D. For example, we can treat the latent representations corresponding to the skip-connections as random variables and apply the D_{KL} to these variables like what we do with the middle representation. Or we can com-



(a) Decrease in clean accuracy (↓C)



(b) Trojan accuracy (T)



(c) Decrease in recovery accuracy (↓R)

Fig. 14: Decreases in clean accuracy (a), Trojan accuracies (b), and decreases in recovery accuracy (c) of Februus when mitigating BadNet+'s Trojans on CIFAR10 w.r.t. different heatmap layers and binary thresholds.

press the whole $\mathtt{F}_{\mathrm{D}}$ by using advanced network compression techniques [30,27,1]. These ideas are out of scope of this paper so we leave them for future work.

E.4 Small amounts of training data

We are curious to know how well our proposed defenses will perform if we reduce the amount of training data. We select the proportion of training data from {0.8, 0.6, 0.4, 0.2, 0.1, 0.05}. In addition, we consider two broader training settings. In the first setting, the number of training epochs is fixed at 600 (Section B.3) regardless of the amount of training data. Because less training data results in fewer iterations per epoch, fixing the number of training epochs means *smaller total number of training iterations for less training data*. In the second setting, we adjust the number of training epochs based on the proportion of training data so that the total number of training iterations is fixed and similar to that when full data is used. Table 19 shows the results of our filtering defenses w.r.t.

Dataset	Defense	BadNet+			noise-BI+			image-BI+			InpAwAtk		
		↓C	T	↓R	↓C	T	↓R	↓C	T	↓R	↓C	T	↓R
MNIST	IF	0.00	75.90	76.30	-0.03	99.30	99.47	0.13	6.96	6.96	0.00	95.74	96.17
	VIF	**-0.07**	49.67	50.17	**-0.07**	4.06	4.23	0.07	0.20	0.23	**-0.23**	63.58	64.21
	AIF	0.47	19.31	**20.27**	0.13	**0.03**	**0.10**	**0.03**	**0.03**	**0.13**	-0.13	**23.90**	**24.53**
CIFAR10	IF	**3.93**	1.57	**8.03**	**2.63**	1.17	**3.30**	**3.60**	18.90	22.20	**3.83**	9.97	14.63
	VIF	8.13	1.87	12.73	6.27	1.33	6.72	6.83	5.67	11.90	7.96	**5.95**	16.87
	AIF	5.67	**1.23**	9.13	4.40	**0.97**	5.47	5.13	**5.10**	**9.63**	5.43	6.73	**14.33**
GTSRB	IF	0.29	**0.11**	**1.58**	0.13	**0.21**	**1.18**	**-0.26**	61.57	62.25	0.11	3.97	4.50
	VIF	0.47	0.32	3.36	0.32	0.37	1.26	0.16	6.28	8.23	0.11	**0.60**	**1.58**
	AIF	**0.11**	0.29	2.08	**-0.05**	0.29	1.74	0.11	**1.21**	**3.23**	**-0.16**	2.02	2.39

Table 15: Decreases in clean accuracy (↓Cs), Trojan accuracies (Ts), and recovery accuracies (↓Rs) of our filtering defenses (IF, VIF, AIF) against different *all-target* Trojan attacks on different datasets. *Smaller values are better*. For a particular dataset, attack, and metric, the best defense is highlighted in bold.

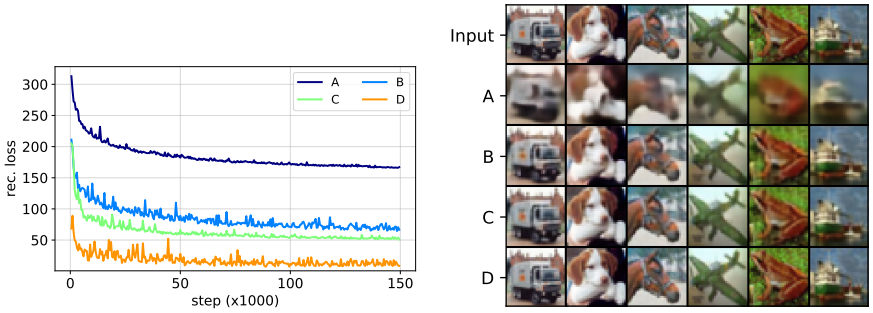
Dataset	Defense	BadNet+		noise-BI+		image-BI+		InpAwAtk	
		FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR
MNIST	IFtC	**0.20**	77.83	0.10	99.63	0.23	7.22	**0.20**	97.44
	VIFtC	**0.20**	49.97	**0.07**	4.19	**0.07**	0.27	0.33	64.85
	AIFtC	0.70	**19.24**	0.27	**0.07**	0.30	**0.07**	0.40	**24.83**
CIFAR10	IFtC	**7.13**	1.00	**6.10**	**0.70**	**6.70**	18.57	**6.30**	9.97
	VIFtC	12.10	1.30	10.14	1.12	10.50	**5.40**	11.13	**5.83**
	AIFtC	9.07	**0.87**	8.03	1.10	8.47	**5.40**	8.90	6.70
GTSRB	IFtC	0.50	**0.08**	0.29	0.29	**0.42**	62.57	**0.34**	3.97
	VIFtC	0.68	0.32	0.58	0.47	0.84	6.15	0.58	**0.55**
	AIFtC	**0.34**	0.42	**0.26**	**0.26**	0.79	**0.92**	0.45	2.05

