Towards Effective and Robust Neural Trojan Defenses via Input Filtering

Kien Do[1], Haripriya Harikumar[1], Hung Le[1], Dung Nguyen[1], Truyen Tran[1], Santu Rana[1], Dang Nguyen[1], Willy Susilo[2], and Svetha Venkatesh[1]

[1] Applied Artificial Intelligence Institute (A2I2), Deakin University, Australia
[2] University of Wollongong, Australia
{k.do, h.harikumar, thai.le, dung.nguyen, truyen.tran, santu.rana, d.nguyen, svetha.venkatesh}@deakin.edu.au
wsusilo@uow.edu.au

Abstract. Trojan attacks on deep neural networks are both dangerous and surreptitious. Over the past few years, Trojan attacks have advanced from using only a single input-agnostic trigger and targeting only one class to using multiple, input-specific triggers and targeting multiple classes. However, Trojan defenses have not caught up with this development. Most defense methods still make inadequate assumptions about Trojan triggers and target classes, thus, can be easily circumvented by modern Trojan attacks. To deal with this problem, we propose two novel *"filtering"* defenses called *Variational Input Filtering (VIF)* and *Adversarial Input Filtering (AIF)* which leverage lossy data compression and adversarial learning respectively to effectively purify potential Trojan triggers in the input at run time without making assumptions about the number of triggers/target classes or the input dependence property of triggers. In addition, we introduce a new defense mechanism called *"Filtering-then-Contrasting"* (FtC) which helps avoid the drop in classification accuracy on clean data caused by "filtering", and combine it with VIF/AIF to derive new defenses of this kind. Extensive experimental results and ablation studies show that our proposed defenses significantly outperform well-known baseline defenses in mitigating five advanced Trojan attacks including two recent state-of-the-art while being quite robust to small amounts of training data and large-norm triggers.

1 Introduction

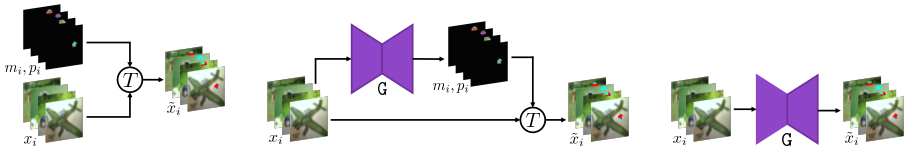
Deep neural networks (DNNs) have achieved superhuman performance in recent years and have been increasingly employed to make decisions on our behalf in various critical applications in computer vision including object detection [36], face recognition [34,39], medical imaging [29,51], surveillance [43] and so on. However, many recent works have shown that besides the powerful modeling capability, DNNs are highly vulnerable to adversarial attacks [7,10,11,25,44]. Currently, there are two major types of attacks on DNNs. The first is *evasion/adversarial attacks* which cause a *successfully trained* model to misclassify

by perturbing the model's input with imperceptible adversarial noise [10,28]. The second is *Trojan/backdoor attacks* in which attackers *interfere with the training process* of a model in order to insert hidden malicious features (referred to as *Trojans/backdoors*) into the model [4,11,25,41]. These Trojans do not cause any harm to the model under normal conditions. However, once they are triggered, they will force the model to output the target classes specified by the attackers. Unfortunately, only the attackers know exactly the Trojan triggers and the target classes. Such stealthiness makes Trojan attacks difficult to defend against.

In this work, we focus on defending against Trojan attacks. Most existing Trojan defenses assume that attacks use only *one input-agnostic* Trojan trigger and/or target only *one* class [3,5,8,12,13,47]. By constraining the space of possible triggers, these defenses are able to find the true trigger of some simple Trojan attacks satisfying their assumptions and mitigate the attacks [4,11]. However, these defenses often do not perform well against other advanced attacks that use *multiple input-specific* Trojan triggers and/or target *multiple* classes [6,32,33]. To address this problem, we propose two novel *filtering"* defenses named *Variational Input Filtering (VIF)* and *Adversarial Input Filtering (AIF)*. Both defenses aim at learning a filter network `F` that can purify potential Trojan triggers in the model's input at run time without making any of the above assumptions about attacks. VIF treats `F` as a variational autoencoder (VAE) [18] and utilizes the lossy data compression property of VAE to discard noisy information in the input including triggers. AIF, on the other hand, uses an auxiliary generator `G` to reveal hidden triggers in the input and leverages adversarial learning [9] between `G` and `F` to encourage `F` to remove potential triggers found by `G`. In addition, to overcome the issue that input filtering may hurt the model's prediction on clean data, we introduce a new defense mechanism called *"Filtering-then-Contrasting" (FtC)*. The key idea behind FtC is comparing the two outputs of the model with and without input filtering to determine whether the input is clean or not. If the two outputs are different, the input will be marked as containing triggers, otherwise clean. We equip VIF and AIF with FtC to arrive at the two defenses dubbed VIFtC and AIFtC respectively. Through extensive experiments and ablation studies, we demonstrate that our proposed defenses are more effective than many well-known defenses [5,8,22,47] in mitigating various advanced Trojan attacks including two recent state-of-the-art (SOTA) [32,33] while being quite robust to small amounts of training data and large trigger's norms.

2 Standard Trojan Attack

We consider image classification as the task of interest. We denote by $\mathbb{I}$ the real interval [0, 1]. In standard Trojan attack scenarios [4,11], an attacker (usually a service provider) *fully controls* the training process of an image classifier $\texttt{C} : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X} \subset \mathbb{I}^{c \times h \times w}$ is the input image domain, and $\mathcal{Y} = \{0, ..., K-1\}$ is the set of K classes. The attacker's goal is to insert a *Trojan* into the classifier `C` so that given an input image $x \in \mathcal{X}$, `C` will misclassify x as belonging to a target class $t \in \mathcal{Y}$ specified by the attacker if x contains the *Trojan trigger* ψ, and will



(a) (b) (c)

Fig. 1: Illustrations of three approaches to model input-specific Trojan triggers $\psi_i = (m_i, p_i)$ w.r.t. x_i: as learnable parameters (a), via a trigger generator (b), and via a Trojan-image autoencoder (c).

predict the true label $y \in \mathcal{Y}$ of x otherwise. A common attack strategy to achieve this goal is poisoning a small portion of the training data with the Trojan trigger ψ. At each training step, the attacker randomly replaces each clean training pair (x, y) in the current mini-batch by a poisoned one $(\tilde{x}, t)$ with a probability ρ $(0 < \rho < 1)$ and trains $\mathtt{C}$ as normal using the modified mini-batch. $\tilde{x}$ is an image embedded with Trojan triggers (or *Trojan image* for short) corresponding to x. $\tilde{x}$ is constructed by combining x with ψ via a Trojan injection function $T(x, \psi)$. A common choice of T is the image blending function [4,11] given below:

$$\tilde{x} = T(x, \psi) = (1 - m) \odot x + m \odot p, \tag{1}$$

where $\psi \triangleq (m, p)$, $m \in \mathbb{I}^{c \times h \times w}$ is the trigger mask, $p \in \mathbb{I}^{c \times h \times w}$ is the trigger pattern, and $\odot$ is the element-wise product. To ensure $\tilde{x}$ cannot be detected by human inspection at test time, $\|m\|$ must be small. Some recent works use more advanced variants of T such as reflection [24] and warping [33] to craft better natural-looking Trojan images.

