NNoculation: Catching BadNets in the Wild

Akshaj Kumar Veldanda
New York University
New York, USA
akv275@nyu.edu

Kang Liu
Huazhong University of Science and Technology
Wuhan, China
kangliu@hust.edu.cn

Benjamin Tan
University of Calgary
Calgary, Canada
benjamin.tan1@ucalgary.ca

Prashanth Krishnamurthy
New York University
New York, USA
prashanth.krishnamurthy@nyu.edu

Farshad Khorrami
New York University
New York, USA
khorrami@nyu.edu

Ramesh Karri
New York University
New York, USA
rkarri@nyu.edu

Brendan Dolan-Gavitt
New York University
New York, USA
brendandg@nyu.edu

Siddharth Garg
New York University
New York, USA
sg175@nyu.edu

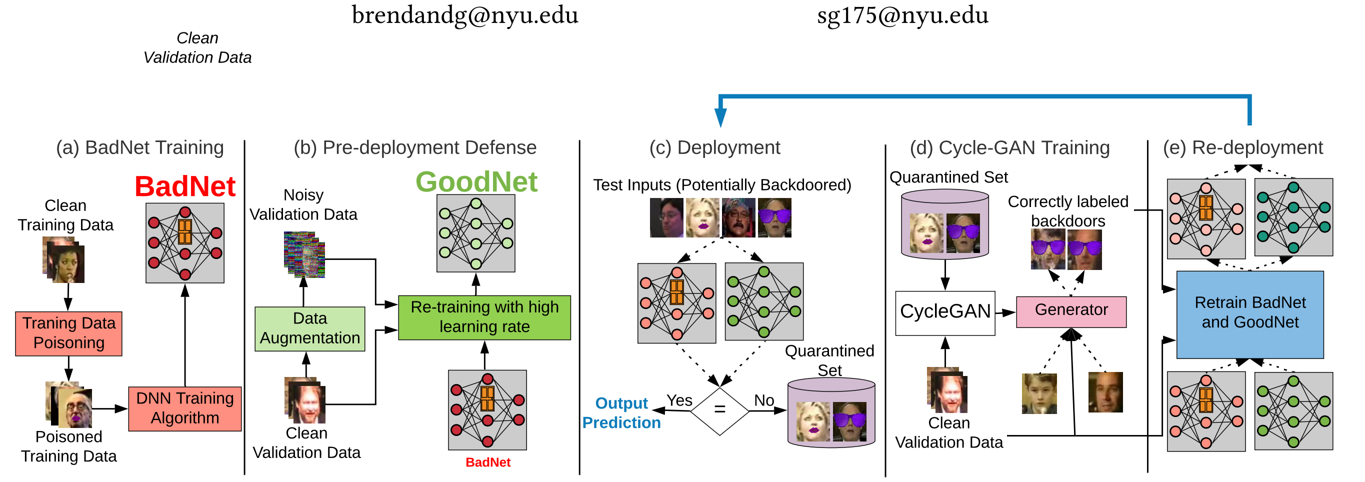


Figure 1: An overview of NNoculation: (a) BadNet Training: First, the user/defender acquires a potential BadNet. (b) Pre-deployment Defense: The BadNet is retrained with noise-augmented validation data and high learning rate to obtain GoodNet. (c) Deployment: The BadNet and GoodNet are deployed as an ensemble that will either quarantine a test input if the outputs disagree or give a prediction if outputs agree. The quarantined set will mostly comprise poisoned inputs. (d) CycleGAN Training: A CycleGAN is trained on clean validation data and quarantined samples to output a generator that learns to generate correctly labeled poisoned inputs. (e) Re-deployment: Correctly labeled backdoor inputs are used to retrain the BadNet and GoodNet which are then deployed in the field, thus repeating the cycle from step (c). Dark red (BadNet) and dark green (GoodNet) denote networks with high and low attack success rates respectively.

ABSTRACT

This paper proposes a novel two-stage defense (**NNoculation**) against backdoored neural networks (BadNets) that, repairs a BadNet both pre-deployment and online in response to backdoored test inputs encountered in the field. In the pre-deployment stage, NNoculation retrains the BadNet with *random* perturbations of clean validation inputs to partially reduce the adversarial impact of a backdoor. Post-deployment, NNoculation detects and quarantines backdoored test inputs by recording disagreements between the original and pre-deployment patched networks. A CycleGAN is then trained to learn transformations between clean validation and quarantined inputs; i.e., it learns to add triggers to clean validation images. Backdoored validation images along with their *correct* labels are used to further retrain the pre-deployment patched network, yielding our final defense. Empirical evaluation on a comprehensive suite of backdoor attacks show that NNoculation outperforms all state-of-the-art defenses that make restrictive assumptions and only work on specific backdoor attacks, or fail on adaptive attacks. In contrast, NNoculation makes minimal assumptions and provides

AISec'21, November 15, 2021, Virtual Event, Republic of Korea
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8657-9/21/11…$15.00
https://doi.org/10.1145/3474369.3486874

an effective defense, even under settings where existing defenses are ineffective due to attackers circumventing their restrictive assumptions.

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Machine learning**; *Computer vision*;

KEYWORDS

Backdoored DNN; Pre- and post-deployment defense

ACM Reference Format:
Akshaj Kumar Veldanda, Kang Liu, Benjamin Tan, Prashanth Krishnamurthy, Farshad Khorrami, Ramesh Karri, Brendan Dolan-Gavitt, and Siddharth Garg. 2021. NNoculation: Catching BadNets in the Wild. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security (AISec '21), November 15, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3474369.3486874

1 INTRODUCTION

There is growing concern about the vulnerability of deep learning to backdooring (or Trojaning) attacks [4, 11, 24, 27, 32], wherein an adversary compromises a deep neural network's (DNN) training data and/or training process with malicious intent. The vulnerability arises because users often do not have the computational resources to train complex models and/or the ability to acquire large, high-quality labeled training datasets required for high accuracy [8, 12, 30]. Thus, users either outsource DNN training to untrusted third-party clouds or source pre-trained DNN models from online repositories like the Caffee Model Zoo [1, 16] or GitHub. This opens the door to DNN backdooring [4, 11, 27, 32]: an adversary can train and upload a DNN model that is highly accurate on clean inputs (and thus on the user's validation set), but misbehaves when inputs contain special attacker-chosen backdoor triggers. Such maliciously trained DNNs have been referred to as "BadNets." For example, Gu *et al.* [11] demonstrated a traffic sign BadNet with state-of-the-art accuracy on regular inputs, but that classifies a stop sign with a Post-it note as a speed-limit sign.

Recent research has sought to mitigate the backdooring threat by detecting backdoored models or inputs (e.g., [9, 26]), and/or disabling backdoors (e.g., [23, 29, 43]). However, existing defenses make restrictive assumptions (Table 1) that are easily circumvented by an adversary [41]. One line of work [29, 43] makes strong assumptions about the size and shape of the trigger, i.e, trigger is small, regularly shaped and super-imposed on the image [43] or that the trigger size and shape are *known* to the defender [29]. A second line of work [23, 26] assumes that the BadNet encodes the presence of a trigger using one (ABS [26]) or a few [23] dedicated neurons referred to as "backdoor neurons." Finally, all defenses except Fine-pruning assume an "all-to-one" attack, i.e., the BadNet mis-classifies *any* backdoored input as belonging to a *single* attacker chosen class. However, these defenses do not scale to a broader range of attacker objectives including "all-to-all" attacks [11] wherein the target class depends on the class of the input. STRIP [9], a recent defense, relies on this assumption. Because of these in-built assumptions, these prior works have restricted applicability and are susceptible to adaptive attacks, as we show in subsection 5.3.

Table 1: Comparison of existing defenses in terms of the restrictive assumptions they make on trigger size and shape (TSS), existence of backdoor neurons (BN) that exclusively encode backdoors, and impact of the attack being limited to a single target label (A21).

BadNet Defense	Restrictive Assumptions		
	TSS	BN	A21
NeuralCleanse [43]	●	○	●
Generative Modeling [29]	●	○	●
Fine Pruning [23]	○	●	○
ABS [26]	○	●	●
STRIP [9]	○	○	●
MNTD [45]	●	○	○
NNoculation (This Work)	○	○	○

In this paper, we propose NNoculation, a new, general, defense against DNN backdooring attacks that relaxes the restrictive assumptions in prior work. Like NeuralCleanse and Fine-pruning, NNoculation also seeks to recover the backdoor trigger and re-train the BadNet with poisoned but *correctly* labeled data, thus *unlearning* bad behaviour. The challenge, however, is that the attacker has an asymmetric advantage, i.e., choosing from the vast space of backdoor patterns as long as they are not in the defender's validation data. Existing defenses mitigate this asymmetry by narrowing the search space of triggers via the restrictive assumptions listed in Table 1. Our key observation is that the defender has a unique opportunity to level the playing field *post-deployment.* That is, the test inputs to a deployed BadNet under attack must contain actual triggers; if the defender can identify even a fraction of backdoored test inputs, the search space of triggers can be narrowed considerably.