Table 16: FPRs and FNRs of our FtC defenses against different *all-target* attacks. *Smaller values are better*. For a particular dataset, attack, and metric, the best defense is highlighted in bold.

the above settings. When the number of epochs is fixed, we see that our defenses often achieve lower Ts yet larger ↓Cs (and ↓Rs) for less training data. This is because the filter `F` has not been fully trained to reconstruct the input image well enough (the first column in Fig. 18). On the other hand, when the total number of iterations is fixed, `F` has been fully trained and we do not see much difference in Trojan accuracy of VIF for different amounts of training data. The Trojan accuracy of AIF slightly increase for less training data but is still acceptable (the last column in Fig. 18). Changes in clean accuracy of our filtering defenses are small (the third column in Fig. 18). In summary, these results suggest that our filtering defenses are quite robust to the limited amount of training data.

Dataset	Def.	BadNet+			noise-BI+			image-BI+			InpAwAtk			WaNet		
		↓C	T	↓R	↓C	T	↓R	↓C	T	↓R	↓C	T	↓R	↓C	T	↓R
MNIST	IP	0.37	**1.29**	**3.60**	**0.03**	3.69	**10.92**	**0.13**	1.81	**11.72**	0.30	1.14	2.10	**0.20**	1.66	1.70
	IF	***0.27***	*2.47*	*4.99*	*0.10*	***0.16***	*13.52*	***0.13***	***1.29***	*12.02*	***0.21***	***0.96***	***2.08***	*0.23*	***0.34***	***0.61***
CIFAR10	IP	4.23	2.74	8.60	3.60	**0.78**	4.97	**4.27**	**35.85**	**33.60**	5.20	20.48	22.80	5.37	34.85	30.37
	IF	***4.15***	***2.30***	***7.79***	***3.32***	*1.01*	***4.43***	*4.76*	*37.48*	*34.30*	***4.47***	***16.35***	***18.96***	***3.21***	***4.82***	***6.80***
GTSRB	IP	0.18	**0.00**	2.79	0.24	**0.00**	1.76	**0.37**	52.61	52.37	0.42	2.06	3.76	10.73	62.33	61.33
	IF	***0.13***	***0.00***	***2.55***	***0.13***	*0.03*	***1.52***	***0.37***	***52.27***	***51.95***	***0.03***	***0.66***	***3.60***	***0.08***	***9.83***	***9.62***
CelebA	IP	**2.83**	9.98	4.23	2.73	25.57	14.72	2.43	73.63	35.82	**2.31**	15.09	7.81	**2.28**	76.49	36.76
	IF	*4.21*	***8.62***	***4.75***	***2.57***	***13.83***	***6.00***	***2.25***	***59.39***	***27.94***	*2.86*	***11.95***	***6.07***	*2.43*	***15.21***	***4.75***

Table 17: Trojan filtering results (in %) of Input Processing [25] and our Input Filtering. For a particular dataset, attack, and metric, the best among the 2 defenses are highlighted in bold. Results taken from Table 2 are shown in italic.



(a) Reconstruction losses (b) Reconstructed images

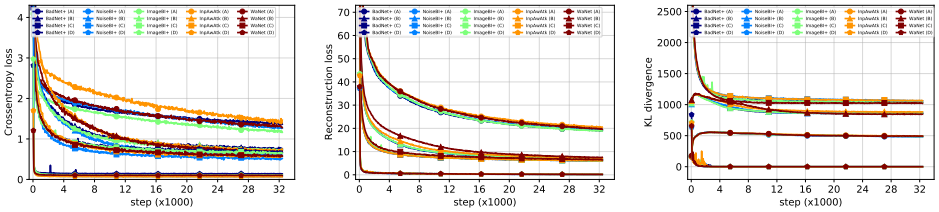
Fig. 15: Reconstruction losses (a) and reconstructed images (b) on CIFAR10 corresponding to the 4 autoencoders described in Table 7.

E.5 Large-norm triggers

In this section, we examine the performances of our defenses against attacks that use large-norm triggers. We consider BadNet+ and noise-BI+ for this study. For BadNet+, we increase the trigger norm by increasing the trigger size s. The results for BadNet+ are shown in Table 20. For noise-BI+, we increase the trigger norm by increasing the blending ratio α. The results for noise-BI+ are shown in Table 21. It is clear that even when triggers have large norms, our filtering defenses, especially VIF, still effectively erase most of the trigger pixels that could activate the Trojans in `C` (Fig. 25) and achieve *low Trojan accuracies* (Tables. 20, 21). However, large-norm triggers cause a lot of difficulty in reconstructing the original clean images from Trojan images (Fig. 25), which leads to large decreases in recovery accuracy of our methods. Note that such poor performance is inevitable for filtering defenses like VIF and AIF. A solution to this problem is using other types of defenses. STRIP [8], Neural Cleanse [47], and NAD [20] are possible yet not good options. As shown in Fig. 19, STRIP works well against BadNet+ with large trigger sizes but poorly against noise-BI+ with