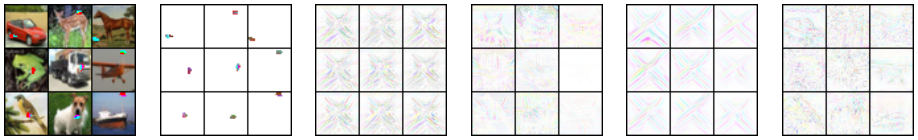
Once trained, the Trojan-infected classifier $\mathtt{C}$ will be provided to victims (usually end-users) for deployment. When the victims test $\mathtt{C}$ with their own clean data, they do not see any abnormalities in performance because the Trojan remains dormant for the clean data. Thus, the victims naively believe that $\mathtt{C}$ is normal and use $\mathtt{C}$ as it is without any modification or additional safeguard.

3 Difficulty in Finding Input-Specific Triggers

In practice, we (as victims) usually have a small dataset $\mathcal{D}_{\text{val}} = \{(x_i, y_i)\}_{i=1}^{N_{\text{val}}}$ containing only clean samples for evaluating the performance of $\mathtt{C}$. We can leverage this set to find possible Trojan triggers associated with the target class t. For standard Trojan attacks [4,11] that use only a global *input-agnostic* trigger $\psi = (m, p)$, ψ can be restored by minimizing the following loss w.r.t. m and p:

$$\mathcal{L}_{\text{gen}}(x, t) = -\log p_{\mathtt{C}}(t|\tilde{x}) + \lambda_0 \max(\|m\| - \delta, 0), \tag{2}$$

where $(x, .) \sim \mathcal{D}_{\text{val}}$, $\tilde{x}$ is derived from x via Eq. 1, $p_{\mathtt{C}}(t|\tilde{x}) = \frac{\exp(\mathtt{C}_t(\tilde{x}))}{\sum_{k=1}^{K} \exp(\mathtt{C}_k(\tilde{x}))}$ is the probability of $\tilde{x}$ belonging to the target class t, $\|\cdot\|$ denotes a L1/L2 norm,



(a) (b) (c) (d) (e) (f)

Fig. 2: Trojan images (a) and the corresponding triggers (b) of an Input-Aware Attack. Triggers synthesized by Neural Cleanse (c) and by the three approaches in Fig. 1 (d, e, f). Trigger pixels are inverted for better visualization.

$\delta \geq 0$ is an upper bound of the norm, and $\lambda_0 \geq 0$ is a coefficient. The second term in Eq. 2 ensures that the trigger is small enough so that it could not be detected by human inspection. $\mathcal{L}_{\text{gen}}$ was used by Neural Cleanse (NC) [47] and its variants [3,12,13], and was shown to work well for standard attacks.

In this work, we however consider finding the triggers of Input-Aware Attack (InpAwAtk) [32]. This is a much harder problem because InpAwAtk uses different triggers $\psi_i = (m_i, p_i)$ for different input images x_i instead of a global one. We examine 3 different ways to model ψ_i: (i) treating m_i, p_i as learnable parameters for each image $x_i \in \mathcal{D}_{\text{val}}$, (ii) via an input-conditional trigger generator $(m_i, p_i) = \texttt{G}(x_i)$, and (iii) generating a Trojan image $\tilde{x}_i$ w.r.t. x_i via a Trojan-image generator $\tilde{x}_i = \texttt{G}(x_i)$ and treating $\tilde{x}_i - x_i$ as ψ_i. These are illustrated in Fig. 1. The first way does not generalize to other images not in $\mathcal{D}_{\text{val}}$ while the second and third do. We reuse the loss $\mathcal{L}_{\text{gen}}$ in Eq. 2 to learn m_i, p_i in the first way and `G` in the second way. The loss to train `G` in the third way is slightly adjusted from $\mathcal{L}_{\text{gen}}$ with $\|m\|$ replaced by $\|\tilde{x} - x\|$. As shown in Fig. 2, neither NC nor the above approaches can restore the original triggers of InpAwAtk, suggesting new methods are on demand.

4 Proposed Trojan Defenses

The great difficulty in finding correct input-specific triggers (Section 3) challenges a majority of existing Trojan defenses which assume a global input-agnostic trigger is applied to all input images [3,8,13,22,23,35,47]. Fortunately, although we may not be able to find correct triggers, in many cases, we can still design effective Trojan defenses by filtering out triggers embedded in the input without concerning about the number or the input dependence property of triggers. The fundamental idea is learning a filter network `F` that maps the original input image x into a filtered image x°, and using x° as input to the classifier `C` instead of x. In order for `F` to be considered as a good filter, x° should satisfy the following two conditions:

– *Condition 1*: If x is clean, x° should look similar to x and should have the same label as x's. This ensures a *high classification accuracy on clean images* (dubbed *"clean accuracy"*).

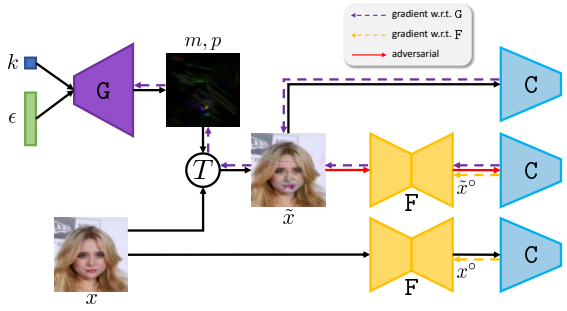


Fig. 3: An illustration of Adversarial Input Filtering.

– *Condition 2*: If $\tilde{x}$ contains triggers, $\tilde{x}^\circ$ should be close to x and should have the same label as x's where x is the clean counterpart of $\tilde{x}$. This ensures a *low attack success rate* and a *high clean-label recovery rate on Trojan images* (dubbed *"Trojan accuracy"* and *"recovery accuracy"*, respectively).

In the next two subsections (4.1, 4.2), we propose two novel filtering defenses that leverage two different strategies to learn a good $\mathtt{F}$ which are lossy data compression and adversarial learning, respectively.

