TrojanZoo: Towards Unified, Holistic, and Practical Evaluation of Neural Backdoors

Ren Pang* Zheng Zhang* Xiangshan Gao† Zhaohan Xi*
Shouling Ji† Peng Cheng† Xiapu Luo‡ Ting Wang*
* *Pennsylvania State University, {rbp5354, zxz147, zxx5113, ting}@psu.edu*
† *Zhejiang University, {corazju, sji, lunar_heart}@zju.edu.cn*
‡ *Hong Kong Polytechnic University, csxluo@comp.polyu.edu.hk*

2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P) | 978-1-6654-1614-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/EUROSP53844.2022.00048

Abstract—**Neural backdoors represent one primary threat to the security of deep learning systems. The intensive research has produced a plethora of backdoor attacks/defenses, resulting in a constant arms race. However, due to the lack of evaluation benchmarks, many critical questions remain under-explored: (i) what are the strengths and limitations of different attacks/defenses? (ii) what are the best practices to operate them? and (iii) how can the existing attacks/defenses be further improved?**

To bridge this gap, we design and implement TROJANZOO, the first open-source platform for evaluating neural backdoor attacks/defenses in a unified, holistic, and practical manner. Thus far, focusing on the computer vision domain, it has incorporated 8 representative attacks, 14 state-of-the-art defenses, 6 attack performance metrics, 10 defense utility metrics, as well as rich tools for in-depth analysis of the attack-defense interactions. Leveraging TROJANZOO, we conduct a systematic study on the existing attacks/defenses, unveiling their complex design spectrum: both manifest intricate trade-offs among multiple desiderata (e.g., the effectiveness, evasiveness, and transferability of attacks). We further explore improving the existing attacks/defenses, leading to a number of interesting findings: (i) one-pixel triggers often suffice; (ii) training from scratch often outperforms perturbing benign models to craft trojan models; (iii) optimizing triggers and trojan models jointly greatly improves both attack effectiveness and evasiveness; (iv) individual defenses can often be evaded by adaptive attacks; and (v) exploiting model interpretability significantly improves defense robustness. We envision that TROJANZOO will serve as a valuable platform to facilitate future research on neural backdoors.

Index Terms—**backdoor attack, backdoor defense, benchmark platform, deep learning security**

1. Introduction

Today's deep learning (DL) systems are large, complex software artifacts. With the increasing system complexity and training cost, it becomes not only tempting but also necessary to exploit pre-trained deep neural networks (DNNs) in building DL systems. It was estimated that as of 2016, over 13.7% of DL-related repositories on GitHub re-use at least one pre-trained DNN [27]. On the upside, this "plug-and-play" paradigm greatly simplifies the development cycles [47]. On the downside, as most pre-trained DNNs are contributed by untrusted third parties [8], their lack of standardization or regulation entails profound security implications.

In particular, pre-trained DNNs can be exploited to launch *neural backdoor* attacks [21], [38], [43], one primary threat to the security of DL systems. In such attacks, a maliciously crafted DNN ("trojan model") forces its host system to misbehave once certain pre-defined conditions ("triggers") are met but to function normally otherwise, which can result in consequential damages in security-sensitive domains [7], [15], [61].

Motivated by this, intensive research has led to a plethora of attacks that craft trojan model via exploiting properties such as neural activation patterns [12], [21], [32], [38], [55], [68] and defenses that mitigate trojan models during inspection [11], [23], [26], [36], [37], [62] or detect trigger inputs at inference [10], [14], [19], [60]. With the rapid development of new attacks/defenses, a number of open questions have emerged: RQ$_1$ – *What are the strengths and limitations of different attacks/defenses?* RQ$_2$ – *What are the best practices (e.g., optimization strategies) to operate them?* RQ$_3$ – *How can the existing backdoor attacks/defenses be further improved?*

Despite their importance for understanding and mitigating the vulnerabilities incurred by neural backdoors, these questions are largely under-explored due to the following challenges.

Non-holistic evaluations – Most studies conduct the evaluation with a fairly limited set of attacks/defenses, resulting in incomplete assessment. For instance, it is unknown whether STRIP [19] is effective against the newer ABE attack [31]. Further, the evaluation often uses simple, macro-level metrics, failing to comprehensively characterize given attacks/defenses. For instance, most studies use attack success rate (*ASR*) and clean accuracy drop (*CAD*) to assess attack performance, which is insufficient to describe the attack's ability of trading between these two metrics.