Arch.	BadNet+			noise-BI+			image-BI+			InpAwAtk			WaNet		
	↓C	T	↓R	↓C	T	↓R	↓C	T	↓R	↓C	T	↓R	↓C	T	↓R
A	40.02	5.96	41.10	40.90	14.52	42.43	35.73	**3.56**	37.92	38.13	3.52	38.97	32.65	6.81	32.94
B	9.23	2.74	13.37	8.73	1.96	10.32	10.48	10.59	18.87	9.75	**1.96**	13.91	9.94	4.22	11.73
C	7.70	**2.52**	**11.27**	6.43	**1.22**	**7.10**	7.53	10.52	**16.50**	7.67	3.07	**12.38**	7.97	**3.96**	**10.67**
D	**0.27**	100.0	84.83	**-0.02**	100.0	84.57	**0.21**	99.81	84.96	**-0.10**	98.85	83.33	**-0.07**	98.96	83.35

Table 18: Trojan filtering results (in %) of VIF against different attacks on CIFAR10 w.r.t. different architectures of `F`. For a particular dataset, attack, and metric, the best defense is highlighted in bold.



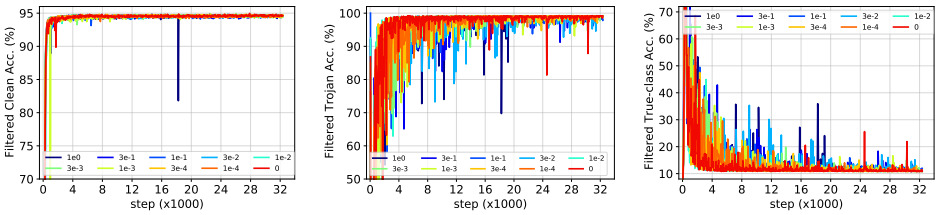
(a) Crossentropy loss (b) Reconstruction loss (c) KL divergence

Fig. 16: Training curves of VIF against InpAwAtk on CIFAR10 ($\mathcal{D}_{\text{val}}$) w.r.t. the 4 architectures of `F` in Table 7.

large blending ratios. We guess the reason is that with large blending ratios, Trojan images of noise-BI+ will look like noises, and superimposing a noise-like Trojan image with a clean image is like adding noise to the clean image, hence, won't affect of the class prediction of the clean image. Neural Cleanse tends to wrongly identify Trojan models as benign if the behind attacks use triggers with large enough norms. This is because the reverse-engineered trigger w.r.t. the true target class also has large norm which is not very different from the norms of the reverse-engineered triggers w.r.t. other classes. NAD achieves impracticably high Trojan accuracies when trigger norms are large for both BadNet+ and noise-BI+. By contrast, our FtC defenses, especially VIFtC, are *good alternatives*. VIFtC achieves very low FNRs (<7% in case of BadNet+ and <2% in case of noise-BI+) while still keeping FPRs in an acceptable range between 10% and 15% (Tables. 20, 21).

E.6 Explicit Normalization of Triggers in AIF

E.6.1 Theoretical Derivation During training AIF, we observed that under the influence of $\mathcal{L}_{\text{VIF-gen}}$ (Eq. 8), the norm $\|m\|$ of a synthetic trigger mask m usually increases overtime despite the fact that $\mathcal{L}_{\text{VIF-gen}}$ also contains a norm regularization term. The reason is that `G` is encouraged to output bigger Trojan triggers to fool `F`. However, too big trigger causes the generated Trojan image $\tilde{x}$ to be very different from the input image x, which affects the learning of `F`. One way to deal with this problem is *explicitly normalizing* m so that its norm



(a) Filtered Clean Acc. (b) Filtered Trojan Acc. (c) Filtered Recovery Acc.

Fig. 17: Test results of VIF using the architecture D (Table 7) for `F` against InpAwAtk on CIFAR10 ($\mathcal{D}_{\text{test}}$) w.r.t. different coefficient values for D_{KL} (λ_2 in Eq. 3).

Def.	Metric	InpAwAtk												
		Default	Fixed #epochs						Fixed total #iterations					
		1.0	0.8	0.6	0.4	0.2	0.1	0.05	0.8	0.6	0.4	0.2	0.1	0.05
IF	↓C	*4.47*	4.73	4.43	5.37	7.90	11.60	25.93	3.53	3.87	3.83	4.20	4.23	4.87
	T	*16.35*	15.11	11.56	9.48	6.70	4.07	6.70	16.11	15.41	20.22	18.37	26.33	21.00
	↓R	*18.96*	19.00	15.10	15.47	15.07	17.47	32.23	18.47	18.03	21.87	20.57	26.63	23.17
VIF	↓C	*7.67*	9.17	8.57	9.83	12.13	17.10	27.80	8.53	7.93	7.87	8.97	9.37	11.27
	T	*3.07*	3.81	2.81	4.30	3.30	2.70	5.67	3.26	4.19	4.26	3.70	3.44	4.67
	↓R	*12.38*	14.90	13.17	16.30	17.37	22.10	32.30	13.73	13.87	13.67	14.73	15.03	17.87
AIF	↓C	*5.28*	5.47	6.53	7.90	11.70	16.67	26.13	5.07	5.23	5.17	6.17	5.77	7.83
	T	*5.30*	4.96	3.59	4.89	3.15	2.89	4.00	6.63	9.04	5.70	4.56	8.48	12.96
	↓R	*11.87*	11.60	11.90	13.87	17.03	21.80	30.40	11.73	13.83	11.77	12.57	14.50	19.93