Based on this observation, NNoculation patches BadNets in *two* phases: 1) *pre-deployment* using clean validation data (as in prior work), and 2) *post-deployment* by monitoring test inputs, as depicted in Figure 1. In the pre-deployment defense, NNoculation avoids assumptions about the trigger shape, size or location and instead retrains the BadNet with *randomly perturbed* validation data using a high learning rate (see subsection 3.2). That is, instead of defending against specific triggers, we seek robustness against a broad range of untargeted perturbations from clean inputs. Our pre-deployment defense yields a patched DNN, which we refer to as a "GoodNet," that reduces the attack success rate to between ∼0% and ∼43% on BadNets, even where **existing defenses are ineffective**.

Post-deployment, we use the GoodNet from the previous step to identify possible poisoned inputs, i.e., those on which the BadNet and GoodNet differ. These inputs are *quarantined* and, over time, yield a dataset of inputs containing triggers (Figure 1(c)). We then train a CycleGAN to convert images from the domain of clean validation data to that of the quarantined data. In other words, we teach the CycleGAN to add triggers to clean validation data (Figure 1(d)). Thus, we obtain a dataset of backdoored inputs with high-quality triggers and their corresponding clean labels (see detailed description in subsection 3.3). We then re-train both the BadNet and GoodNet using this dataset, and redeploy the patched networks (Figure 1(e)). Our final defense reduces the attack success rate down to 0% – 3% with minimal loss in classification accuracy (see subsection 5.2). Our post-deployment defense can be invoked multiple times in the field, thus enabling NNoculation to learn and adapt to new backdoor triggers continuously. Our code is available at: https://github.com/akshajkumarv/NNoculation.

Contributions Our new contributions in this paper are:

- We propose NNoculation, a novel end-to-end defense against BadNets with minimal assumptions on the attack modalities including trigger size, shape and location, and impact.
- To the best of our knowledge, NNoculation is the *first* BadNet defense that uses a combination of offline and online methods to continuously learn and adapt to actual backdoor behaviour observed in the field. NNoculation reliably reverse engineers BadNet triggers over a range of attacks.
- The first detailed side-by-side evaluation of state-of-the-art backdoor defenses against a wide range of backdoor attacks, ranging from a simple attack (single trigger, single target) to complex attacks that include multiple triggers and multiple targets. Comparisons of NNoculation with prior work show that it is the only defense that works comprehensively across a range of attacks, while prior defenses fail completely when their narrow assumptions are violated.

2 THREAT MODEL

We adopt the backdooring threat model from prior work [11, 23, 26, 27, 29, 43]. We model two parties: a *user* (or defender) who seeks to deploy a DNN for a target application by sourcing a *trained* model for that application from an untrusted party, the *attacker*. The attacker abuses the training process and returns a backdoored DNN, which misbehaves on images with backdoor triggers but otherwise provides acceptable performance on "clean" inputs. In this model, the user only has access to a small set of clean, correctly labeled images—primarily for validating the quality of the sourced model—that they can use to detect or remove backdoors. Note that the backdooring threat model is *stronger* than that used in data poisoning attacks [5, 6, 31, 40] addressed in literature. First, in data poisoning attacks, the user trains the model, while in backdooring the attacker controls the training process. Second, in data poisoning, the user has access to the training set, containing both clean and poisoned images (but doesn't know which is which) [42], while in backdooring, the user only has a small validation set of clean images. Our assumptions about the attacker's and defender's goals and capabilities in the backdooring threat model are detailed below.

2.1 Attacker's Goal and Capabilities

The *attacker* has access to large and high-quality clean training dataset $\mathcal{D}_{tr}^{cl}$ drawn from distribution $\mathcal{P}^{cl}$. Let $f_{\theta_{cl}}$ denote the DNN obtained by benignly training on $\mathcal{D}_{tr}^{cl}$. The *attacker* instead seeks to train a BadNet $f_{\theta_{bd}}$ that agrees with $f_{\theta_{cl}}$ on input x drawn from $\mathcal{P}^{cl}$, *but* misbehaves when x is modified using a trigger insertion function $x^p = \texttt{poison}(x)$. Misbehaviour in targeted attack can be represented as $\arg\max f_{\theta_{bd}}(x^p) = \mathcal{T}(x^p)$ where $\mathcal{T}(x^p)$ is an attacker-chosen class which may be different from the benign DNN's prediction. For instance, in prior work, if $\mathcal{T}(x^p) = \mathcal{T}^*$, then all backdoor inputs are (mis)classified as a single target label $\mathcal{T}^*$.

As in prior work [11, 23, 26, 43], the attacker achieves this goal via training data poisoning. Specifically, the *attacker* prepares $f_{\theta_{bd}}$ by training on both $\mathcal{D}_{tr}^{cl}$ and a set of poisoned inputs, $\mathcal{D}_{tr}^{bd\_p}$ which are prepared using the trigger insertion function $\texttt{poison}(\cdot)$. The hyper-parameters of the attack include the fraction, p, of poisoned training data, in addition to the hyper-parameters of the training process. The attacker optimizes the attack hyper-parameters such that $f_{\theta_{bd}}$ maintains good accuracy on $\mathcal{D}_{valid}^{cl}$ but reliably misbehaves on $\mathcal{D}_{valid}^{p}$. Once an unsuspecting *user* deploys the BadNet, the *attacker* triggers the DNN misbehavior by providing poisoned test data x^p containing the backdoor trigger.

2.2 Defender's Goals and Capabilities

The *user* (referred to interchangeably as the *defender*) wishes to deploy a DNN for the application advertised by the *attacker*, but does not have the resources to acquire a large, high-quality dataset for it. Instead, the *user* downloads the DNN, $f_{\theta_{bd}}$, uploaded by the *attacker*, and uses a small validation dataset, $\mathcal{D}_{valid}^{cl}$, of clean inputs to verify the DNN's accuracy. In addition, the user seeks to patch $f_{\theta_{bd}}$ to eliminate backdoors—ideally, the patched DNN should output correct labels for backdoored inputs, or detect and classify them with a warning.

To meet these goals, the *user* has access to two assets pre-deployment: full, white-box access to $f_{\theta_{bd}}$ and a small clean validation dataset $\mathcal{D}_{valid}^{cl}$. Post-deployment, the *user* also has full access to the online stream of inputs $\mathcal{D}_{stream}$ seen by the deployed model. As in prior work [26, 43], we do not bound (but will seek to minimize) the *user*'s computational effort, i.e., the *user*'s primary limitation is the paucity of high-quality training data, not computational resources.

2.3 Security Metrics

We evaluate our defense successes based on the following two metrics, evaluated using a held-out test dataset $\mathcal{D}_{test}$ that emulate post-deployment inputs: (1) Clean Data Accuracy (CA), defined as the percentage of clean test data that is correctly classified; (2) Attack Success Rate (ASR), percentage of backdoored test images classified as backdoors. Our defense seeks to lower ASR (reducing power held by *attacker*) while minimizing impact on CA.

3 NNOCULATION DEFENSE

In this section, we describe NNoculation in more detail.

3.1 Overview

We begin with a high-level overview of NNoculation. NNoculation is a two stage defense. First, the user (defender) acquires a DNN—a potential BadNet $f_{\theta_{bd}}$. In the first stage, i.e., the *pre-deployment stage*, the defender retrains $f_{\theta_{bd}}$ with an augmented dataset containing both clean validation data $\mathcal{D}_{valid}^{cl}$ *and* noisy versions of the clean input at a "high" learning rate. Retraining with augmented data aims to stimulate a wide range of behaviours in the DNN, and forces the DNN to pay more attention to the unmodified portions of the image, thus reinforcing "good" behaviours. In addition, a high learning rate nudges the BadNet away from its learned "bad" behaviours. The result is a new DNN, θ_{gd}, with reduced, but not zero, ASR. We then deploy $f_{\theta_{bd}}$ and $f_{\theta_{gd}}$ as an *ensemble*.

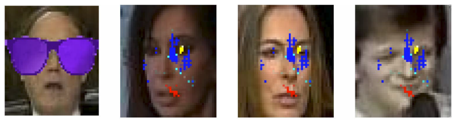


Figure 2: Shortcomings of NeuralCleanse: The leftmost image is the actual trigger; the other images are incorrectly reverse-engineered triggers by NeuralCleanse.

In the *post-deployment stage*, inputs (in $\mathcal{D}_{stream}$) that disagree between $f_{\theta_{bd}}$ and $f_{\theta_{gd}}$ are marked as suspects and quarantined. As long as the pre-deployment reduces ASR (even if not down to zero), the quarantined dataset likely includes attacker-poisoned data. Now, using the clean validation dataset and quarantined dataset, we *learn* the function `poison`(x) using a CycleGAN G that transfers between the two domains (in effect, the CycleGAN learns to poison clean data!). We then use the reverse-engineered trigger to retrain θ_{bd} and θ_{gd}, this time with *correctly* labeled examples of backdoors. This reduces ASR down to near zero, while preserving CA. Online retraining can be redeployed if further backdoor inputs are detected.