Non-unified platforms – Due to the lack of unified benchmarks, different attacks/defenses are often evaluated under inconsistent settings, leading to non-comparable conclusions. For instance, TNN [38] and LB [68] are evaluated with distinct trigger definitions (*i.e.*, shape, size, and transparency), datasets, and DNNs, making it difficult to directly compare their assessment.

Non-adaptive attacks – The evaluation of the existing defense [19], [23], [36], [62] often assume static, non-adaptive attacks, without fully accounting for the adversary's possible countermeasures, which however is

© 2022, Ren Pang. Under license to IEEE.
DOI 10.1109/EuroSP53844.2022.00048

critical for modeling the adversary's optimal strategies and assessing the attack vulnerabilities in realistic settings.

Our Work

To this end, we design, implement, and evaluate TrojanZoo, an open-source platform for assessing neural backdoor attacks/defenses in a unified, holistic, and practical manner. Note that while it is extensible to other domains (*e.g.*, NLP), currently, TrojanZoo focuses on the image classification task in the computer vision domain. Our contributions are summarized in three major aspects:

Platform – To our best knowledge, TrojanZoo represents the first open-source platform specifically designed for evaluating neural backdoor attacks/defenses. At the moment of writing (02/06/2022), focusing on the computer vision domain, TrojanZoo has incorporated 8 representative attacks, 14 state-of-the-art defenses, 6 attack performance metrics, 10 defense utility metrics, as well as a benchmark suite of 5 DNN models, 5 downstream models, and 6 datasets. Further, TrojanZoo implements a rich set of tools for in-depth analysis of the attack-defense interactions, including measuring feature-space similarity, tracing neural activation patterns, and comparing attribution maps.

Measurement – Leveraging TrojanZoo, we conduct a systematic study of the existing attacks/defenses, unveiling the complex design spectrum for the adversary and the defender. Different attacks manifest delicate trade-offs among effectiveness, evasiveness, and transferability. For instance, weaker attacks (*i.e.*, lower *ASR*) tend to show higher transferability. Meanwhile, different defenses demonstrate trade-offs among robustness, utility-preservation, and detection accuracy. For instance, while effective against a variety of attacks, model sanitization [36], [40] also incur a significant accuracy drop. These observations indicate the importance of using comprehensive metrics to evaluate neural backdoor attacks/defenses, and suggest the optimal practices of applying them under given settings.

Exploration – We further explore improving existing attacks/defenses, leading to a number of previously unknown findings including (*i*) one-pixel triggers often suffice (over 95% *ASR*); (*ii*) training from scratch often outperforms perturbing benign models to forge trojan models; (*iii*) leveraging DNN architectures (*e.g.*, skip connects) in optimizing trojan models improves the attack effectiveness; (*iv*) most individual defenses are vulnerable to adaptive attacks; and (*v*) exploiting model interpretability significantly improves defense robustness. We envision that the TrojanZoo platform and our findings will facilitate future research on neural backdoors and shed light on designing and building DL systems in a more secure and informative manner.[1]

Roadmap

The remainder of the paper proceeds as follows. § 3 introduces fundamental concepts and assumptions; § 4 details the design and implementation of TrojanZoo and systemizes existing attacks/defenses; equipped with TrojanZoo, § 5 conducts a systematic evaluation of existing attacks/defenses; § 6 explores their further improvement; § 7 discusses the limitations of TrojanZoo and points to future directions; the paper is concluded in § 8.

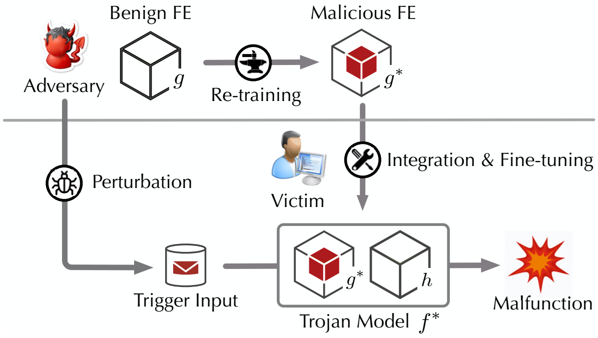


Figure 1: Illustration of neural backdoor attacks.