Table 19: Trojan filtering results (in %) of IF, VIF, and AIF against InpAwAtk on CIFAR10 w.r.t. different proportions of training data (the third row) and two broader training settings (the second row): i) fixed number of epochs, and ii) fixed total number of iterations. Results taken from Table 2 are shown in italic.

is always upper-bounded by δ. Denoted by $\bar{m}$ the δ-normalized version of m. $\bar{m}$ can be computed as follows:

$$\bar{m} = \begin{cases} m & \text{if } \|m\| \leq \delta \\ \delta \frac{m}{\|m\|} & \text{if } \|m\| > \delta \end{cases} \tag{11}$$

or more compactly,

$$\begin{aligned} \bar{m} &= m \times \left(1 - \frac{\max(\|m\| - \delta, 0)}{\|m\|}\right) \\ &= m \times \left(1 - \frac{\text{ReLU}(\|m\| - \delta)}{\|m\|}\right) \end{aligned} \tag{12}$$

In case the norm is L2, the second expression in Eq. 11 can be seen as the projection of m onto the surface of a sphere of radius δ. By replacing ReLU($\cdot$)

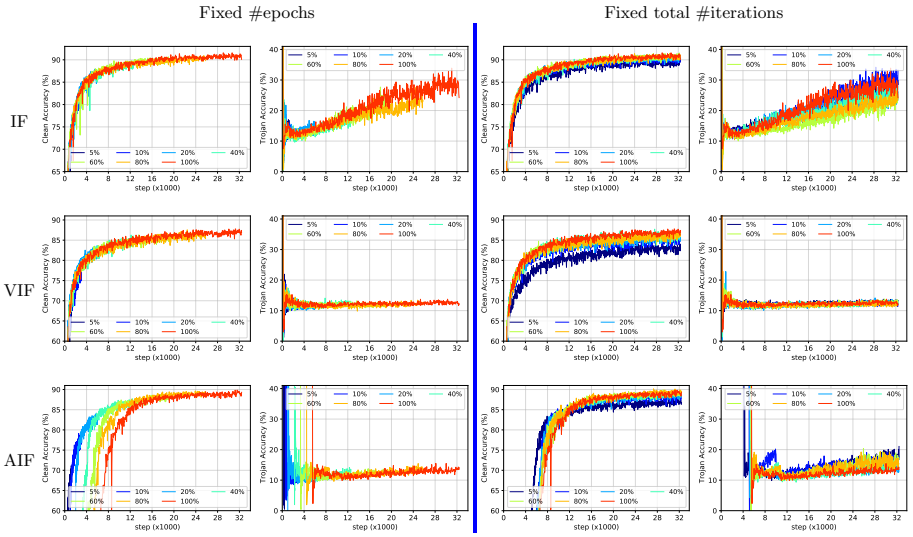


Fig. 18: Test clean accuracy and Trojan accuracy curves of our filtering defenses (IF, VIF, AIF) against InpAwAtk on CIFAR10 w.r.t. different proportions of training data and 2 broader training settings: i) fixed number of epochs and ii) fixed total number of iterations.

in Eq. 12 with Softplus$(\cdot)$, we obtain a soft version of $\bar{m}$:

$$\begin{aligned}\bar{m}_s &= m \times \left(1 - \frac{\text{Softplus}(\|m\| - \delta, \tau)}{\|m\|}\right) \\ &= m \times \left(1 - \frac{\tau \log(1 + \exp((\|m\| - \delta)/\tau))}{\|m\|}\right)\end{aligned} \tag{13}$$

where $\tau > 0$ is a temperature. $\bar{m}_s$ with smaller τ approximates $\bar{m}$ better. Generally, using $\bar{m}_s$ gives better gradient update than using $\bar{m}$. However, since Softplus is an upper-bound of ReLU, Softplus$(\|m\| - \delta, \tau)$ can be greater than $\|m\|$, which causes $\bar{m}_s$ to be negative. To avoid that, we slightly modify Eq. 13 into the equation below:

$$\bar{m}_s = m \times \left(1 - \frac{\tau \log(1 + \exp((\|m\| - \delta)/\tau))}{\|m\| + \Omega}\right) \tag{14}$$

Defense	BadNet+																			
	$s=5$					$s=11$					$s=17$					$s=23$				
	↓C	T	↓R	FPR	FNR	↓C	T	↓R	FPR	FNR	↓C	T	↓R	FPR	FNR	↓C	T	↓R	FPR	FNR
IF | IFtC	***4.15***	*2.30*	***7.79***	***7.47***	***1.70***	5.83	6.33	29.27	**8.83**	6.81	**4.17**	22.52	52.50	**6.83**	22.37	**4.30**	64.41	78.77	**7.47**	63.74
VIF | VIFtC	*7.70*	*2.52*	*11.27*	*11.00*	*2.63*	11.07	3.89	31.07	14.43	3.85	9.13	**5.41**	55.60	12.80	**4.89**	9.77	**7.22**	77.73	14.07	**6.81**
AIF | AIFtC	*5.60*	*2.37*	*9.03*	*8.87*	*1.96*	**5.53**	**3.33**	**23.97**	9.03	**3.15**	4.67	7.56	**48.40**	8.30	8.04	5.10	28.07	**77.70**	8.30	28.0
AIF* | AIFtC*	5.90	**2.15**	10.00	9.30	2.15	6.50	5.00	34.33	9.97	5.30	5.77	14.63	52.67	9.33	13.70	6.97	29.96	77.93	10.60	30.19

Table 20: Trojan filtering results (in %) of IF, VIF, AIF, and AIF without explicit trigger normalization (AIF*) against BadNet+ with different trigger sizes (s) on CIFAR10. For a particular trigger size and metric, the best result is highlighted in bold. Results taken from Tables 2, 3 are shown in italic.