4.1 Variational Input Filtering

A natural choice for $\mathtt{F}$ is an autoencoder (AE) which should be complex enough so that $\mathtt{F}$ can reconstruct clean images well to achieve high clean accuracy. However, if $\mathtt{F}$ is too complex, it can capture every detail of a Trojan image including the embedded triggers, which also causes high Trojan accuracy. In general, an optimal $\mathtt{F}$ should achieve good balance between preserving class-related information and discarding noisy information of the input. To reduce the dependence of $\mathtt{F}$ on architecture, we propose to treat $\mathtt{F}$ as a variational autoencoder (VAE)[3] [18] and train it with the *"Variational Input Filtering"* (VIF) loss given below:

$$\mathcal{L}_{\text{VIF}}(x, y) = -\log p_{\mathtt{C}}(y|x^\circ) + \lambda_1 \|x^\circ - x\| + \lambda_2 D_{\text{KL}}(q_{\mathtt{F}}(z|x)\|p(z)) \quad (3)$$

$$= \mathcal{L}_{\text{IF}} + \lambda_2 D_{\text{KL}}(q_{\mathtt{F}}(z|x)\|p(z)), \quad (4)$$

where $(x, y) \sim \mathcal{D}_{\text{val}}$, $x^\circ = \mathtt{F}(x)$ is the filtered version of x, z is the latent variable, $q_{\mathtt{F}}(z|x)$ denotes the variational posterior distribution and is parameterized via the stochastic encoder of $\mathtt{F}$, $p(z) = \mathcal{N}(0, \mathrm{I})$ is the standard Gaussian distribution, D_{KL} denotes the KL divergence, $\lambda_1, \lambda_2 \geq 0$ are coefficients. In Eq. 3, the first two terms force $\mathtt{F}$ to preserve class-related information of x and to reduce the visual dissimilarity between x° and x as per condition 1, 2. Meanwhile, the last term encourages $\mathtt{F}$ (or more precisely, $q_{\mathtt{F}}(z|x)$) to discard noisy information of

[3] Denoising Autoencoder (DAE) [46] is also a possible choice but is quite similar to VAE in terms of idea so we do not consider it here.

x, which we refer to as *"lossy data compression"*. This can be explained via the following relationship [54]:

$$\mathbb{E}_{p(x)}\left[D_{\mathrm{KL}}(q_{\mathtt{F}}(z|x)\|p(z))\right] = D_{\mathrm{KL}}(q_{\mathtt{F}}(z)\|p(z)) + I_{\mathtt{F}}(x,z), \tag{5}$$

where $q_{\mathtt{F}}(z) = \mathbb{E}_{p(x)}\left[q_{\mathtt{F}}(z|x)\right]$. Clearly, minimizing the LHS of Eq. 5 decreases the mutual information between z and x. And because z is used to compute x° (in decoding), this also reduces the information between x° and x.

In Eq. 3, the first two terms alone constitute the "Input Filtering" (IF) loss $\mathcal{L}_{\mathrm{IF}}$. To the best of our knowledge, IF has not been proposed in other Trojan defense works. Input Processing (IP) [25] is the closest to IF but it is trained on *unlabeled data* using *only the reconstruction loss* (the second term in Eq. 3). In Appdx. E.2, we show that IP performs worse than IF, which highlights the importance of the term $-\log p_{\mathtt{C}}(y|x^\circ)$.

4.2 Adversarial Input Filtering

VIF, owing to its generality, do not make use of any Trojan-related knowledge in $\mathtt{C}$ to train $\mathtt{F}$. We argue that if $\mathtt{F}$ is exposed to such knowledge, $\mathtt{F}$ could be more selective in choosing which input information to discard, and hence, could perform better. This motivates us to use synthetic Trojan images as additional training data for $\mathtt{F}$ besides clean images from $\mathcal{D}_{\mathrm{val}}$. We synthesize a Trojan image $\tilde{x}$ from a clean image x as follows:

$$(m_k, p_k) = \mathtt{G}(\epsilon, k), \tag{6}$$

$$\tilde{x} = T(x, (m_k, p_k)), \tag{7}$$

where $\epsilon \sim \mathcal{N}(0, \mathrm{I})$ is a standard Gaussian noise, k is a class label sampled uniformly from $\mathcal{Y}$, $\mathtt{G}$ is a conditional generator, T is the image blending function (Eq. 1). We choose the image blending function to craft Trojan images because it is the *most general* Trojan injection function (its output range spans the whole image space $\mathbb{I}^{c\times h\times w}$). To make sure that the synthetic Trojan images are useful for $\mathtt{F}$, we *form an adversarial game between* $\mathtt{G}$ *and* $\mathtt{F}$ in which $\mathtt{G}$ attempts to generate hard Trojan images that can fool $\mathtt{F}$ into producing the target class (sampled randomly from $\mathcal{Y}$) while $\mathtt{F}$ becomes more robust by correcting these images. We train $\mathtt{G}$ with the following loss:

$$\mathcal{L}_{\text{AIF-gen}}(x,k) = \mathcal{L}_{\mathrm{gen}}(x,k) - \lambda_3 \log p_{\mathtt{C}}(k|\tilde{x}^\circ), \tag{8}$$

where $\mathcal{L}_{\mathrm{gen}}$ is similar to the one in Eq. 2 but with m replaced by m_k (Eq. 6), $\tilde{x}^\circ = \mathtt{F}(\tilde{x})$, $\lambda_3 \geq 0$. The loss of $\mathtt{F}$ must conform to conditions 1, 2 and is:

$$\mathcal{L}_{\mathrm{AIF}}(x,y) = \mathcal{L}_{\mathrm{IF}}(x,y) - \lambda_4 \log p_{\mathtt{C}}(y|\tilde{x}^\circ) + \lambda_5\|\tilde{x}^\circ - x\| \tag{9}$$

$$= \mathcal{L}_{\mathrm{IF}}(x,y) + \mathcal{L}'_{\mathrm{IF}}(\tilde{x},y), \tag{10}$$

where AIF stands for *"Adversarial Input Filtering"*, $\mathcal{L}_{\mathrm{IF}}$ was described in Eq. 4, $\tilde{x}$ is computed from x via Eq. 7, $\lambda_4, \lambda_5 \geq 0$. Note that the last term in Eq. 10 is

the reconstruction loss between $\tilde{x}^\circ$ and x (not $\tilde{x}$). Thus, we denote the last two terms in Eq. 9 as $\mathcal{L}'_{\text{IF}}$ instead of $\mathcal{L}_{\text{IF}}$. AIF is depicted in Fig. 3.

During experiment, we observed that sometimes training `G` and `F` with the above losses may not result in good performance. The reason is that when `F` becomes better, `G` tends to produce large-norm triggers to fool `F` despite the fact that a regularization was applied to the norms of these triggers. Large-norm triggers make learning `F` harder as $\tilde{x}$ is no longer close to x. To handle this problem, we explicitly normalize m_k so that its norm is always bounded by δ. We provide technical details and empirical study about this normalization in Appdx. E.6. We also empirically examine the robustness of AIF and VIF in filtering large-norm triggers in Appdx. E.5.