2. Related Work

Some recent studies have surveyed neural backdoor attacks/defenses (*e.g.*, [33]); yet, none of them provides benchmark implementation or empirical evaluation to explore their strengths/limitations. Compared with the rich collection of platforms for adversarial attacks/defenses (*e.g.*, CleverHans [2], DeepSec [35], and AdvBox [1]), only few platforms currently support evaluating neural backdoors. For instance, ART [3] integrates 3 attacks and 3 defenses.

In comparison, TrojanZoo differs in major aspects: (*i*) to our best knowledge, it features the most comprehensive library of attacks/defenses; (*ii*) it regards the evaluation metrics as a first-class citizen and implements 6 attack performance metrics and 10 defense utility metrics, which holistically assess given attacks/defenses; (*iii*) besides reference implementation, it also provides rich utility tools for in-depth analysis of attack-defense interactions, such as measuring feature-space similarity, tracing neural activation patterns, and comparing attribution maps. r The work closest to ours is perhaps TrojAI [4], which is a contest platform for model-inspection defenses against neural backdoors. While compared with TrojanZoo, TrojAI provides a much larger pool of trojan models (over 10K) across different modalities (*e.g.*, vision and NLP), TrojanZoo departs from TrojAI in majors aspects and offers its unique value. (*i*) Given its contest-like setting, TrojAI is a closed platform focusing on evaluating model-inspection defenses (*i.e.*, detecting trojan models) against fixed attacks, while TrojanZoo is an open platform that provides extensible datasets, models, attacks, and defenses. Thus, TrojanZoo may serve the needs ranging from conducting comparative studies of existing attacks/defenses to exploring and evaluating new attacks/defenses. (*ii*) While TrojAI focuses on model-inspection defenses, TrojanZoo integrates four major defense categories. (*iii*) In TrojAI, for its purpose, the concrete attacks behind the trojan models are unknown, which makes it challenging to assess the strengths/limitations of given defenses with respect to different attacks, while in TrojanZoo one may directly evaluate such interactions. (*iv*) As the attacks are fixed in TrojAI, one may not

[1] All the data, models, and code used in the paper are released at: https://github.com/ain-soph/trojanzoo.

evaluate adaptive attacks. (*v*) The main metric used in TrojAI is the accuracy that defenses successfully detect trojan models, while TrojanZoo provides a much richer set of metrics to characterize attacks/defenses.

3. Fundamentals

We first introduce fundamental concepts and assumptions used throughout the paper. The important notations are summarized in Table 1.

Notation	Definition
$\mathcal{A}, \mathcal{D}$	attack, defense
x, x^*	clean input, trigger input
x_i	i-th dimension of x
r	trigger
m	mask (α for each pixel)
f, f^*	benign model, trojan model
f_{feat}	upstream feature extractor
g, g^*	downstream classifier, surrogate classifier
t	adversary's target class
$\mathcal{T}$	reference set
$\mathcal{R}_\epsilon, \mathcal{F}_\delta$	trigger, model feasible sets

Table 1. Symbols and notations.

3.1. Preliminaries

Deep neural networks (DNNs) – Deep neural networks (DNNs) represent a class of ML models to learn high-level abstractions of complex data. We assume a predictive setting, in which a DNN f_θ (parameterized by θ) encodes a function $f_\theta : \mathbb{R}^n \to \mathbb{S}^m$, where n and m denote the input dimensionality and the number of classes. Given input x, $f(x)$ is a probability vector (simplex) over m classes.

Pre-trained DNNs – Today, it becomes not only tempting but also necessary to reuse pre-trained models in domains in which data labeling or model training is expensive [70]. Under the transfer learning setting, as shown in Figure 1, the feature extractor (FE) g of a pre-trained model is often reused and composed with a classifier h to form an end-to-end model f. As the data used to train g may differ from the downstream task, it is often necessary to fine-tune $f = h \circ g$ in a supervised manner. One may opt to perform full-tuning to train both g and h or partial-tuning to train h only with g fixed [27].

Neural backdoor attacks – With the increasing use of DNN models in security-sensitive domains, the adversary is strongly incentivized to forge malicious FEs as attack vectors and lure victim users to re-use them during system development [21]. Specifically, through a malicious FE, the backdoor attack infects the target model with malicious functions desired by the adversary, which are activated once pre-defined conditions ("triggers") are present. We refer to such infected models as "trojan models". Typically, a trojan model reacts to trigger-embedded inputs (*e.g.*, images with specific watermarks) in a highly predictable manner (*e.g.*, misclassified to a target class) but functions normally otherwise.