Defense	noise-BI+																			
	$\alpha=0.1$					$\alpha=0.3$					$\alpha=0.5$					$\alpha=0.7$				
	↓C	T	↓R	FNR	FPR	↓C	T	↓R	FNR	FPR	↓C	T	↓R	FNR	FPR	↓C	T	↓R	FNR	FPR
IF/IFtC	***3.32***	*1.01*	***4.43***	***6.57***	*0.93*	**4.30**	1.26	36.73	**7.80**	1.19	**3.53**	27.59	73.03	**7.10**	26.74	**3.87**	49.37	83.10	**7.13**	49.00
VIF/VIFtC	*6.43*	*1.22*	*7.10*	*10.67*	*1.26*	7.53	1.44	27.57	11.80	1.56	7.67	**0.74**	**67.33**	11.63	**0.74**	7.93	**0.22**	80.80	11.57	**0.26**
AIF/AIFtC	*4.87*	*1.14*	*6.02*	*8.73*	*0.89*	4.90	1.33	39.50	8.60	1.37	5.30	7.00	72.30	9.10	7.11	6.07	31.41	**80.70**	8.87	31.67
AIF*/AIFtC*	5.60	**0.78**	7.50	9.53	**0.74**	6.07	**0.48**	45.13	10.10	**0.74**	6.27	1.48	74.83	10.67	1.78	6.47	38.74	80.97	10.00	39.56

Table 21: Trojan filtering results (in %) of IF, VIF, AIF, and AIF without explicit trigger normalization (AIF*) against noise-BI+ with different blending ratios (α) on CIFAR10. For a particular blending ratio and metric, the best result is highlighted in bold. Results taken from Tables 2, 3 are shown in italic.

where $\Omega > 0$ is an added term to ensure that $\bar{m}_s \geq 0$. Ω can be computed from τ and δ by solving the following inequality:

$$\begin{aligned}
&\tau \log(1+\exp((\|m\|-\delta)/\tau)) \leq \|m\| + \Omega \\
\Leftrightarrow &1+\exp\left(\frac{\|m\|-\delta}{\tau}\right) \leq \exp\left(\frac{\|m\|+\Omega}{\tau}\right) \\
\Leftrightarrow &1 \leq \exp\left(\frac{\|m\|}{\tau}\right)\left(\exp\frac{\Omega}{\tau} - \exp\left(\frac{-\delta}{\tau}\right)\right) \\
\Leftrightarrow &\exp\left(\frac{-\|m\|}{\tau}\right) \leq \exp\frac{\Omega}{\tau} - \exp\left(\frac{-\delta}{\tau}\right)
\end{aligned}$$

Since $\exp\left(\frac{-\|m\|}{\tau}\right)$ is always smaller than 1, we can choose Ω so that:

$$\begin{aligned}
&\exp\frac{\Omega}{\tau} - \exp\left(\frac{-\delta}{\tau}\right) = 1 \\
\Leftrightarrow &\Omega = \tau \log\left(1+\exp\left(\frac{-\delta}{\tau}\right)\right)
\end{aligned}$$

In our experiments, we set $\delta = 0.05$ and $\tau = 0.01$. We also observed that the hard normalization (Eq. 12) and the soft normalization (Eq. 14) of m give roughly the same performance.

E.6.2 Empirical Results From Table 22, we see that AIF with explicit trigger normalization (denoted as AIF-w) always achieve smaller ↓Cs and sometimes