4.3 Filtering then Contrasting

VIF and AIF always filter x even when x does not contain triggers, which often leads to the decrease in clean accuracy after filtering. To overcome this drawback, we introduce a new defense mechanism called *"Filtering then Contrasting"* (FtC) which works as follows: Instead of just computing the predicted label $\hat{y}^\circ$ of the filtered image $x^\circ = \texttt{F}(x)$ and treat it as the final prediction, we also compute the predicted label $\hat{y}$ of x without filtering and compare $\hat{y}$ with $\hat{y}^\circ$. If $\hat{y}$ is different from $\hat{y}^\circ$, x will be marked as containing triggers and discarded. Otherwise, x will be marked as clean and $\hat{y}$ will be used as the final prediction. FtC is especially useful for defending against attacks with large-norm triggers (Appdx. E.5) because it helps avoid the significant drop in clean accuracy caused by the large visual difference between x° and x. Under the FtC defense mechanism, we derive two new defenses VIFtC and AIFtC from VIF and AIF, respectively.

5 Experiments

5.1 Experimental Setup

Datasets Following previous works [11,32,38], we evaluate our proposed defenses on four image datasets namely MNIST, CIFAR10 [19], GTSRB [42], and CelebA [26]. For CelebA, we follow Salem et al. [38] and select the top 3 most balanced binary attributes (out of 40) to form an 8-class classification problem. The chosen attributes are *"Heavy Makeup"*, *"Mouth Slightly Open"*, and *"Smiling"*. Like other works [8,47], we assume that we have access to the test set of these datasets. We use 70% data of the test set for training our defense methods ($\mathcal{D}_{\text{val}}$ in Sections 3, 4) and 30% for testing (denoted as $\mathcal{D}_{\text{test}}$). For more details about the datasets, please refer to Appdx. B.1. Sometimes, we do not test our methods on all images in $\mathcal{D}_{\text{test}}$ but on those *not* belonging to the target class. This set is denoted as $\mathcal{D}'_{\text{test}}$. We also provide results with less training data in Appdx. E.4.

Benchmark Attacks We use 5 different benchmark Trojan attacks for our defenses, which are BadNet+, noise-BI+, image-BI+, InpAwAtk [32], and WaNet

Dataset	Benign	BadNet+		noise-BI+		image-BI+		InpAwAtk			WaNet		
	Clean	Clean	Trojan	Clean	Trojan	Clean	Trojan	Clean	Trojan	Cross	Clean	Trojan	Noise
MNIST	99.56	99.61	99.96	99.46	100.0	99.50	100.0	99.47	99.41	96.05	99.48	98.73	99.38
CIFAR10	94.82	94.88	100.0	94.69	100.0	95.15	99.96	94.58	99.43	88.68	94.32	99.59	92.58
GTSRB	99.72	99.34	100.0	99.30	100.0	99.18	100.0	98.90	99.54	95.19	99.12	99.54	99.03
CelebA	79.12	79.41	100.0	78.75	100.0	78.81	99.99	78.18	99.93	77.16	78.48	99.94	77.24

Table 1: Test clean and Trojan accuracies of various Trojan attacks.

[33]. InpAwAtk and WaNet are recent SOTA attacks that were shown to break many strong defenses completely. BadNet+ and noise/image-BI+ are variants of BadNet [11] and Blended Injection (BI) [4] that use multiple triggers instead of one. They are described in detail in Appdx. C.1. The training settings for the 5 attacks are given in Appdx. B.2.

We also consider 2 attack modes namely *single-target* and *all-target* [32,53]. In the first mode, only one class t is chosen as target. Every Trojan image $\tilde{x}$ is classified as t regardless of the ground-truth label of its clean counterpart x. Without loss of generality, t is set to 0. In the second mode, $\tilde{x}$ is classified as $(k+1) \mod K$ if x belongs to the class k. If not clearly stated, attacks are assumed to be *single-target*.

We report the test clean and Trojan accuracies of the benchmark attacks (in single-target mode) in Table 1. It is clear that all attacks achieve very high Trojan accuracies with little or no decrease in clean accuracy compared to the benign model's, hence, are qualified for our experimental purpose. For results of the attacks on $\mathcal{D}_{\text{test}}$, please refer to Appdx. C.

Baseline Defenses We consider 5 well-known baseline defenses namely Neural Cleanse (NC) [47], STRIP [8], Network Pruning (NP) [22], Neural Attention Distillation (NAD) [20], and Februus [5].

Neural Cleanse (NC) assumes that attacks (i) choose only one target class t and (ii) use *at least* (not exactly) one input-agnostic trigger associated with t. We refer to (i) as the *"single target class"* assumption and (ii) as the *"input-agnostic trigger"* assumption. Based on these assumptions, NC finds a trigger $\psi_k = (m_k, p_k)$ for every class $k \in \mathcal{Y}$ via reverse-engineering (Eq. 2), and uses the L1 norms of the synthetic trigger masks $\{m_1, ..., m_K\}$ to detect the target class. The intuition is that if t is the target class, $\|m_t\|_1$ will be much smaller than the rest. A z-value of each mask norm is calculated via Median Absolute Deviation and the z-value of the smallest mask norm (referred to as the *anomaly index*) is compared against a threshold ζ (2.0 by default). If the anomaly index is smaller than ζ, `C` is marked as clean. Otherwise, `C` is marked Trojan-infected with the target class corresponding to the smallest mask norm. In this case, the Trojans in `C` can be mitigated via pruning or via checking the cleanliness of input images. Both mitigation methods make use of ψ_t and are analyzed in Appdx. D.

STRIP assumes triggers are input-agnostic and argues that if an input image x contains triggers then these triggers still have effect if x is superimposed (blended) with other images. Therefore, STRIP superimposes x with N_{s} random

clean images from $\mathcal{D}_{\text{val}}$ and computes the *average entropy* $\mathcal{H}(x)$ of N_{s} predicted class probabilities corresponding to N_{s} superimposed versions of x. If $\mathcal{H}(x)$ is smaller than a predefined threshold, x is considered as trigger-embedded, otherwise, clean. The threshold is set according to the false positive rate (FPR) over the average entropies of all images in $\mathcal{D}_{\text{val}}$, usually at FPR equal to 1/5/10%. We evaluate the performance of STRIP against an attack using M_{s} random clean images from $\mathcal{D}_{\text{test}}$ and M_{s} corresponding Trojan images generated by that attack. Following [8], we set $N_{\text{s}} = 100$ and $M_{\text{s}} = 2000$.

Network Pruning (NP) hypothesizes that idle neurons are more likely to store Trojan-related information. Thus, it ranks neurons in the second top layer of `C` according to their average activation over all data in $\mathcal{D}_{\text{val}}$ and gradually prunes them until a certain decrease in clean accuracy is reached, usually at 1/5/10% decrease in clean accuracy.