3.2. Specifics

Trigger mixing operator – For given trigger r, the operator $\oplus$ mixes a clean input $x \in \mathbb{R}^n$ with r to generate a trigger input $x \oplus r$. Typically, r comprises three parts: (*i*) mask $m \in \{0,1\}^n$ specifies where r is applied (*i.e.*, x's i-th feature x_i is retained if m_i is on and mixed with r otherwise); (*ii*) transparency $\alpha \in [0,1]$ specifies the mixing weight; and (*iii*) pattern $p(x) \in \mathbb{R}^n$ specifies r's color intensity, which can be a constant, randomly drawn from a distribution (*e.g.*, by perturbing a template), or dependent on x [45]. Formally, the trigger embedding operator is defined as:

$$x \oplus r = (1 - m) \odot [(1 - \alpha)x + \alpha p(x)] + m \odot x \qquad (1)$$

where $\odot$ denotes element-wise multiplication.

Attack objectives – The trojan model satisfies that with high probability, (*i*) trigger inputs are classified to the target class desired by the adversary and (*ii*) clean input are still correctly classified. Formally, the adversary forges the malicious FE by optimizing the following objective:

$$\min_{r \in \mathcal{R}, \theta} \mathbb{E}_{(x,y)\in\mathcal{T}} \left[\ell(f_\theta(x \oplus r), t) + \lambda \ell(f_\theta(x), y)\right] \qquad (2)$$

where $\mathcal{T}$ represents the training set, t denotes the target class, and trigger r is selected from the feasible set $\mathcal{R}$ (which constrains r's shape, transparency, and/or pattern). Intuitively, the first and second terms describe (*i*) and (*ii*), respectively, and the hyper-parameter λ balances the two objectives.

Adversary's knowledge – If the downstream classifier h is known to the adversary, f shares the same architecture with the model $h \circ g$ used by the victim; otherwise, the adversary may resort to a surrogate classifier h^* (*i.e.*, $h^* \circ g$) or re-define the loss $\ell(f(x \oplus r), t)$ in terms of latent representations [43], [68] as $\Delta(g(x \oplus r), \phi_t)$, that is, the difference(*e.g.*, MSE loss) between $g(x \oplus r)$ and ϕ_t, where ϕ_t is the average latent representation of class t.

Malicious FE training – To optimize Eqn. 2, one may perturb a benign FE [38], [55] or train the malicious FE from scratch (details in § 6). To satisfy the trigger constraint, r can be fixed [21], partially defined [38] (*e.g.*, with its mask fixed), or optimized with f jointly [43].

4. Platform

As illustrated in Figure 2, TrojanZoo comprises three major components: (*i*) the attack library integrates a set of representative attacks that, for given benign models and clean inputs, are able to generate trojan models and trigger inputs; (*ii*) the defense library integrates a set of state-of-the-art defenses that are able to provide model- and input-level protection against trojan models and trigger inputs; and (*iii*) the analysis engine, equipped with attack performance metrics, defense utility metrics, and feature-rich utility tools, is able to conduct unified and holistic evaluation across different attacks/defenses.

In its current implementation, TrojanZoo has incorporated 8 attacks, 14 defenses, 6 attack performance metrics, and 10 defense utility metrics, which we systematize as follows.

4.1. Attack Library

While neural backdoor attacks can be characterized from a number of aspects, here we focus on 4 key design choices by the adversary that directly impact attack