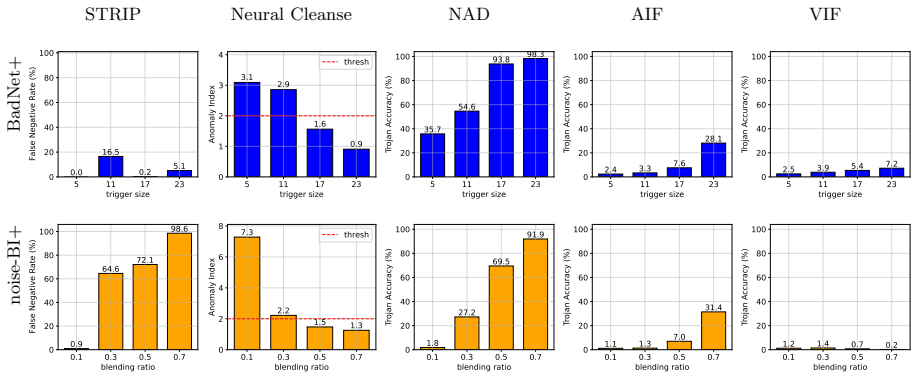
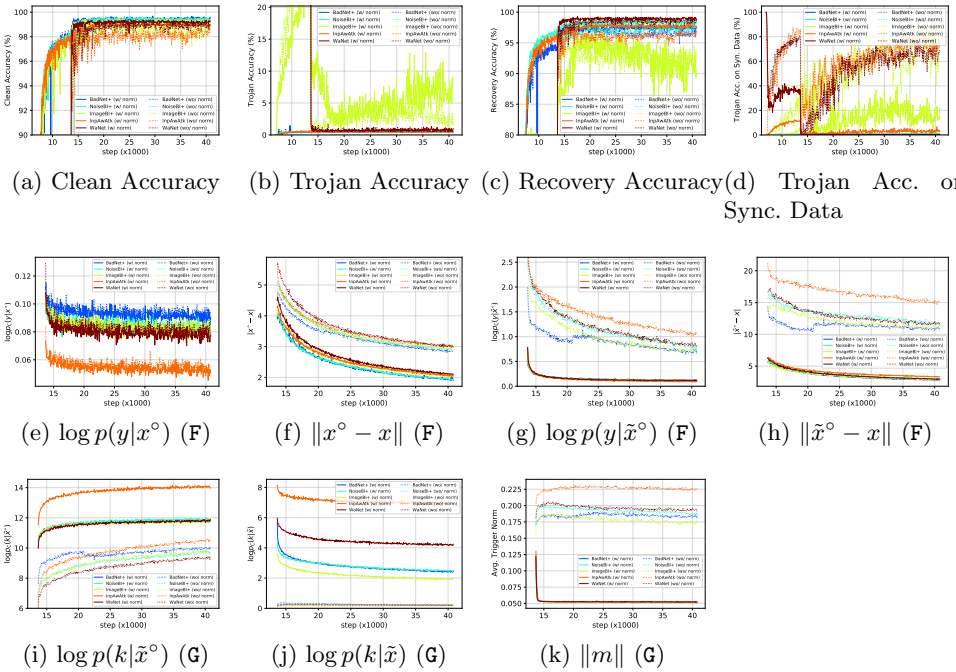


Fig. 19: FNRs at 10% FPR of STRIP, anomaly indices of Neural Cleanse, Trojan accuracies of NAD, and Trojan accuracies of our filtering defenses VIF, AIF against BadNet+ with different trigger sizes (top) and noise-BI+ with different blending ratios (bottom).

Dataset	Norm.	BadNet+			noise-BI+			image-BI+			InpAwAtk			WaNet		
		↓C	T	↓R	↓C	T	↓R	↓C	T	↓R	↓C	T	↓R	↓C	T	↓R
CIFAR10	w/	***5.60***	*2.37*	***9.03***	***4.87***	*1.14*	***6.02***	***5.23***	*1.96*	***7.10***	***5.28***	*5.30*	*11.87*	***4.30***	***1.22***	***5.67***
	wo/	5.90	**2.15**	10.00	5.60	**0.78**	7.50	6.60	**1.56**	7.53	6.20	**2.56**	**9.37**	5.80	2.22	8.10
GTSRB	w/	***-0.16***	***0.00***	***1.87***	***0.05***	***0.00***	***0.81***	***0.13***	*7.47*	*9.54*	***-0.03***	*0.05*	***1.37***	***-0.05***	*0.50*	***0.42***
	wo/	0.13	0.08	3.18	0.24	0.00	1.26	0.45	**0.92**	**5.44**	0.45	**0.00**	2.60	0.21	**0.11**	0.45

Table 22: Trojan filtering results (in %) of AIF with and without explicit trigger normalization against different attacks on CIFAR10 and GTSRB. For a particular attack, dataset and metric, the best result is highlighted in bold. Results taken from Table 2 are shown in italic.

achieve larger Ts than the counterpart *without* explicit trigger normalization (denoted as AIF-wo). In general, AIF-w usually achieves lower ↓Rs than AIF-wo and is considered to be better so we set it as default. Fig. 20 provides a deeper insight into the results in Table 22. Without explicit trigger normalization, the generator `G` can easily fool `F` (low crossentropy losses in Fig. 20i) by just increasing the norm of the synthetic triggers (Fig. 20k). This makes $\tilde{x}$ more different from x and causes more difficulty for `F` to force $\tilde{x}^\circ$ close to x (Fig. 20h) as well as correcting the label of $\tilde{x}$ (Fig. 20g). The large difference between $\tilde{x}$ and x also negatively affects the performance of `F` on reconstructing clean images (Fig. 20f). As a result, AIF-wo achieves much poorer Ts on synthetic Trojan images (Fig. 20d) than AIF-w. On ground-truth Trojan images, AIF-wo performs as well as AIF-w in terms of T (Fig. 20b) and worse than AIF-w in terms of ↓C (Fig. 20a) and ↓R (Fig. 20d).