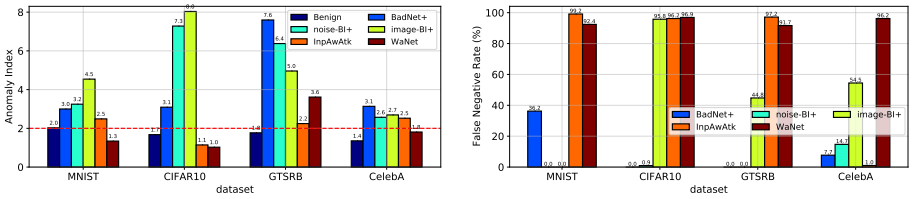
Neural Attention Distillation (NAD) [20] is a distillation-based Trojan defense. It first fine-tunes the pretrained classifier `C` on clean images in $\mathcal{D}_{\text{val}}$ to obtain a fine-tune classifier `T`. Then, it treats `T` and `C` as the teacher and student respectively, and performs attention-based feature distillation [52] between `T` and `C` on $\mathcal{D}_{\text{val}}$ again. Since `T` is `C` fine-tuned on clean data, `T` is expected to have most of the Trojan in `C` removed. Via distillation, such Trojan-free knowledge is transferred from `T` to `C` while performance of `C` on clean data is still preserved.

Among the baselines, Februus is the most related to our filtering defenses since it mitigates Trojan attacks via input purification. It uses GradCAM [40] to detect regions in an input image x that may contain triggers. Then, it removes all pixels in the suspected regions and generates new ones via inpainiting. The inpainted image is expected to contain no trigger and is fed to `C` instead of x.

Model Architectures and Training Settings Please refer to Appdx. B.3.

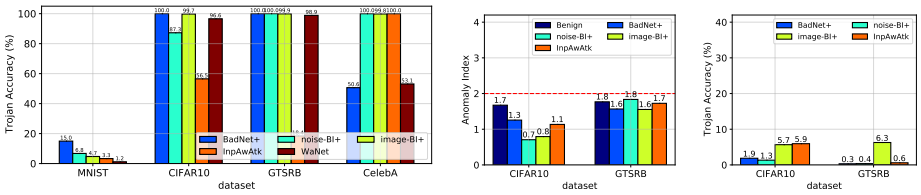
Metrics We evaluate VIF/AIF using 3 metrics namely *decrease in clean accuracy* (↓C), *Trojan accuracy* (T), and *decrease in recovery accuracy* (↓R). The first is the difference between the classification accuracies of clean images before and after filtering. The second is the attack success rate of Trojan images after filtering. The last is the difference between the classification accuracy of clean images before filtering and that of the corresponding Trojan images after filtering. Smaller values of the metrics indicate better results. ↓C and ↓R are computed on $\mathcal{D}_{\text{test}}$. T is computed on $\mathcal{D}'_{\text{test}}$ under single-target attacks and $\mathcal{D}_{\text{test}}$ under all-target attacks. This ensures that T can be 0 in the best case. Otherwise, T will be around $1/K$ where K is the total number of classes. ↓C and ↓R are upper-bounded by 1 and can be negative.

We evaluate VIFtC/AIFtC using FPR and FNR. FPR/FNR is defined as the proportion of clean/Trojan images having different/similar class predictions when the filter `F` is applied and not applied. FPR is computed on $\mathcal{D}_{\text{test}}$. FNR is computed on $\mathcal{D}'_{\text{test}}$ under single-target attacks and $\mathcal{D}_{\text{test}}$ under all-target attacks. Both metrics are in [0, 1] and smaller values of them are better. Interestingly, FPR and FNR are strongly correlated to ↓C and T, respectively. FPR/FNR is exactly equal to ↓C/T if `C` achieves perfect clean/Trojan accuracy.



(a) Neural Cleanse (b) STRIP

Fig. 4: (a) Anomaly indices of Neural Cleanse. The red dashed line indicates the threshold. (b) FNRs of STRIP at 10% FPR.



(a) Network Pruning (b) Neural Cleanse (c) VIF

Fig. 5: (a) Trojan accuracies at 10% decrease in clean accuracy of different Trojan classifiers pruned by Network Pruning. (b) Anomaly indices of NC and (c) Trojan accuracies of VIF against *all-target* attacks on CIFAR10 and GTSRB.

5.2 Results of Baseline Defenses

In Fig. 4, we show the detection results of Neural Cleanse (NC) and STRIP w.r.t. the aforementioned attacks. The two defenses are effective against BadNet+ and image/noise-BI+. This is because STRIP and NC generally do not make any assumption about the number of triggers. However, STRIP performs poorly against InpAwAtk and WaNet (FNRs $> 90\%$) since these advanced attacks break its "input-agnostic trigger" assumption. NC also fails to detect the Trojan classifiers trained by WaNet on most datasets for the same reason. What surprises us is that in our experiment NC correctly detect the Trojan classifiers trained by InpAwAtk on 3/4 datasets while in the original paper [32], it was shown to fail completely. We are confident that this inconsistency does not come from our implementation of InpAwAtk since we used the same hyperparameters and achieved the same classification results as those in the original paper (Table 1 versus Fig. 3b in [32]). However, NC is still unable to mitigate all Trojans in these correctly-detected Trojan classifiers (Appdx. D.3). In addition, as shown in Fig. 5b, NC is totally vulnerable to *all-target attacks* since its "single target class" assumption is no longer valid under these attacks. Network Pruning (NP), despite being assumption-free, cannot mitigate Trojans from most attacks (high Trojan accuracies in Fig. 5a) as it fails to prune the correct neurons containing Trojans. Februus has certain effects on mitigating Trojans from BadNet+

Dataset	Defense	Benign	BadNet+			noise-BI+			image-BI+			InpAwAtk			WaNet		
		↓C	↓C	T	↓R	↓C	T	↓R	↓C	T	↓R	↓C	T	↓R	↓C	T	↓R
MNIST	Feb.	5.96	39.08	96.24	86.32	2.30	100.0	89.58	8.19	100.0	89.58	9.90	92.40	83.32	25.43	80.46	88.75
	NAD	0.45	0.82	35.72	36.41	0.75	84.83	76.22	0.78	88.34	79.18	0.80	4.46	5.29	0.42	0.44	0.98
	IF	**0.10**	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
	VIF	0.13	**0.17**	**2.36**	**3.63**	0.12	**0.04**	0.63	**0.03**	**0.11**	0.40	0.20	1.25	1.83	**0.10**	0.48	0.53
	AIF	**0.10**	**0.17**	3.80	4.86	0.13	0.15	**0.11**	0.10	**0.11**	**0.10**	**0.03**	1.14	**1.66**	0.13	**0.15**	**0.20**
CIFAR10	Feb.	32.67	49.17	12.63	19.57	26.73	43.59	78.90	39.70	92.67	81.00	53.43	49.52	66.50	55.80	98.70	83.30
	NAD	**3.16**	**3.81**	35.71	41.68	**2.52**	1.81	28.89	**3.87**	**1.63**	18.92	**2.98**	**1.81**	**4.75**	**2.95**	**0.93**	**5.42**
	IF	3.34	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
	VIF	7.81	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
	AIF	4.67	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
GTSRB	Feb.	42.01	35.30	21.02	44.11	43.40	75.75	95.90	32.18	97.83	97.37	21.27	70.02	72.71	33.18	70.10	71.69
	NAD	-0.13	**-0.35**	**0.00**	8.20	**-0.32**	**0.00**	4.06	**-0.42**	**0.05**	**8.33**	**-0.28**	0.05	**0.56**	**-0.40**	**0.00**	**0.11**
	IF	0.12	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
	VIF	0.18	0.45	**0.00**	3.55	0.18	**0.00**	1.12	0.37	12.12	16.56	0.11	**0.03**	1.87	0.55	3.67	3.89
	AIF	**0.05**	-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
CelebA	Feb.	12.71	18.80	42.96	21.33	11.76	93.27	49.05	13.30	98.59	49.84	5.60	99.98	49.71	9.16	97.30	48.53
	NAD	3.06	**3.19**	12.14	9.98	3.56	25.31	23.07	3.51	16.97	9.46	3.14	13.85	11.24	2.51	11.48	**3.21**
	IF	**2.23**	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
	VIF	3.74	4.63	9.28	4.90	3.20	**11.51**	**4.08**	3.54	**14.32**	**5.62**	3.89	11.55	6.27	3.96	**8.30**	4.19
	AIF	4.95	6.46	**7.85**	6.49	4.18	12.56	6.52	4.37	18.40	9.23	3.71	**10.43**	7.65	4.02	12.82	5.74