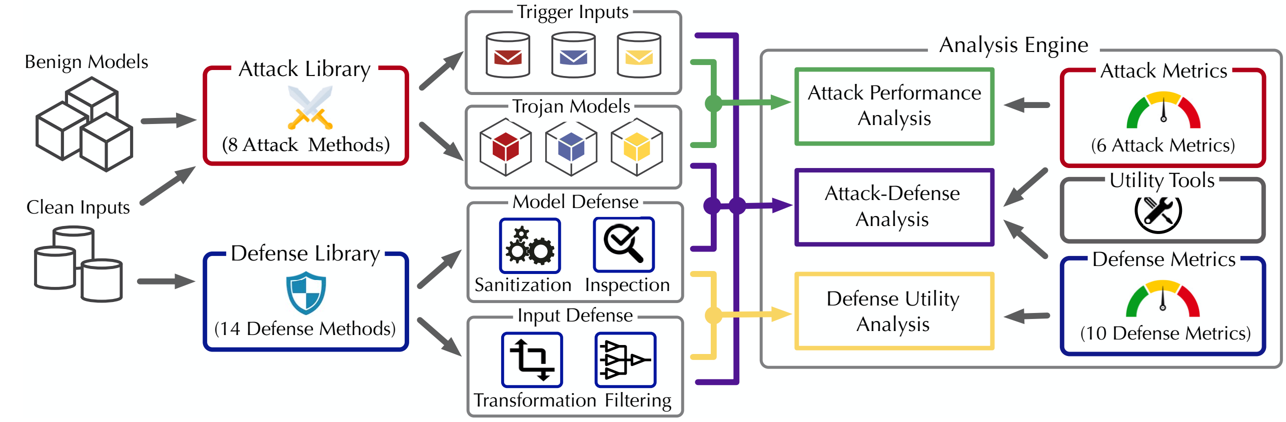


Figure 2: Overall system design of TrojanZoo.

performance. Table 2 summarizes the representative neural backdoor attacks currently implemented in TrojanZoo, which are characterized along the above 4 dimensions. More specifically,

Attack	Architecture Modifiability	Trigger Optimizability	Fine-tuning Survivability	Defense Adaptivity
BN [21]	○	○	○	○
ESB [57]	●	○	○	○
TNN [38]	○	◐	○	○
RB [39]	○	◐	○	○
TB [12]	○	◐	○	○
LB [68]	○	○	●	○
ABE [31]	○	○	○	●
IMC [43]	○	●	●	●

Table 2. Summary of representative neural backdoor attacks currently implemented in TrojanZoo (● – full optimization, ◐ – partial optimization, ○ – no optimization)

Non-optimization – The attack simply solves Eqn. 2 under pre-defined triggers (*i.e.*, shape, transparency, and pattern) without optimization for other desiderata.

– BadNet (BN) [21], as the representative, pre-defines trigger r, generates trigger inputs $\{(x \oplus r, t)\}$, and crafts the trojan model f^* by re-training a benign model f with such data.

Architecture modifiability – whether the attack is able to change the DNN architecture. Being allowed to modify both the architecture and the parameters enables a larger attack spectrum, but also renders the trojan model more susceptible to certain defenses (*e.g.*, model specification checking).

– Embarrassingly-Simple-Backdoor (ESB) [57], as the representative, modifies f's architecture by adding a module which overwrites the prediction as t if r is recognized. Without disturbing f's original configuration, f^* retains f's predictive power on clean inputs.

Trigger optimizability – whether the attack uses a fixed, pre-defined trigger or optimizes it during crafting the trojan model. Trigger optimization often leads to stronger attacks with respect to given desiderata (*e.g.*, trigger stealthiness).

– TrojanNN (TNN) [38] fixes r's shape and position, optimizes its pattern to activate neurons rarely activated by clean inputs in pre-processing, and then forges f^* by re-training f in a manner similar to BN.

– Reflection-Backdoor (RB) [39] optimizes trigger stealthiness by defining r as the physical reflection of a clean image x^r (selected from a pool): $r = x^r \otimes k$, where k is a convolution kernel, and $\otimes$ is the convolution operator.

– Targeted-Backdoor (TB) [12] randomly generates r's position in training, which makes f^* effective regardless of r's position and allows the adversary to optimize r's stealthiness by placing it at the most plausible position (*e.g.*, an eyewear watermark over eyes).

Fine-tuning survivability – whether the backdoor remains effective if the model is fine-tuned. A pre-trained model is often composed with a classifier and fine-tuned using the data from the downstream task. It is desirable to ensure that the backdoor remains effective after fine-tuning.

– Latent Backdoor (LB) [68] accounts for the impact of downstream fine-tuning by optimizing g with respect to latent representations rather than final predictions. Specifically, it instantiates Eqn. 2 with the following loss function: $\ell(g(x \oplus r), t) = \Delta(g(x \oplus r), \phi_t)$, where Δ measures the difference of two latent representations and ϕ_t denotes the average representation of class t, defined as $\phi_t = \arg\min_\phi \mathbb{E}_{(x,t)\in\mathcal{T}}[g(x)]$.