(a) Clean Accuracy (b) Trojan Accuracy (c) Recovery Accuracy (d) Trojan Acc. on Sync. Data

(e) $\log p(y|x^\circ)$ (F) (f) $\|x^\circ - x\|$ (F) (g) $\log p(y|\tilde{x}^\circ)$ (F) (h) $\|\tilde{x}^\circ - x\|$ (F)

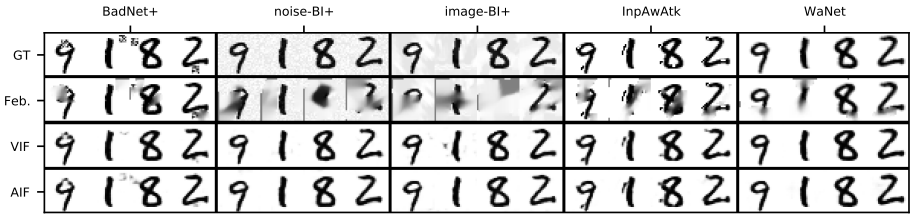
(i) $\log p(k|\tilde{x}^\circ)$ (G) (j) $\log p(k|\tilde{x})$ (G) (k) $\|m\|$ (G)

Fig. 20: Test result curves and training loss curves of F, G of VIF with and without explicit trigger normalization against different attacks on GTSRB. These plots correspond to the results in the bottom 2 rows in Table 22.

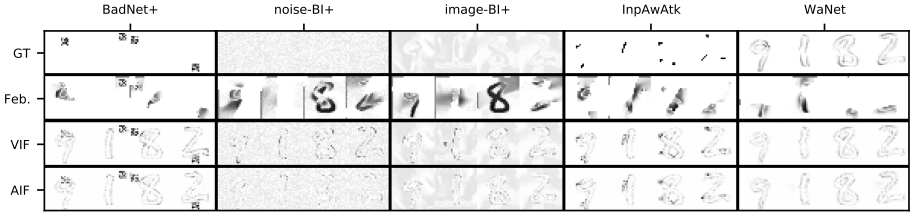
F Qualitative Results of Our Defenses

F.1 Against Benchmark Attacks

In Figs. 21a, 22a, 23a, 24a, we show some ground-truth (GT) Trojan images $\tilde{x}$ and their filtered counterparts $\tilde{x}^\circ$ computed by Februus (Feb.) and our filtering defenses (VIF, AIF) for different Trojan attacks and datasets. We also show the corresponding "counter-triggers" of $\tilde{x}^\circ$ defined as $|\tilde{x}^\circ - \tilde{x}|$ in Figs. 21b, 22b, 23b, 24b in comparison with the ground-truth Trojan triggers $|\tilde{x} - x|$. It is apparent that VIF and AIF correctly filter the true triggers of all the attacks without knowing them while Februus fails to filter the true triggers of noise-BI+, image-BI+, InpAwAtk and WaNet. The failure of Februus comes from the fact that GradCAM is unable to find regions containing full-sized, distributed triggers of noise/image-BI+ or isomorphic, input-specific triggers of InpAwAtk/WaNet. For BadNet+ and InpAwAtk, our filtering defenses mainly blur the triggers of these attacks instead of completely removing the triggers but this is enough to deactivate the triggers.



(a) Filtered images

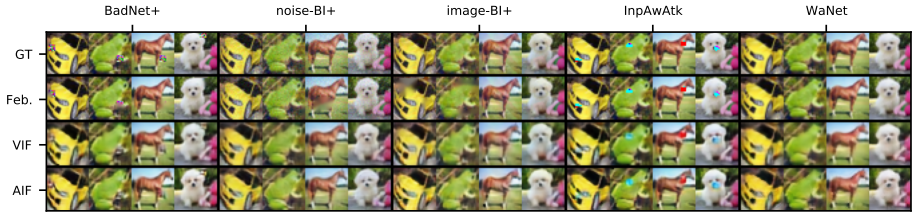


(b) Counter-triggers

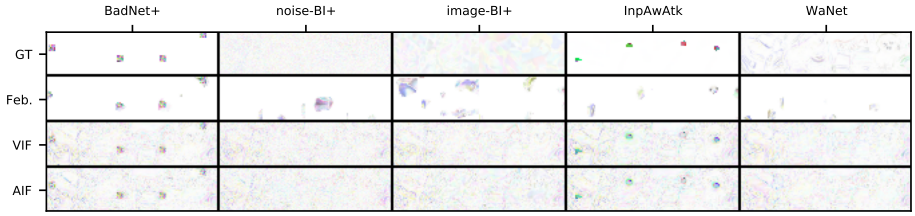
Fig. 21: (a): Ground-truth (GT) Trojan images of different attacks and the corresponding filtered images computed by Februus (Feb.), VIF, and AIF on MNIST. (b): GT triggers and counter-triggers w.r.t. the filtered images in (a).

F.2 Against Attacks with Large-Norm Triggers

In Fig. 25, we visualize the filtered images and their corresponding counter-triggers computed by our methods for Trojan images of BadNet+ with different trigger sizes and of noise-BI+ with different blending ratios. In general, our filtering defenses can effectively deactivate triggers embedded in the Trojan images via modifying the trigger pixels but cannot fully reconstruct the original clean images. These qualitative results correspond to the quantitative results in Tables 20, 21.