Table 2: Trojan filtering results (in %) of Februus, NAD, and our filtering defenses against different attacks. *Smaller values are better*. For a particular dataset, attack, and metric, the best defense is highlighted in bold.

while being useless against the remaining attacks (high Ts in Table 2). This is because GradCAM, the method used by Februus, is only suitable for detecting patch-like triggers of BadNet+, not full-size noise-like triggers of image/noise-BI+ or polymorphic triggers of InpAwAtk/WaNet. We also observe that Februus significantly reduces the clean accuracy (high ↓Cs in Table 2) as it removes input regions that contain no Trojan trigger yet are highly associated with the output class. This problem, however, was not discussed in the Februus paper. NAD, thanks to its distillation-based nature, usually achieves better clean accuracies than our filtering defenses (Table 2). This defense is also effective against InpAwAtk and WaNet. However, NAD performs poorly in recovering Trojan samples from BadNet+ and noise/image-BI+ (high ↓Rs), especially on MNIST. Besides, NAD is much less robust to large-norm triggers than our filtering defenses (Appdx. E.5). For more analyses of the baseline defenses, please refer to Appdx. D.

5.3 Results of Proposed Defenses

From Table 2, it is clear that VIF and AIF achieve superior performances in mitigating Trojans of all the single-target attacks compared to most of the baseline defenses. For example, on MNIST and GTSRB, our filtering defenses impressively reduce T from about 100% (Table 1) to less than 2% for most attacks yet only cause less than 1% drop of clean accuracy (↓C $< 1\%$). On more diverse datasets such as CIFAR10 and CelebA, VIF and AIF still achieve T less than 6% and 12% for most attacks while maintaining ↓C below 8% and 5%, respectively. We note that on CelebA, the nonoptimal performance of `C` (accuracy $\approx 79\%$)

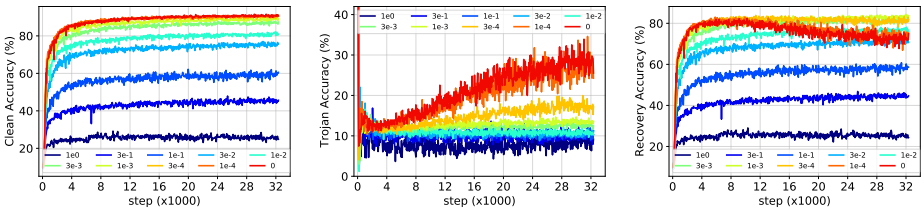
Dataset	Defense	Benign	BadNet+		noise-BI+		image-BI+		InpAwAtk		WaNet	
		FPR	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR
MNIST	IFtC	0.40	0.50	**2.21**	0.37	0.22	0.37	1.51	0.53	**1.71**	0.60	1.33
	VIFtC	0.27	**0.30**	2.84	0.23	**0.07**	0.40	0.15	0.47	1.99	0.30	1.51
	AIFtC	**0.23**	0.32	4.17	**0.17**	0.15	**0.17**	**0.11**	**0.33**	1.87	**0.13**	**1.18**
CIFAR10	IFtC	**6.83**	**7.47**	**1.70**	**6.57**	0.93	**7.83**	36.56	**7.25**	16.89	**7.17**	5.15
	VIFtC	12.30	11.00	2.63	10.67	1.26	11.03	10.89	10.63	**3.67**	11.40	4.26
	AIFtC	8.63	8.87	1.96	8.73	**0.89**	8.77	**2.15**	8.27	5.93	7.93	**1.56**
GTSRB	IFtC	**0.38**	0.45	**0.00**	0.24	0.03	0.66	52.91	**0.29**	1.00	0.66	10.41
	VIFtC	0.45	0.74	0.03	0.47	**0.00**	0.87	12.63	0.53	**0.37**	1.31	4.25
	AIFtC	0.50	**0.37**	0.03	0.39	**0.00**	**0.63**	**7.87**	0.47	0.40	**0.60**	**1.08**
CelebA	IFtC	**14.24**	**15.78**	8.36	**14.94**	14.25	**13.99**	59.43	**12.84**	11.95	**13.08**	15.27
	VIFtC	17.74	18.90	9.09	18.50	**11.67**	18.09	**14.30**	16.37	11.54	17.22	**8.34**
	AIFtC	20.24	20.95	**7.71**	19.08	12.82	19.29	18.65	16.54	**10.43**	16.55	12.87

Table 3: Trojan mitigation results (in %) of our FtC defenses against different attacks. *Smaller values are better*. For a particular attack, dataset, and metric, the best defense is highlighted in bold.

makes T higher than normal because T may contains the error of samples from non-target classes misclassified as the target class. However, it is not trivial to disentangle the two quantities so we leave this problem for future work. As there is no free lunch, our filtering defenses may be not as good as some baselines in some specific cases. For example, on CIFAR10, STRIP achieves FNRs $\approx 0\%$ against BadNet+/noise-BI+ (Fig. 4b) while VIF/AIF achieves Ts $\approx$ 1-3%. However, the gaps are very small and in general, our filtering defenses are still much more effective than the baseline defenses against all the single-target attacks. Our filtering defenses also perform well against all-target attacks (Fig. 5c and Appdx. E.1) as ours are not sensitive to the number of target classes. To gain a better insight into the performance of VIF/AIF, we visualize the filtered images produced by VIF/AIF and their corresponding *"counter-triggers"* in Appdx. F.1.