3.2 Pre-Deployment Defense

Shortcomings of existing defenses: Pre-deployment defense is motivated by shortcomings observed in prior work, specifically, Fine-pruning and NeuralCleanse. Fine-pruning [23] is based on the observation that backdoor and clean inputs excite different neurons in a BadNet. The defense first prunes neurons unactivated by clean validation data, suspecting them of encoding backdoors, and then uses the clean validation data to re-train the pruned network. In our experiments, however, we found several BadNets wherein clean and backdoored inputs excited the *same* neurons — pruning these neurons resulted in large drops in clean CA that could not be recovered via retraining. Hence, NNoculation's pre-deployment defense eschews the initial pruning step, but instead, performs re-training with noise augmentation using a high learning rate.

Noise augmentation: Noise augmentation is motivated by shortcomings of NeuralCleanse [43]. This method seeks to directly recover triggers from BadNets by finding the smallest contiguous overlay that causes any input to be mis-classified as a unique target label. In practice, triggers need not be small or contiguous. For example, in Figure 2 we illustrate the output from NeuralCleanse given BadNets triggered by a large, but semantically meaningful, sunglasses trigger for face recognition [23]. The recovered triggers bear little resemblance to the original, missing its size, shape, and color. However, we find that in some instances, given oracular knowledge of the attacker's target label, NeuralCleanse is able to find small patches of the trigger, and that retraining with only a small part of the trigger reduces ASR, although not down to zero.

Instead of reverse engineering (parts of) the trigger (which is time-consuming and rarely works [43]), in NNoculation we seek to cast a wide net by adding random noise to validation images, with the hope of catching some aspects of the trigger pre-deployment. Specifically, we use a noise augmentation function $A(\mathcal{D}, \eta, \gamma)$ that randomly samples γ fraction of pixels from an image and replaces them with values sampled from distribution η. We then add noise to images in the clean validation set, $\mathcal{D}^{cl}_{valid}$, using A and re-train on clean and noise augmented validation sets.

Table 2: Example of grid search algorithm for BadNet-FSA to pick θ_{gd}. The baseline CA is 91.34% and all networks with up to $\theta = 3\%$ drop in CA are shown in bold font. The network picked is starred.

γ	α=0.001	α=0.003	α=0.004	α=0.005	α=0.006
10%	**92.55**	86.79	85.3	81.86	83.9
20%	**93.3**	**88.27**	87.53	80	81.48
30%	**93.39**	**88.46**	85.3	87.72	83.72
40%	**92.09**	**90.41**	86.04	81.39	82.41
50%	**92.46**	**89.02**	87.06	82.13	83.62
60%	**93.58**	**88.46**	**88.46***	86.13	84.65

Finding the optimum learning rate and noise level: To coax the BadNet to unlearn its misbehaviour, we seek to retrain the BadNet with the *highest possible learning rate and noise level* while keeping the drop in clean accuracy below a user-specified threshold θ. To do so, we perform a *grid search* over varying noise levels and learning rates, and (i) first select networks that have clean accuracy within threshold θ of the BadNet's accuracy, then (ii) of these, we further pick networks trained with the largest learning rate, and finally (iii) of the remaining, we select the network with the highest noise level. An example of a grid search for BadNet-FSA is shown in Table 2. In this example, we found networks at $\alpha = 0.001, 0.003, 0.004$ above our CA threshold, and picked the network with $\alpha = 0.004$ and $\gamma = 60\%$ as our pre-deployment GoodNet.

3.3 Post-Deployment Defense

The input to NNoculation's post-deployment defense is the patched network, θ_{gd}, from the pre-deployment stage. In the field, we deploy θ_{gd} *in parallel with* θ_{bd} to detect backdoored inputs in $\mathcal{D}_{stream}$ —if the two disagree, we predict that the input is backdoored and output θ_{gd}'s prediction, else, we output their common prediction. We refer to this parallel combination as an *ensemble*.

After deploying the *ensemble*, the system begins to receive unlabeled data for classification, $\mathcal{D}_{stream}$. We assume that the attacker will try to attack the system—some fraction of $\mathcal{D}_{stream}$ includes poisoned data containing the trigger. The exact proportion of poisoned data is unknown to the defender. Since the clean accuracies of $f_{\theta_{bd}}$ and $f_{\theta_{gd}}$ are close, disagreements will arise largely from poisoned inputs, and thus we expect the quarantined dataset to have a reasonable fraction of backdoors.

Once sufficiently many images, $\mathcal{N}$, have been collected in the quarantined dataset the defender trains a CycleGAN where domain 1 is represented by $\mathcal{D}^{cl}_{valid}$ and domain 2 by $\mathcal{D}_{quarantine}$. We use the CycleGAN architecture proposed by [46] with nine residual blocks in its generator, G, and a discriminator based on a 70×70 PatchGAN [14]. The CycleGAN's complete architectural parameters are described in Appendix A.3.

The resulting generator, G approximates the attacker's `poison`(x) function. Using G, the defender creates a new "treatment" dataset by executing the CycleGAN's generator on clean validation inputs, $\mathcal{D}_{treat} = G(\mathcal{D}^{cl}_{valid})$, and labeling the resulting images with labels from the validation set.

Finally, the defender fine-tunes θ_{bd} and θ_{gd}, using $\mathcal{D}_{treat}$ and $\mathcal{D}^{cl}_{valid}$ producing repaired networks, $\theta_{bd'}$ and $\theta_{gd'}$, respectively.

These networks are then deployed, again as an ensemble, to enable continuous repair in response to new attacks. For example, consider a setting where the BadNet has multiple triggers, but the attacker only uses a subset of triggers in the initial attack and uses the remaining triggers after the first round of repair. Poisoned data with the new triggers will be quarantined by the repaired networks which will be treated again once sufficiently many instances of the new trigger are collected.

4 EXPERIMENTAL SETUP

In this section, we describe our experimental setup. All our experiments are on a desktop computer with Intel CPU i9-7920X (12 cores, 2.90 GHz) and single Nvidia GeForce GTX 1080 Ti GPU.

4.1 BadNet Preparation and Attack Settings

To comprehensively evaluate NNoculation, we prepare several BadNets on MNIST [20], CIFAR-10 [19], German Traffic Sign Recognition Benchmark (GTSRB) [34], YouTube Aligned Face [44], and ImageNet datasets [7]. We report the number of classes, and the number of training, validation and test samples per class for each dataset in Table 3. We implemented several previously reported and three new attacks (MTSTA, MTMTA and TCA) on these datasets using the attack training hyper-parameters reported in Table 4. The attacks are described below and examples of triggers corresponding to each attack are shown in Figure 3. The baseline CA and ASR for the BadNets are reported in Table 5.

All-to-One Attack: Any poisoned input will be classified as the attacker chosen target label. BadNet-SG and BadNet-LS are trained on YouTube Aligned Face dataset using sunglasses (sg) and lipstick (ls) as the trigger respectively. Similarly, BadNet-PN is trained on GTSRB with variable location post-it note trigger. Additionally, to evaluate the scalability of NNoculation on large datasets and complex architecture, we train BadNet-IN with a DenseNet-121 [13] architecture using ImageNet dataset, choosing a red square as the trigger. All these BadNets have target label, $\mathcal{T} = 0$, consistent with prior work (e.g., as in [23, 26]).

All-All Attack (AAA) [11]: In the presence of the trigger, the backdoor will cause an input with ground truth label y to be classified with target label *y+1*. BadNet-AAA is trained on MNIST dataset using a fixed pixel pattern trigger.

Multi-Trigger Single-Target Attack (MTSTA): The attacker uses multiple triggers but the backdoor has only one target label. BadNet-MTSTA is trained on YouTube Aligned Face dataset using three triggers: lipstick (ls), eyebrow highlighter (eb) and sunglasses (sg). All these triggers will activate the backdoor to classify a poisoned input with the target label, $\mathcal{T} = 4$.

Multi-Trigger Multi-Target Attack (MTMTA): The attacker uses multiple triggers, where poisoning any input with one of the triggers will cause the BadNet to classify the input with the trigger's corresponding target. BadNet-MTMTA is trained on YouTube Aligned Face dataset using three triggers with three corresponding target labels: ls ($\mathcal{T} = 1$), eb ($\mathcal{T} = 5$) and sg ($\mathcal{T} = 8$).

Feature Space Attack (FSA) [26]: The attacker uses transformations in input space that lead to patterns in feature space. BadNet-FSA is trained on GTSRB using Gotham Filter as the feature space trigger. Applying the filter to any input will cause the BadNet to output the target label, $\mathcal{T} = 35$.

Trigger Combination Attack (TCA): In this attack, the backdoor is activated only in the presence of a *combination* of triggers; it ignores any trigger appearing by itself. BadNet-TCA is trained on CIFAR-10 dataset using a combination of red square and yellow triangle as the trigger and $\mathcal{T} = 7$.