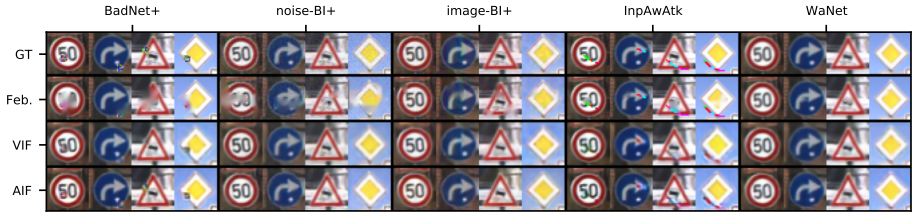


(a) Filtered images

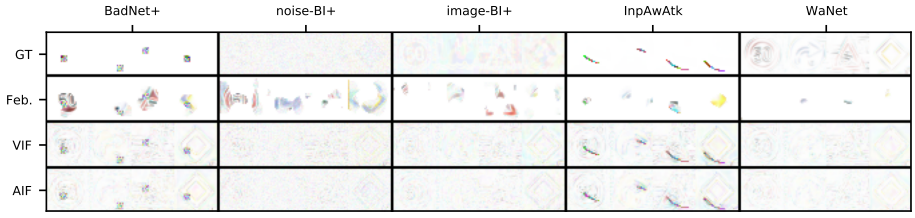


(b) Counter-triggers

Fig. 22: (a): Ground-truth (GT) Trojan images of different attacks and the corresponding filtered images computed by Februus (Feb.), VIF, and AIF on CIFAR10. (b): GT triggers and counter-triggers w.r.t. the filtered images in (a).

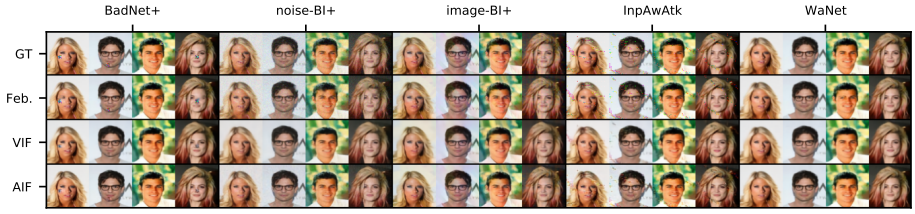


(a) Filtered images

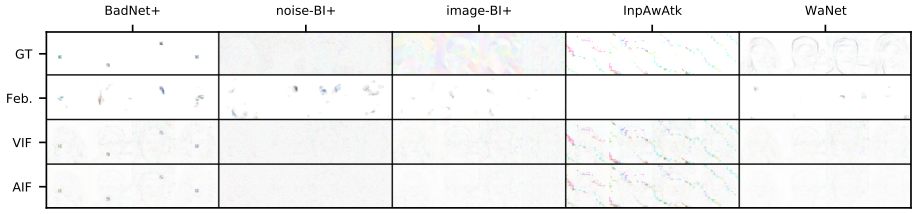


(b) Counter-triggers

Fig. 23: (a): Ground-truth (GT) Trojan images of different attacks and the corresponding filtered images computed by Februus (Feb.), VIF, and AIF on GTSRB. (b): GT triggers and counter-triggers w.r.t. the filtered images in (a).

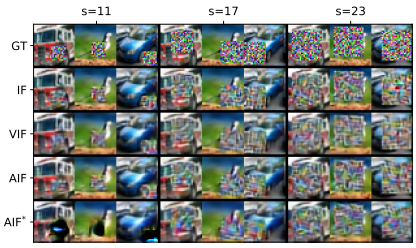


(a) Filtered images

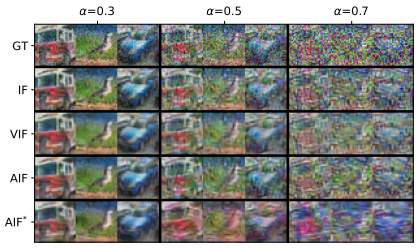


(b) Counter-triggers

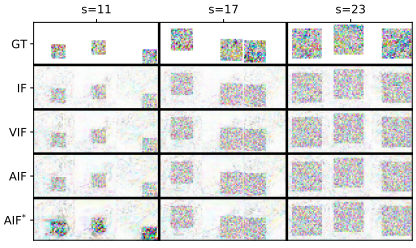
Fig. 24: (a): Ground-truth (GT) Trojan images of different attacks and the corresponding filtered images computed by Februus (Feb.), VIF, and AIF on CelebA. (b): GT triggers and counter-triggers w.r.t. the filtered images in (a).



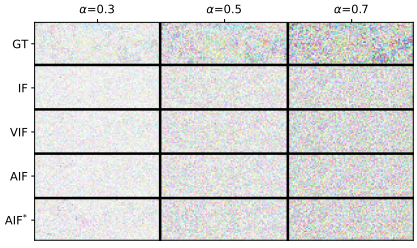
(a) Filtered images (BadNet+)



(b) Filtered images (noise-BI+)



(c) Counter-triggers (BadNet+)



(d) Counter-triggers (noise-BI+)

Fig. 25: (a)/(b): Ground-truth (GT) Trojan images of BadNet+/noise-BI+ and the corresponding filtered images computed by IF, VIF, AIF, and AIF without explicit trigger normalization (AIF*). (c)/(d): GT triggers and counter-triggers w.r.t. the filtered images in (a)/(b).