Among the filtering defenses, IF usually achieves the smallest ↓Cs because its loss does not have any term that encourages information removal like VIF's and AIF's. The gaps in ↓C between IF and AIF/VIF are the largest on CIFAR10 but do not exceed 5%. However, IF usually performs much worse than VIF/AIF in mitigating Trojans, especially those from image-BI+, InpAwAtk, and WaNet. For example, on CIFAR10, GTSRB, and CelebA, IF reduces the attack success rate (T) of image-BI+ to 37.48%, 52.27%, and 59.39% respectively. These numbers are only 1.96%, 7.47%, and 18.40% for AIF and 10.52%, 12.12%, and 14.32% for VIF. Therefore, when considering the trade-off between ↓C and T, VIF and AIF are clearly better than IF. We also observe that AIF usually achieves lower ↓Cs and ↓Rs than VIF. It is because AIF discards only potential malicious information instead of all noisy information like VIF. However, VIF is simpler and easier to train than AIF.

From Table 3, we see that the FPRs and FNRs of VIFtC/AIFtC are close to the ↓Cs and Ts of VIF/AIF respectively on MNIST, GTSRB, and CIFAR10. This is because `C` achieves nearly 100% clean and Trojan accuracies on these



(a) Clean Accuracy (b) Trojan Accuracy (c) Recovery Accuracy

Fig. 6: Clean, Trojan, and recovery accuracy curves of VIF against InpAwAtk on CIFAR10 w.r.t. different values of λ_2 in Eq. 3. The Trojan accuracy curves in (b) fluctuate around 10% since they are computed on $\mathcal{D}_{\text{test}}$ instead of $\mathcal{D}'_{\text{test}}$.

datasets. Thus, we can interpret the results of VIFtC/AIFtC in the same way as what we have done for VIF/AIF. Since FPR only affects the classification throughput not (clean) accuracy, VIFtC/AIFtC are preferred to VIF/AIF in applications that favor (clean) accuracy (e.g., defending against attacks with large-norm triggers in Appdx. E.5).

5.4 Ablation Studies

It is undoubted that our defenses require some settings to work well. However, these settings *cannot be managed by attackers* unlike the assumptions of most existing defenses [8,47]. Due to space limit, below we only examine the contribution of lossy data compression to the performance of VIF. For studies on the robustness of our proposed defenses to different architectures of `F`, to small amounts of training data, and to large-norm triggers, please refer to Appdx. E.3, E.4, and E.5 respectively.

5.4.1 Different data compression rates in VIF The lossy data compression in VIF can be managed via changing the coefficients of D_{KL} in $\mathcal{L}_{\text{VIF}}$ (λ_2 in Eq. 3). A smaller values of λ_2 means a lower data compression rate and vice versa. From Fig. 6, it is clear that when λ_2 is small, most information in the input including both semantic information and embedded triggers is retained, thus, the clean accuracy (C) and the Trojan accuracy (T) are both high. To decide the optimal value of λ_2, we base on recovery accuracy (R) since R can be seen as a combination of C and T to some extent. From the results on CIFAR10 (Fig. 6) and on other datasets, we found $\lambda_2 = 0.003$ to be the best.

6 Related Work

Due to space limit, in this section we only discuss related work about Trojan defenses. Related work about Trojan attacks are provided in Appdx. A. A

large number of Trojan defenses have been proposed so far, among which Neural Cleanse (NC) [47], Network Pruning (NP) [22], STRIP [8], Neural Attention Distillation (NAD) [20], and Februus [5] are representative for five different types of defenses and are carefully analyzed in Section 5.1. DeepInspect [3], MESA [35] improve upon NC by synthesizing a distribution of triggers for each class instead of just a single one. TABOR [12] adds more regularization losses to NC to better handle large and scattered triggers. STS [13] restores triggers by minimizing a novel loss function which is the pairwise difference between the class probabilities of two random synthetic Trojan images. This makes STS independent of the number of classes and more efficient than NC on datasets with many classes. ABS [23] is a quite complicated defense inspired by brain stimulation. It analyzes all neurons in the classifier `C` to find "compromised" ones and use these neurons to validate whether `C` is attacked or not. DL-TND [48], B3D [6] focus on detecting Trojan-infected models in case validation data are limited. However, all the aforementioned defenses derived from NC still make the same "input-agnostic trigger" and "single target class" assumptions as NC, and hence, are supposed to be ineffective against attacks that break these assumptions such as input-specific [32,33] and all-target attacks. Activation Clustering [2] and Spectral Signatures [45] regard hidden activations as a clue to detect Trojan samples from BadNet [11]. They base on an empirical observation that the hidden activations of Trojan samples and clean samples of the target class usually form distinct clusters in the hidden activation space. These defenses are of the same kind as STRIP and are not applicable to all-target attacks. Mode Connectivity Repair (MCR) [53] mitigates Trojans by choosing an interpolated model near the two end points of a parametric path connecting a Trojan model and its fine-tuned version. MCR was shown to be defeated by InpAwAtk in [32]. Adversarial Neuron Pruning (ANP) [50] leverages adversarial learning to find compromised neurons for pruning. Our AIF is different from ANP in the sense that we use adversarial learning to train an entire generator `G` for filtering input instead of pruning neurons.

7 Conclusion

We have proposed two novel "filtering" Trojan defenses dubbed VIF and AIF that leverage lossy data compression and adversarial learning respectively to effectively remove all potential Trojan triggers embedded in the input. We have also introduced a new defense mechanism called "Filtering-then-Contrasting" (FtC) that circumvents the loss in clean accuracy caused by "filtering". Unlike most existing defenses, our proposed filtering and FtC defenses make no assumption about the number of triggers/target classes or the input dependency property of triggers. Through extensive experiments, we have demonstrated that our proposed defenses significantly outperform many well-known defenses in mitigating various strong attacks. In the future, we would like to extend our proposed defenses to other domains (e.g., texts, graphs) and other tasks (e.g., object detection, visual reasoning) which we believe are more challenging than those considered in this work.