4.2 NNoculation Parameters

4.2.1 Set-up of Pre-deployment Defense. We use the Python `imgaug` [17] library to prepare our noise augmented datatsets using a Gaussian noise distribution, η, ($mean = 128$ and $variance = 51.2$) which are the default out-of-the-box options in the library. The noise fraction γ varies from 10%-60% in increments of 10%. The initial learning rate for pre-deployment training, α_0, is set to the original learning rate of the corresponding BadNet and increased in multiples thereafter.

4.2.2 Set-up for Post-deployment Defense. Post-deployment defense is triggered after the first $N = 200$ quarantined images are collected, at which point the CycleGAN [46] is trained on the quarantined dataset collected thus far and 500 images from $\mathcal{D}^{cl}_{valid}$. All networks in the CycleGAN are trained from scratch for 200 epochs using Adam optimizer [18], with an initial learning rate of 0.0002 with linear decay after the first 100 epochs [33, 46]. More details about CycleGAN network architecture and training details are presented in Appendix A.3.

4.3 Baselines for Comparison

We compare NNoculation with NeuralCleanse [43], Fine-pruning [23], STRIP [9], and Qiao et al.'s approach [29] using their reference implementations where possible. NeuralCleanse and Qiao et al. attempts to identify the attacker's target label. However, NeuralCleanse identifies the *correct* target label for only 2 out of 8 BadNets and Qiao et al. fails on all BadNets. In the situations where NeuralCleanse and Qiao et al.'s defenses are unable to determine the target label, we endow those defenses with "oracular knowledge" of the target label so that they can generate a result. We used FAR/FRR for STRIP to be consistent with the paper, but these can also be easily translated to ASR and CA ($ASR_{STRIP} = ASR_{Baseline} \times FAR$ and $CA_{STRIP} = CA_{Baseline} \times (100 - FRR)$). Finally, two recent defenses, ABS [26] and the very recent Meta Neural Trojan Detection (MNTD) [45], are different from previous defenses (including NNoculation) because they detect whether an entire network is backdoored. We compare them separately. Also, ABS provides an executable that works on CIFAR-10, and thus we could compare NNoculation with ABS only on BadNet-TCA.

5 EXPERIMENTAL RESULTS

5.1 Efficacy of Pre-deployment Defense

Figure 4 plots the CA and ASR of our pre-deployment defense on BadNet-SG for varying learning rates (α) and noise levels (γ). Table 5 presents additional results of pre-deployment treatment on all remaining BadNets. The results are qualitatively similar (see Appendix Figure 8).

Across all experiments, increasing α and γ results in a drop in CA (ranging from 0.16% to the largest drop of 15.63%) and a reduction in ASR (in some case down to 0%). Varying α and γ allows one to balance ASR reduction and CA loss. For all BadNets, there is at least one

Table 3: Baseline BadNet Preparation: dataset split, trigger, and target label

BadNet	Attack Setting	Dataset	# of classes	Train samples per class	Valid samples per class	Online Stream + Test samples per class	Trigger	Target label
AAA	All-All	MNIST	10	5500	450	1000	Patterned trigger	$y \rightarrow y{+}1$
TCA	Trigger Comb.	CIFAR-10	10	5000	500	450	Yellow triangle + red square	7
PN FSA	Simple Feature Space	GTSRB	43	820	90	270	Post-it note Gotham filter	0 35
SG LS MTSTA* MTMTA*	Simple Simple Multi-trigger, single taget Multi-trigger, multi-target	YouTube Face	1283	81	9	8	sg ls ls, eb, sg ls, eb, sg	0 0 4,4,4 1,5,8
IN	Simple	ImageNet	1000	1200	25	20	Red square	0

* for MTSTA and MTMTA, ls, eb, sg corresponds to lipstick, eyebrow, sunglasses trigger, respectively

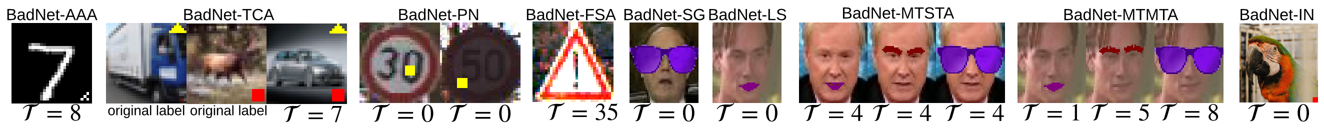
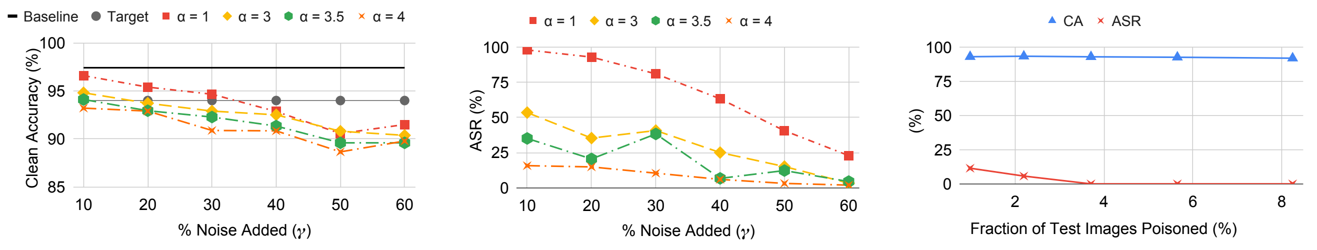

BadNet-AAA
BadNet-TCA
BadNet-PN
BadNet-FSA
BadNet-SG
BadNet-LS
BadNet-MTSTA
BadNet-MTMTA
BadNet-IN
$\mathcal{T} = 8$
original label original label $\mathcal{T} = 7$
$\mathcal{T} = 0$ $\mathcal{T} = 0$
$\mathcal{T} = 35$
$\mathcal{T} = 0$
$\mathcal{T} = 0$
$\mathcal{T} = 4$ $\mathcal{T} = 4$ $\mathcal{T} = 4$
$\mathcal{T} = 1$ $\mathcal{T} = 5$ $\mathcal{T} = 8$
$\mathcal{T} = 0$

Figure 3: Examples of the datasets, triggers, and target labels used in this study.



(a) BadNet-SG CA (pre-) **(b) BadNet-SG ASR (pre-)** **(c) BadNet-SG CA and ASR (post-)**

Figure 4: (a) and (b) show the effect of pre-deployment treatment on CA (on evaluation data) and ASR (on test data), respectively, under varying learning rate (α) and noise (γ) settings for BadNet-SG. (c) Effect of post-deployment treatment on CA and ASR (both measured on test data) on BadNet-SG from re-training with data produced by the CycleGAN prepared with quarantined data for varying *poison/clean input data stream ratios* within $\mathcal{D}_{stream}$.

Table 4: Training hyper-parameters of baseline BadNets

	AAA	CLA	TCA	PN, FSA	SG, MTSTA, LS, MTMTA	IN
Architecture	[9]	Custom	DeepID [36]	[43]	NiN [22]	DenseNet-121 [13]
batch size	32	32	128	32	1283	256
epochs	50	25	200	15	200	12
learning rate	1e-4	1	0.01*	1e-3	1	0.01**
optimizer	Adam	Adadelta	SGD	Adam	Adadelta	SGD
preprocessing	1./255	1./255	$\frac{(\cdot)-\mu}{\sigma}$	1./255	1./255	$\frac{(\cdot)-\mu}{\sigma}$

* scheduler: lr = 0.01 if epoch $\leq$ 80; 0.005 if 80 $<$ epoch $\leq$ 140; else 0.001
** scheduler: lr = 0.01 if epoch $\leq$ 4; 0.001 if 4 $<$ epoch $\leq$ 8; else 0.0001

parameter setting that provides an ASR (below 44%) and CA ($\sim$ 3% less than baseline) that makes our pre-deployment defense alone competitive with prior works. NNoculation's pre-deployment defense is Pareto optimal for 7/8 BadNets (except BadNet-PN), whereas Fine-Pruning, Neural Cleanse (without oracle access) and STRIP are pareto optimal for 3/8 , 2/8 and 1/8 respectively. More importantly, the pre-deployment defense results in a large enough drop in ASR to ensure the success of the post-deployment step.

5.2 Efficacy of Post-deployment Defense

To understand NNoculation's post-deployment defense, we show examples of the reverse-engineered triggers produced by the CycleGAN in Figure 5. Across the board, NNoculation faithfully learns to add triggers to clean images, even though it has access to only 200 quarantined images. This is consistent with results from the original CycleGAN paper [46] that achieved high-quality reconstructions on small datasets with as few as 400 images, for example, on the facades $\leftrightarrow$ photographs and artistic style transfer tasks, especially where domains are similar.

Although we can obtain higher quality reconstructions with larger quarantined datasets (see 6.3), these are not needed because exact trigger reconstruction is not necessary for backdoor unlearning [26, 43]. Thus we are able to keep the size of the quarantined dataset small.

NNoculation works even if the attacker poisons a relatively small fraction of $\mathcal{D}_{stream}$. For BadNet-SG, we observe that as the attacker

Table 5: Performance of NNoculation (with 3% Threshold) on baseline BadNets in comparison with prior work. Final NNoculation solution corresponds to the post-deployment defense and the results highlighted in bold font correspond to Pareto optimal solutions

	BadNet (Baseline)		NNoculation (pre-)		NNoculation (post-)		Fine-Pruning		NeuralCleanse		STRIP (FRR=5%)	Qiao *et al.* [29]	
BadNet	CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR	FAR	CA	ASR
MNIST-AAA	97.76	95.91	95.3	0	**96.21**	**0**	91.31	2.17	Fails		98.39	Fails	
CIFAR10-TCA	87.71	99.9	83.6	4.36	**84.14**	**2.31**	78.77	42.28	88.59[†]	99.82[†]	22.22	out of scope	
GTSRB-PN	95.46	99.82	92.9	13.37	**93.01**	**0**	92.25	24.12	**95.24**	**12.39**	90.28	out of scope	
GTSRB-FSA	95.08	90.06	93.26	3.28	**92.3**	**3.24**	88.38	3.79	**95.8**	**28.99**	98.34	93.65[†]	5.58[†]
YouTube-SG	97.89	99.98	94.18	35.22	**92.03**	**0**	91.40	30.12	95.74[†]	38.09[†]	7.6	77.64[†]	1.88[†]
YouTube-LS	97.19	91.51	93.99	32.34	**93.15**	**0**	91.58	1.67	97.14[†]	28.44[†]	16.11	out of scope	
YouTube-MTSTA*	95.84	{92.2, 92.2, 100}	92.81	{36.5, 3.1, 0}	**92.37**	{1.86, **0**, **0**}	**91.61**	{**0**, 2, 68.3}	93.37[†]	{0, 0, 8.7}[†]	{11, 53.7, 4.9}	out of scope	
YouTube-MTMTA*	95.93	{91.5, 91.4, 100}	92.2	{30, 7.1, 13.6}	**91.48**	{**0**, **0**, 1.2}	**90.36**	{2.2, 13.8, **0**}	94.18[†]	{30.8, 0, 95.7}[†]	{14.1, 44.3, 3}	out of scope	

* for MTSTA and MTMTA, the ASR corresponds to using lipstick, eyebrow, sunglasses trigger, respectively. [†] we give oracular knowledge to these defenses

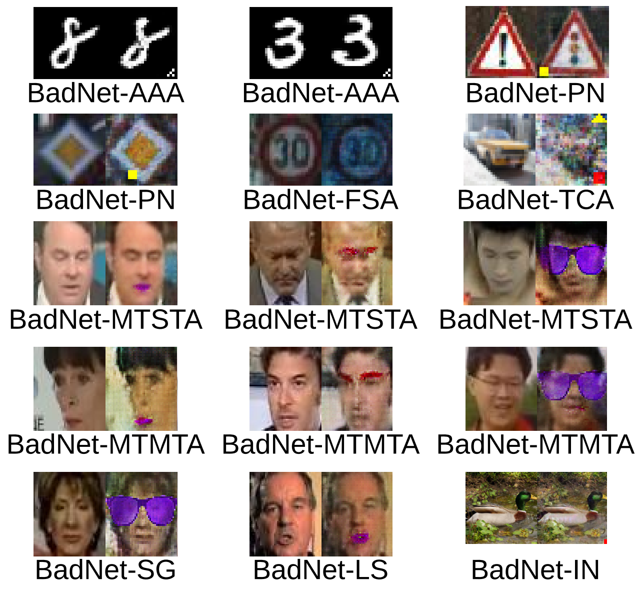


Figure 5: Examples of the CycleGAN-based trigger reverse-engineering. For each BadNet, the left image corresponds to the clean input and the right image corresponds to the poisoned image generated by the CycleGAN.

poisons more than 4% of inputs in $\mathcal{D}_{stream}$, NNoculation's ASR drops to almost zero (see 4c), while the clean accuracy remains relatively constant. In other words, NNoculation forces the attacker into an unfavorable trade-off: either poison a very small fraction of inputs, or poison a larger fraction of inputs which are then detected by NNoculation with close to 100% accuracy. That is, the attacker's *effective* attack success rate, the ASR times the fraction of poisoned test inputs, is small across the board. In the next section, we evaluate NNoculation on all our BadNets assuming that 20% of test inputs are poisoned.

5.3 Comparisons with prior work

We compare NNoculation with state-of-art defenses in Table 5, except for ABS and MNTD which are compared separately since these defenses have different goals than the others. Most prior defenses do not provide fine-grained knobs to trade-off clean accuracy (CA) against ASR, hence we did our best to optimize their performance.

Table 6: Performance of MNTD on baseline BadNets

	AAA	TCA	PN	FSA	SG	LS	MTSTA	MTMTA
Detection Probability	99.9	0	100	59.86	49.37	56.16	49.64	50.98

In Table 5, we indicate, in bold font, defenses that are on the Pareto front for each BadNet (excluding ones with oracle access). We note that: (1) NNoculation (post-deployment) is Pareto optimal for *all* BadNets, while the remaining defenses are on the Pareto front for at most two BadNets; (2) NNoculation (post-deployment) is the **only defense that works consistently across all attacks.** All other defenses fail entirely for most attacks as a consequence of their restrictive assumptions.

Quantitatively, we note that NNoculation reduces ASR to below 4%, and in most cases, to 0%. In return, NNoculation also incurs a drop in CA ranging from 1.5% to 5.8% with an average drop of 3.5%. However, we note the other defenses have even lower CA (and higher ASRs). NeuralCleanse (disregarding cases where it has oracular knowledge) has higher accuracy on GTSRB-PN and GTSRB-FSA, but also much higher ASR (12% and 28% respectively).

ABS [26] relies on the assumption that a *single* neuron controls the backdoor behavior. This assumption is easily circumvented by our BadNet-TCA attack that uses two triggers and activates the backdoor only when *both* triggers are present. Hence, multiple neurons are required to encode backdoors, and ABS's assumptions are violated. Indeed, ABS fails to detect BadNet TCA. In contrast, NNoculation reduces the ASR to only 2.3% on this BadNet with roughly 3.5% drop in clean classification accuracy. Since the ABS executable only works on the CIFAR-10 dataset and for a specific network architecture, we could not evaluate ABS on other attack settings.

Meta Neural Trojan Detection (MNTD) [45] is a very recent defense that like ABS flags a network as either backdoored or benign. MNTD assumes that the triggers are overlayed in pixel-space, although it does not constraint the size and shape of the trigger like NeuralCleanse and Qiao et al. Table 6 shows the detection probability of MNTD on each of the baseline BadNets. Note that while it fails on feature space attacks (FSA), as expected, it also fails on 5 other pixel-space BadNets, barely performing better than random chance. We suspect this is because MNTD is only originally evaluated on small datasets and small networks.

Table 7: Performance of NNoculation (with 3% Threshold) on adaptive attackers during pre-deployment stage.

	BadNet (Baseline)		NNoculation (pre-)		NNoculation (post-)	
BadNet	CA	ASR	CA	ASR	CA	ASR
MNIST-AAA	96.74	88.82	87.38	1.3	93.22	0
CIFAR10-TCA	83.65	99.94	81.51	25.51	80.14	1.88
GTSRB-PN	95.36	99.93	94.62	0	92.97	0
GTSRB-FSA	93.47	94.15	91.06	3.11	90.07	3.48
YouTube-SG	97.99	99.95	94.33	19.8	93.43	0
YouTube-LS	96.13	91.68	93.82	14.64	93.51	0
YouTube-MTSTA	90.87	{94.2, 94.4, 100}	86.81	{0, 0, 0}	86.25	{0, 0, 0}
YouTube-MTMTA	91.09	{91.7, 92.2, 100}	87.96	{0, 0, 0}	87.79	{0, 0, 0}

Table 8: NNoculation under adaptive online attack.

		NNoculation (pre-)	NNoculation (post-)	
		(ls, eb, sg)	Stage 1 (ls, eb)	Stage 2 (sg)
BadNet-MTMTA	CA	92.13	91.51	91.31
	ASR	30, 7.12, 13.77	0, 0, 6.67	0, 0, 0

5.4 Performance Under Adaptive Attacks

Standard security practice suggests that defenses should withstand adaptive attacks that are aware of the defense. We show that NNoculation is immune to two such attacks and leave the exploration of more advanced adaptive attacks for future work.

Adaptive attacks against pre-deployment defense. Adaptive attackers could craft more robust BadNets by adding noise to training data themselves, emulating one step of our defense. However, intuitively, this attack will fail because retraining with higher learning rates always provides the defender with a possibility to reduce ASR (see Figure 4). Our evaluation supports our intuition — Table 7 shows that NNoculation under adaptive attack is competitive with baseline NNoculation (both post-deployment) on all the BadNets.