References

1. van Baalen, M., Louizos, C., Nagel, M., Amjad, R.A., Wang, Y., Blankevoort, T., Welling, M.: Bayesian bits: Unifying quantization and pruning. arXiv preprint arXiv:2005.07093 (2020)
2. 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. arXiv preprint arXiv:1811.03728 (2018)
3. Chen, H., Fu, C., Zhao, J., Koushanfar, F.: DeepInspect: A Black-box Trojan Detection and Mitigation Framework for Deep Neural Networks. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence. AAAI Press. pp. 4658–4664 (2019)
4. 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)
5. Doan, B.G., Abbasnejad, E., Ranasinghe, D.C.: Februus: Input purification defense against trojan attacks on deep neural network systems. In: Annual Computer Security Applications Conference. pp. 897–912 (2020)
6. Dong, Y., Yang, X., Deng, Z., Pang, T., Xiao, Z., Su, H., Zhu, J.: Black-box detection of backdoor attacks with limited information and data. arXiv preprint arXiv:2103.13127 (2021)
7. Fawzi, A., Fawzi, H., Fawzi, O.: Adversarial vulnerability for any classifier. arXiv preprint arXiv:1802.08686 (2018)
8. Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D.C., Nepal, S.: Strip: A Defence Against Trojan Attacks on Deep Neural Networks. In: Proceedings of the 35th Annual Computer Security Applications Conference. pp. 113–125 (2019)
9. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in neural information processing systems **27** (2014)
10. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
11. Gu, T., Dolan-Gavitt, B., Garg, S.: Badnets: Identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733 (2017)
12. Guo, W., Wang, L., Xing, X., Du, M., Song, D.: Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. arXiv preprint arXiv:1908.01763 (2019)
13. Harikumar, H., Le, V., Rana, S., Bhattacharya, S., Gupta, S., Venkatesh, S.: Scalable backdoor detection in neural networks. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD). pp. 289–304. Springer (2020)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
15. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European conference on computer vision. pp. 630–645. Springer (2016)
16. Ji, Y., Zhang, X., Wang, T.: Backdoor attacks against learning systems. In: IEEE Conference on Communications and Network Security. pp. 1–9. IEEE (2017)
17. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980 (2014)
18. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)

19. Krizhevsky, A.: Learning Multiple Layers of Features from Tiny Images. Tech. rep. (2009)
20. Li, Y., Lyu, X., Koren, N., Lyu, L., Li, B., Ma, X.: Neural attention distillation: Erasing backdoor triggers from deep neural networks. arXiv preprint arXiv:2101.05930 (2021)
21. Li, Y., Li, Y., Wu, B., Li, L., He, R., Lyu, S.: Backdoor attack with sample-specific triggers. arXiv preprint arXiv:2012.03816 (2020)
22. Liu, K., Dolan-Gavitt, B., Garg, S.: Fine-pruning: Defending against backdooring attacks on deep neural networks. In: International Symposium on Research in Attacks, Intrusions, and Defenses. pp. 273–294. Springer (2018)
23. Liu, Y., Lee, W.C., Tao, G., Ma, S., Aafer, Y., Zhang, X.: ABS: Scanning Neural Networks for Back-doors by Artificial Brain Stimulation. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security. pp. 1265–1282 (2019)
24. Liu, Y., Ma, X., Bailey, J., Lu, F.: Reflection backdoor: A natural backdoor attack on deep neural networks. In: European Conference on Computer Vision. pp. 182–199. Springer (2020)
25. Liu, Y., Xie, Y., Srivastava, A.: Neural trojans. In: 2017 IEEE International Conference on Computer Design (ICCD). pp. 45–48. IEEE (2017)
26. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: Proceedings of the IEEE international conference on computer vision. pp. 3730–3738 (2015)
27. Louizos, C., Ullrich, K., Welling, M.: Bayesian compression for deep learning. arXiv preprint arXiv:1705.08665 (2017)
28. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)
29. Moeskops, P., Veta, M., Lafarge, M.W., Eppenhof, K.A., Pluim, J.P.: Adversarial training and dilated convolutions for brain mri segmentation. In: Deep learning in medical image analysis and multimodal learning for clinical decision support, pp. 56–64. Springer (2017)
30. Molchanov, D., Ashukha, A., Vetrov, D.: Variational dropout sparsifies deep neural networks. In: International Conference on Machine Learning. pp. 2498–2507. PMLR (2017)
31. Muñoz-González, L., Pfitzner, B., Russo, M., Carnerero-Cano, J., Lupu, E.C.: Poisoning attacks with generative adversarial nets. arXiv preprint arXiv:1906.07773 (2019)
32. Nguyen, A., Tran, A.: Input-aware dynamic backdoor attack. arXiv preprint arXiv:2010.08138 (2020)
33. Nguyen, A., Tran, A.: Wanet–imperceptible warping-based backdoor attack. International Conference on Learning Representations (2021)
34. Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition (2015)
35. Qiao, X., Yang, Y., Li, H.: Defending neural backdoors via generative distribution modeling. arXiv preprint arXiv:1910.04749 (2019)
36. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016)
37. Saha, A., Subramanya, A., Pirsiavash, H.: Hidden trigger backdoor attacks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 11957–11965 (2020)
38. Salem, A., Wen, R., Backes, M., Ma, S., Zhang, Y.: Dynamic backdoor attacks against machine learning models. arXiv preprint arXiv:2003.03675 (2020)

39. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 815–823 (2015)
40. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision. pp. 618–626 (2017)
41. Shafahi, A., Huang, W.R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., Goldstein, T.: Poison frogs! targeted clean-label poisoning attacks on neural networks. arXiv preprint arXiv:1804.00792 (2018)
42. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: Man vs. Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition. Neural Networks pp. 323–332 (2012)
43. Sultani, W., Chen, C., Shah, M.: Real-world anomaly detection in surveillance videos. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6479–6488 (2018)
44. Thys, S., Van Ranst, W., Goedemé, T.: Fooling automated surveillance cameras: adversarial patches to attack person detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2019)
45. Tran, B., Li, J., Madry, A.: Spectral Signatures in Backdoor Attacks. In: Advances in Neural Information Processing Systems. pp. 8000–8010 (2018)
46. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on Machine learning. pp. 1096–1103 (2008)
47. Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., Zhao, B.Y.: Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In: IEEE Symposium on Security and Privacy. pp. 707–723. IEEE (2019)
48. Wang, R., Zhang, G., Liu, S., Chen, P.Y., Xiong, J., Wang, M.: Practical detection of trojan neural networks: Data-limited and data-free cases. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16. pp. 222–238. Springer (2020)
49. Wenger, E., Passananti, J., Bhagoji, A.N., Yao, Y., Zheng, H., Zhao, B.Y.: Backdoor attacks against deep learning systems in the physical world. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6206–6215 (2021)
50. Wu, D., Wang, Y.: Adversarial neuron pruning purifies backdoored deep models. Advances in Neural Information Processing Systems **34** (2021)
51. Yang, D., Xu, D., Zhou, S.K., Georgescu, B., Chen, M., Grbic, S., Metaxas, D., Comaniciu, D.: Automatic liver segmentation using an adversarial image-to-image network. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 507–515. Springer (2017)
52. Zagoruyko, S., Komodakis, N.: Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. arXiv preprint arXiv:1612.03928 (2016)
53. Zhao, P., Chen, P.Y., Das, P., Ramamurthy, K.N., Lin, X.: Bridging mode connectivity in loss landscapes and adversarial robustness. arXiv preprint arXiv:2005.00060 (2020)
54. Zhao, S., Song, J., Ermon, S.: Infovae: Information maximizing variational autoencoders. arXiv preprint arXiv:1706.02262 (2017)

55. Zhu, C., Huang, W.R., Li, H., Taylor, G., Studer, C., Goldstein, T.: Transferable clean-label poisoning attacks on deep neural nets. In: International Conference on Machine Learning. pp. 7614–7623. PMLR (2019)