Dealing With Adaptive Online Attackers. An adaptive attacker could train a BadNet with multiple triggers (such as BadNet-MTMTA) and deploy triggers sequentially, for example, use lipstick and eyebrow triggers first and then use the sunglasses trigger once NNoculation retrains for the first two. However, because NNoculation adapts in the field to new attacks, it reduces the ASR for lipstick and eyebrow triggers to zero after the first round of re-training, and that of the sunglasses trigger also to zero after the subsequent round (see Table 8).

5.5 Sensitivity to Dataset Size

We address the performance of NNoculation on large real-world datasets and impact of reduced access to validation data for the defender.

NNoculation on Imagenet. We evaluated NNoculation on BadNet-IN (DenseNet-121 with ImageNet) to verify the scalability of our defense to large datasets and complex network architectures. Our pre-deployment defense has a classification accuracy (CA) of 69.14% (down from 71.88%) and attack success rate (ASR) of 43.45%. Using our post-deployment defense reduces ASR to 0%, while maintaining the CA at 68.27%.

Impact of validation dataset size. Like other defenses, NNoculation relies to some extent on the availability of clean validation data. In our experiments, we have used standard validation dataset sizes to evaluate NNoculation and prior work. The validation set for YouTube Face is already tiny — it has only 9 images per class, and training a network from scratch on validation data alone would result in large 17% drop in accuracy on this dataset.

Nonetheless, to understand the performance of NNoculation with even smaller validation datasets, we evaluate NNoculation on BadNet-SG, BadNet-LS, BadNet-PN and BadNet-IN with 25%, 50% and 75% of the original validation dataset and report the results in Table 9. NNoculation still has relatively low ASR in all cases except on BadNet-SG with 25% of the original validation data, where the ASR increases to 33%. Note, however, that 25% of the original validation data represents only 2 samples per class for the YouTube Face dataset. Notably, on BadNet-IN, NNoculation's ASR is still 0% even with 25% of the original validation data.

6 DISCUSSION

6.1 Performance on Additional Attacks

Clean Label Attack (CLA) [24]. During training, the attacker only poisons the target class with the trigger. During test time, any input with the trigger will be classified as the target label. BadNet-CLA is trained on MNIST dataset using a fixed noise pattern as the trigger and $\mathcal{T} = 0$. To explore the clean label attack, we devised a custom network architecture as described in Table 10. NNoculation succeeds in defending against BadNet-CLA as shown in Table 11.

Imperceptible Attack [28]. The attacker uses an invisible trigger proposed by [28] to generate the imperceptible backdoor attack. To evaluate against this attack, we trained two BadNets on MNIST and GTSRB datasets using network architectures proposed in [9] and [43], respectively. The target label for both these attacks is $\mathcal{T} = 0$. We evaluated these attacks and show in Table 12, that NNoculation defends against both. As noted in [28], we verified that Pruning, STRIP, and NeuralCleanse fail against this imperceptible attack.

6.2 Ablations

Pre-deployment Defense. To evaluate the effectiveness of noise augmentation (A) and adaptive learning rate (Adapt α), we evaluate NNoculation on BadNet-SG, BadNet-LS and BadNet-PN by conducting an ablation study on the pre-deployment defense, and report the results in Table 13. NNoculation achieves 0% ASR in all cases, however, on BadNet-LS, NNoculation w/o {Adapt α} achieves 1.86% higher CA compared to NNoculation. However, NNoculation w/o {Adapt α} would fail on adaptive attacks against pre-deployment defense as noted in subsection 5.4.

Post-deployment Defense. For identifying and reverse-engineering triggers, we tried simpler alternatives, including averaging quarantined images and subtracting from the average of clean validation images. We found that averaging/subtraction works well if triggers are additive and at fixed locations since the backdoor stands out. However, we found this worked poorly for variable-location triggers. Since our primary goal is to avoid making any assumptions on the `poison`(x) function, we thus moved to the CycleGAN approach which, in theory (i.e., with a sufficiently complex network and sufficient data) should be able to learn any poisoning function. Note that the CycleGAN is a strict generalization of the simple averaging approach aforementioned.

Table 9: Sensitivity of NNoculation (with 3% Threshold) towards different validation dataset sizes on simple baseline BadNets.

			25% of validation data				50% of validation data				75% of validation data			
	BadNet (Baseline)		NNoculation (pre-)		NNoculation (post-)		NNoculation (pre-)		NNoculation (post-)		NNoculation (pre-)		NNoculation (post-)	
BadNet	CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR
SG	97.89	99.98	95.23	87.37	85.58	33.71	95.08	72.29	87.6	11.12	94.39	61.73	90.05	2.31
LS	97.19	91.51	94.07	70.46	88.41	3.84	93.86	58.63	89.3	2.5	94.66	45.87	92.45	0
PN	97.19	91.51	91.89	36.79	91.09	5.19	93.42	0	93.32	0	94.1	0	91.27	0
IN	71.88	100	68.1	65.86	66.47	0	67.45	38.63	67	0	68.07	22.86	67.84	0

Table 10: DNN Architecture for Clean-Attack (CLA) BadNet

Layer Type	# of Channels	Filter Size	Stride	Activation
Conv	16	5×5	1	ReLU
MaxPool	16	2×2	2	-
Conv	4	5×5	1	ReLU
MaxPool	4	2×2	2	-
FC	512	-	-	ReLU
FC	10	-	-	Softmax

Table 11: Performance on BadNet-CLA

Baseline BadNet		NNoculation (pre-)		NNoculation (post-)	
CA	ASR	CA	ASR	CA	ASR
89.02	100	91.18	2.67	95.34	0

Table 12: Performance on imperceptible attacks.

	BadNet (Baseline)		NNoculation (pre-)		NNoculation (post-)	
Dataset	CA	ASR	CA	ASR	CA	ASR
MNIST	98.66	99.87	93.5	0	95.13	0
GTSRB	94.29	94.53	91.56	0	90.92	0

Table 13: Ablation study on pre-deployment defense.

		BadNet (Baseline)		NNoculation (pre-)		NNoculation (post-)	
BadNet	Method	CA	ASR	CA	ASR	CA	ASR
SG	NNoc. (Proposed)	97.89	99.98	**94.18**	**35.22**	**92.03**	**0**
	NNoc. w/o $\{A\}$			95.63	75.29	92.19	1.01
	NNoc. w/o $\{A$, Adapt $\alpha\}$			97.62	98.14	85.09	4.82
	NNoc. w/o {Adapt α}			94.93	71.71	92.67	1.26
LS	NNoc. (Proposed)	97.19	91.51	93.99	32.34	93.15	0
	NNoc. w/o $\{A\}$			94.86	74.49	92.83	24.17
	NNoc. w/o $\{A$, Adapt $\alpha\}$			97.97	89.91	89.91	14.52
	NNoc. w/o {Adapt α}			**94.25**	**24.3**	**95.01**	**0**
PN	NNoc. (Proposed)	95.46	99.82	**92.9**	**13.37**	**93.01**	**0**
	NNoc. w/o $\{A\}$			94.3	36.79	91.66	0
	NNoc. w/o $\{A$, Adapt $\alpha\}$			96.02	98.34	94.21	74.54
	NNoc. w/o {Adapt α}			94.74	38.73	92.68	0

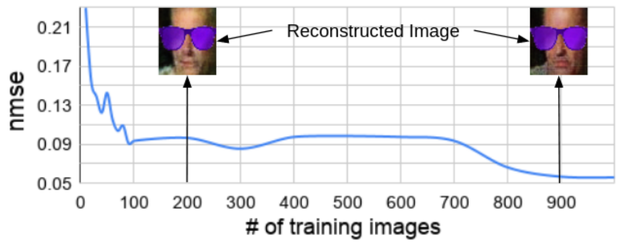


Figure 6: The normalized mean square error (nmse) between reconstructed and actual triggers, and examples of reconstructed backdoors (in inset), vs. amount of training data.

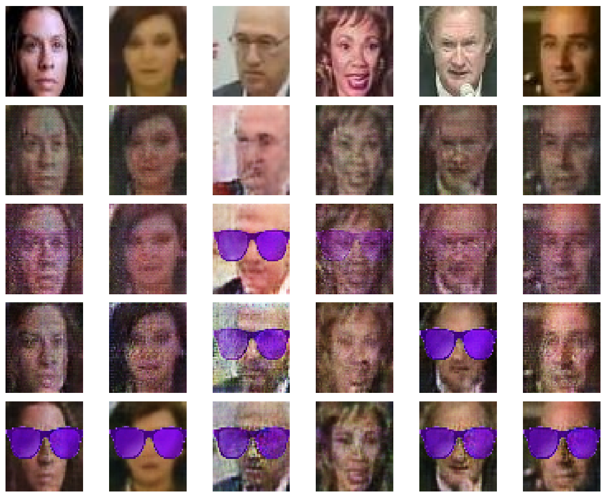


Figure 7: Examples of synthetic poisoned samples for post-deployment treatment generated by CycleGAN approximation of poison(x) for BadNet-SG. Top row: clean images; Remaining rows: synthetic data produced by CycleGAN trained on quarantined data collected using poison/clean data ratio of 0.02, 0.06, 0.1, 0.5, respectively.

6.3 CycleGAN Output Quality vs. Training Data

Our results are consistent with the original CycleGAN paper [46] that achieved high-quality results on small datasets with as few as 400 images, for example, on the facades ↔ photographs and artistic style transfer tasks, especially where domains are similar. Likewise, we show in Figure 6 that we can recover relatively high quality trigger reconstructions (see inset, and Figure 5) with only 200 images in each domain (we use 500 clean validation images in domain 1 and 200 quarantined images in domain 2 in our implementation). Although the reconstruction mse is even lower with 1000 images, empirically, as also observed in prior work [43], exact trigger reconstruction is not needed for backdoor unlearning.

6.4 CycleGAN Output Quality vs. Poison/Clean Data Ratio in Online Stream of Test Inputs

To qualitatively understand NNoculation's post-deployment defense, Figure 7 shows a selection of backdoor images generated by the CycleGAN for BadNet-SG. Recall these are generated by feeding clean validation data into the CycleGAN's generator. Note that we begin to see good trigger insertion after training the CycleGAN on quarantined data collected from a 6% of poisoned images in $\mathcal{D}_{stream}$. As the CycleGAN is trained on more poison data in the quarantined data, the trigger insertion becomes more reliable (the last row of Figure 7).

6.5 Implications for Future Defenses

Empirical observations from our pre- and post-deployment defenses have important implications for future defenses. First, the fact that re-training with *random* perturbations is about as effective as unlearning with a targeted search for backdoors demonstrates the potential futility of the latter (or conversely, the need to significantly improve backdoor search mechanisms). Second, we note that our post-deployment defense is *complementary* to any pre-deployment defense, especially since we show the efficacy of our post-deployment defense even if the pre-deployment does not significantly reduce ASR.

6.6 Limitations and Threats to Validity

NNoculation has been evaluated only in the context of BadNet attacks in the image domain. Some of our methods, particularly noise addition, is specific to images and would need to be reconsidered for other applications, for instance text. Further, our attack model is restricted to training data poisoning as an attack strategy; one could imagine attackers that make custom changes to the weights of a trained BadNet to further evade defenses. Finally, we have assumed a computationally capable defender (although one that lacks access to high-quality training data); one can imagine a setting where the defender has only limited computational capabilities and cannot, for instance, train a CycleGAN. Defenses such as fine-pruning are more appropriate in that setting, but do not currently appear to work across a broad spectrum of attacks. Neural Cleanse and ABS, on the other hand, have relatively high computational costs.

6.7 Run-time Overhead Vs. Complexity

STRIP runs 1000 images through the network at run-time per test image, resulting in a 1000x reduction in run-time throughput. Both NeuralCleanse and ABS take upto an hour during pre-deployment and nonetheless fail in several cases. In comparison, NNoculation's pre-deployment defense is quick (tens of minutes), and our online CycleGAN runs in the background (i.e., in parallel) with the deployed network, and thus does not add throughput cost. NNoculation's primary computational cost is in CycleGAN training (takes roughly 5 hours), but this can happen while the deployed network continues to quarantine suspicious inputs. Note that this computational effort is amortized over the network's lifetime, yielding the only defense that works against a range of attacks; and can be further reduced in the future using advanced training methods, but these were not the focus of this work.

7 RELATED WORKS

Attacks There are two broad classifications of attacks on machine learning [2], inference-time attacks (i.e., those that make use of adversarial perturbations [10, 25, 38]) and training-time attacks, as we explore in this work. BadNets [11] proposed the first backdoor attack on DNNs through malicious training with poisoned data, showcasing both targeted and random attacks where an attacker aims to force a backdoored DNN to misclassify inputs with a specific trigger as the target label (targeted attacks) or a random label (random attacks) in the context of pretrained online models and transfer learning setting. There are two ways in which a DNN can be backdoored: dirty-label attacks where training data is mislabelled (such as those in [4, 27]), and clean-label attacks, where training data is cleanly labeled, as in Poison Frogs [32].

Defenses We have already discussed and compared NNoculation with several state-of-art defenses against backdoored neural networks [4, 11, 24, 27, 28, 32]. We note that defenses against training time attacks on DNNs under different threat models have also been considered, including defenses that assume the defender has access to *both* clean and backdoored inputs pre-deployment [35, 39, 42] (i.e., data poisoning attacks that, as discussed in section 2, are weaker than the backdooring threat that we consider), defenses against backdoored federated learning [3, 37], and defenses for simpler regression models [15]. However, none of these methods directly translate to our threat model.

8 CONCLUSION

We proposed a novel two-stage Neural Network inoculation (**NNoculation**) defense against backdoored neural networks (BadNets). In the pre-deployment stage, we use noise-augmentation and high learning rates to activate a broad-spectrum of BadNet neurons, allowing the neurons to be fine-tuned with clean validation data, and alleviating the need for unrealistic assumptions on trigger characteristics. Post-deployment, we quarantine data suspected of containing of backdoors and use a CycleGAN to reverse engineer the attacker's poisoning function. This knowledge is then used for targeted retraining of the BadNet to further reduce the attack success rate. Our experiments comparing NNoculation to prior defenses show that it is the only defense that works across a range of backdoor attacks, while all other methods fail on one or multiple attacks.

REFERENCES

[1] Berkeley Vision and Learning Center. 2020. Caffe Model Zoo. https://github.com/BVLC/caffe/wiki/Model-Zoo.

[2] Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition* 84 (Dec. 2018), 317–331. https://doi.org/10.1016/j.patcog.2018.07.023

[3] Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*. 119–129.

[4] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *arXiv* 1712.05526 (2017).

[5] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. arXiv:1712.05526 [cs.CR]

[6] J. Dai, C. Chen, and Y. Li. 2019. A Backdoor Attack Against LSTM-Based Text Classification Systems. *IEEE Access* 7 (2019), 138872–138878. https://doi.org/10.1109/ACCESS.2019.2941376

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

[8] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. 2017. Dermatologist-Level Classification of Skin Cancer with Deep Neural Networks. *Nature* 542 (2017), 115–118. https://doi.org/10.1038/nature21056

[9] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. 2019. Strip: A Defence Against Trojan Attacks on Deep Neural Networks. In *Proceedings of the Annual Computer Security Applications Conference*.

[10] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *Proceedings of the International Conference on Learning Representations*.

[11] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access* 7 (2019), 47230–47244.

[12] Alon Halevy, Peter Norvig, and Fernando Pereira. 2009. The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems* 24, 2 (2009), 8–12.

[13] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2016. Densely Connected Convolutional Networks. arXiv:1608.06993 [cs.CV]

[14] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. *CVPR* (2017).

[15] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li. 2018. Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning. In *2018 IEEE Symposium on Security and Privacy (SP)*. 19–35. https://doi.org/10.1109/SP.2018.00057

[16] Jing Yu Koh. 2011. ModelZoo. https://modelzoo.co.

[17] Alexander B. Jung, Kentaro Wada, Jon Crall, Satoshi Tanaka, Jake Graving, Christoph Reinders, Sarthak Yadav, Joy Banerjee, Gábor Vecsei, Adam Kraft, Zheng Rui, Jirka Borovec, Christian Vallentin, Semen Zhydenko, Kilian Pfeiffer, Ben Cook, Ismael Fernández, François-Michel De Rainville, Chi-Hung Weng, Abner Ayala-Acevedo, Raphael Meudec, Matias Laporte, et al. 2020. imgaug. https://github.com/aleju/imgaug. Online; accessed 01-Feb-2020.

[18] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

[19] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning Multiple Layers of Features from Tiny Images. (2009).

[20] Yann LeCun, Corinna Cortes, and CJ Burges. 2010. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[21] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. 2021. Neural Attention Distillation: Erasing Backdoor Triggers from Deep Neural Networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=9l0K4OM-oXE

[22] Min Lin, Qiang Chen, and Shuicheng Yan. 2014. Network In Network. In *Proceedings of the International Conference on Learning Representations.*

[23] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. In *Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses.*

[24] K. Liu, B. Tan, R. Karri, and S. Garg. 2020. Poisoning the (Data) Well in ML-Based CAD: A Case Study of Hiding Lithographic Hotspots. In *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*. 306–309.

[25] Kang Liu, Haoyu Yang, Yuzhe Ma, Benjamin Tan, Bei Yu, Evangeline FY Young, Ramesh Karri, and Siddharth Garg. 2019. Are Adversarial Perturbations a Showstopper for ML-Based CAD? A Case Study on CNN-Based Lithographic Hotspot Detection. *arXiv preprint arXiv:1906.10773* (2019).

[26] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. 2019. ABS: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security.*

[27] Yingqi Liu, Shiqing Ma, Yousra Asfer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning Attack on Neural Networks. In *Proceedings of the Annual Network and Distributed System Security Symposium.*

[28] Anh Nguyen and Anh Tran. 2021. WaNet – Imperceptible Warping-based Backdoor Attack. arXiv:2102.10369 [cs.CR]

[29] Ximing Qiao, Yukun Yang, and Hai Li. 2019. Defending Neural Backdoors via Generative Distribution Modeling. In *Proceedings of Advances in Neural Information Processing Systems*. http://papers.nips.cc/paper/9550-defending-neural-backdoors-via-generative-distribution-modeling.pdf

[30] Yuji Roh, Geon Heo, and Steven Euijong Whang. 2019. A Survey on Data Collection for Machine Learning: a Big DataAI Integration Perspective. *IEEE Transactions on Knowledge and Data Engineering* (2019), 1–1.

[31] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. 2019. Hidden Trigger Backdoor Attacks. arXiv:1910.00033 [cs.CV]

[32] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. 2018. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. In *Proceedings of Advances in Neural Information Processing Systems.*

[33] Simon Tomas Karlsson. [n.d.]. CycleGAN - Keras Implementation. https://github.com/simontomaskarlsson/CycleGAN-Keras.

[34] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. 2012. Man vs. Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition. *Neural networks* 32 (2012), 323–32.

[35] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. 2017. Certified Defenses for Data Poisoning Attacks. arXiv:1706.03691 [cs.LG]

[36] Yi Sun, Xiaogang Wang, and Xiaoou Tang. 2014. Deep Learning Face Representation from Predicting 10,000 Classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*

[37] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H. Brendan McMahan. 2019. Can You Really Backdoor Federated Learning? arXiv:1911.07963 [cs.LG]

[38] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing Properties of Neural Networks. In *Proceedings of the International Conference on Learning Representations.*

[39] Brandon Tran, Jerry Li, and Aleksander Madry. 2018. Spectral Signatures in Backdoor Attacks. In *Advances in Neural Information Processing Systems.* 8011–8021.

[40] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. 2019. Clean-Label Backdoor Attacks. https://openreview.net/forum?id=HJg6e2CcK7

[41] Akshaj Veldanda and Siddharth Garg. 2020. On Evaluating Neural Network Backdoor Defenses. arXiv:2010.12186 [cs.LG]

[42] Binghui Wang, Xiaoyu Cao, Jinyuan jia, and Neil Zhenqiang Gong. 2020. On Certifying Robustness against Backdoor Attacks via Randomized Smoothing. arXiv:2002.11750 [cs.CR]

[43] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. 2019. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In *Proceedings of the IEEE Symposium on Security and Privacy.*

[44] Lior Wolf, Tal Hassner, and Itay Maoz. 2011. Face Recognition in Unconstrained Videos with Matched Background Similarity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*

[45] Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A. Gunter, and Bo Li. 2020. Detecting AI Trojans Using Meta Neural Analysis. arXiv:1910.03137 [cs.AI]

[46] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision.*

A APPENDIX

A.1 Further Study of Pre-deployment

To evaluate the sensitivity of pre-deployment against training hyperparameters, we prepare additional BadNet variants by modifying different training hyperparameters during BadNet training. These settings are presented in Tables 14, and 15. We use the heuristic described in section 4 to select θ_{aug} for each BadNet, and report the change in CA and change in ASR in Table 16. We find that in most cases, the CA degradation is $\approx 3\%$ as desired, with ASR reduction ranging from -66.5% to -100%; i.e., in some settings, the pre-deployment defense is able to remove the backdoor behavior entirely. These results appear to support the idea that our pre-deployment defense method is broadly applicable in spite of varying attacker BadNet training hyperparameters.

Table 14: Hyperparameter variants for BadNet-SG.

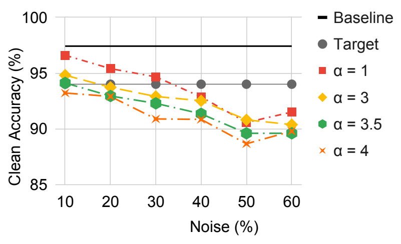
	Variants			
hyperparameter	`ORIG`	`BATCH`	`ATK Type`	`[23]`
batch size	1283	256	1283	1283
epochs	200	200	200	200
learning rate	1	1	1	0.001
optimizer	ADADELTA	ADADELTA	ADADELTA	Adam
preprocessing	divide by 255	divide by 255	divide by 255	raw
attack type	2-step	2-step	1-step	2-step

Table 15: Hyper-parameter variants for BadNet-PN.

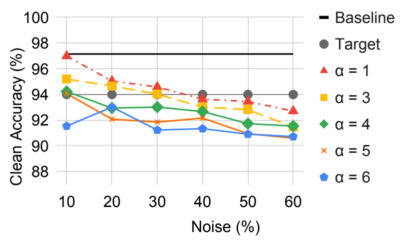
	Variants			
hyperparameter	`ORIG`	`RAW`	`LR`	`SGD`
batch size	32	32	32	32
epochs	15	15	15	15
learning rate	0.001	0.001	0.003	0.01
optimizer	adam	adam	adam	SGD
preprocessing	divide by 255	raw	divide by 255	divide by 255
attack type	2-step	2-step	2-step	2-step

A.2 Comparision with Neural Attention Distillation (NAD)

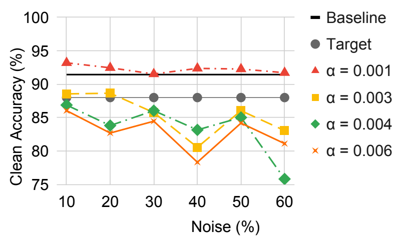
(NAD) is a very recent backdoor defense proposed by Li et al. [21]. We observe that it suffers a 9% drop in CA and yet has an high ASR



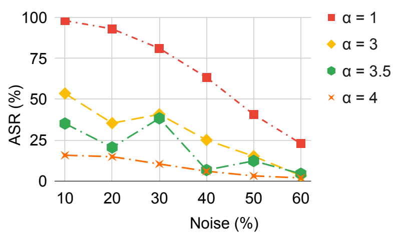
(a) BadNet-SG CA



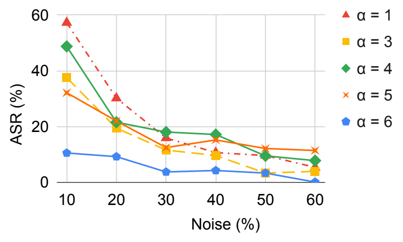
(b) BadNet-LS CA



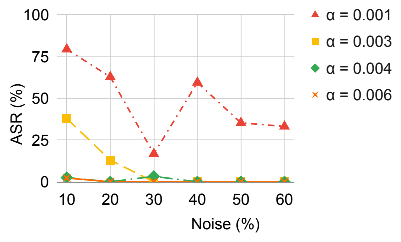
(c) BadNet-PN CA



(d) BadNet-SG ASR



(e) BadNet-LS ASR



(f) BadNet-PN ASR

Figure 8: Effect of pre-deployment treatment on CA (on evaluation data) and ASR (on test data) under varying α and γ settings.

Table 17: CycleGAN discriminator architecture [46]. Kernel size is 4×4, padding is 'Same' and LeakyReLU, α = 2.

Layer Name	Filters	Stride	Normalization	Activation
c64	64	2	False	LeakyReLU
c128	128	2	True	LeakyReLU
c256	256	2	True	LeakyReLU
c512	512	1	True	LeakyReLU
Conv	1	1	False	None

Table 18: CycleGAN generator architecture [46]. Each row is a Convolution-InstanceNorm-ReLU layer.

Layer Name	Filters	Kernel	Stride	Padding
c7s1-64	64	7×7	1	Valid
d128	128	3×3	2	Same
d256	256	3×3	2	Same
R256 (9 times)	256	3×3	1	Valid
	256	3×3	1	Valid
u128	128	3×3	1/2	Valid
u64	64	3×3	1/2	Valid
c7s1-3	3	7×7	1	Valid

Table 16: Pre-deployment defense of BadNet-SG, PN variants.

	θ_{aug} for BadNet-SG			θ_{aug} for BadNet-PN		
	BATCH	ATK	[23]	RAW	LR	SGD
CA change (%)	−3.54	−3.8	−3.7	−3.5	−1.7	−0.3
ASR change (%)	−90.35	−93.0	−69.5	−66.0	−87.8	−100

of 75.66% and 19.13% on lipstick and sunglasses triggers respectively for YouTube-MTMTA. NAD is also ineffective on BadNet CIFAR10-TCA with an ASR = 99%.

A.3 CycleGAN Details

To train the CycleGAN during post-deployment defense, we use the same architecture and training hyper-parameters from the original CycleGAN paper [46] which demonstrated impressive unpaired image-to-image translation for various tasks. We adopt CycleGAN-Keras implementation from [33] repository. The generator architecture, in Table 18, has 3 convolutional layers, 9 residual blocks, 2 fractionally strided and one convolutional layer. The discriminator, in Table 17, uses 70×70 PatchGANs [14] and produces a 1-dimensional output.

We train all the networks from scratch for 200 epochs using Adam optimizer [18]. The learning rate is held constant at 0.0002 for first 100 epochs and then lineary decay to zero for the last 100 epochs. During optimizing the discriminator, the objective is halved to ensure that the discriminator learns slowly relative to the generator.