Defense adaptivity – whether the attack is optimizable to evade possible defenses. For the attack to be effective, it is essential to optimize the evasiveness of the trojan model and the trigger input with respect to the deployed defenses.

– Adversarial-Backdoor-Embedding (ABE) [31] accounts for possible defenses in forging g^*. In solving Eqn. 2, ABE also optimizes the indistinguishability of the latent representations of trigger and clean inputs. Specifically, it uses a discriminative network d to predict the representation of a given input x as trigger or clean. Formally, the loss is defined as $\Delta(d \circ g(x), b(x))$, where $b(x)$ encodes whether x is trigger or clean, while g^* and d are trained using an adversarial learning framework [20].

Multi-optimization – whether the attack is optimizable with respect to multiple objectives listed above.

– Input-Model Co-optimization (IMC) [43] is motivated by the mutual-reinforcement effect between r and f^*: optimizing one amplifies the effectiveness of the other. Instead of solving Eqn. 2 by first pre-defining r and then optimizing f^*, IMC optimizes r and f^* jointly, which enlarges the search spaces for r and f^*, leading to attacks satisfying multiple desiderata (*e.g.*, fine-tuning survivability and defense adaptivity).

Neural Backdoor Defense	Category	Mitigation		Detection Target			Design Rationale
		Input	Model	Input	Model	Trigger	
Randomized-Smoothing (Rs) [14]	Input Reformation	✓					$\mathcal{A}$'s fidelity (x's and x^*'s surrounding class boundaries)
Down-Upsampling (Du) [66]		✓					$\mathcal{A}$'s fidelity (x's and x^*'s high-level features)
Manifold-Projection (Mp) [41]		✓					$\mathcal{A}$'s fidelity (x's and x^*'s manifold projections)
Activation-Clustering (Ac) [10]	Input Filtering			✓			distinct activation patterns of $\{x\}$ and $\{x^*\}$
Spectral-Signature (Ss) [59]				✓			distinct activation patterns of $\{x\}$ and $\{x^*\}$ (spectral space)
STRIP (Strip) [19]				✓			distinct self-entropy of x's and x^*'s mixtures with clean inputs
NEO (Neo) [60]				✓			sensitivity of f^*'s prediction to trigger perturbation
Adversarial-Retraining (Ar) [40]	Model Sanitization		✓				$\mathcal{A}$'s fidelity (x's and x^*'s surrounding class boundaries)
Fine-Pruning (Fp) [36]			✓				$\mathcal{A}$'s use of neurons rarely activated by clean inputs
NeuralCleanse (Nc) [62]	Model Inpsection				✓	✓	abnormally small perturbation from other classes to t in f
DeepInspect (Di) [11]					✓	✓	abnormally small perturbation from other classes to t in f^*
TABOR (Tabor) [23]					✓	✓	abnormally small perturbation from other classes to t in f
NeuronInspect (Ni) [26]					✓		distinct explanations of f and f^* with respect to clean inputs
ABS (Abs) [37]					✓	✓	$\mathcal{A}$'s use of neurons elevating t's prediction

Table 3. Summary of representative neural backdoor defenses currently implemented in TrojanZoo ($\mathcal{A}$ – backdoor attack, x – clean input, x^* – trigger input, f – benign model, f^* – trojan model, t – target class)

4.2. Attack Performance Metrics

Currently, TrojanZoo incorporates 6 metrics to assess the effectiveness, evasiveness, and transferability of given attacks.

Attack success rate (*ASR*) – which measures the likelihood that trigger inputs are classified to the target class t:

$$\text{Attack Success Rate } (\textit{ASR}) = \frac{\text{\# successful trials}}{\text{\# total trials}} \tag{3}$$

Typically, higher *ASR* indicates more effective attacks.

Trojan misclassification confidence (*TMC*) – which is the average confidence score assigned to class t of trigger inputs in successful attacks. Intuitively, *TMC* complements *ASR* and measures attack efficacy from another perspective. For two attacks with the same *ASR*, we consider the one with higher *TMC* a stronger one.

Clean accuracy drop (*CAD*) – which measures the difference of the classification accuracy of benign and trojan models; *CAD* measures whether the attack directs its influence to trigger inputs only.

Clean classification confidence (*CCC*) – which is the average confidence assigned to the ground-truth classes of clean inputs; *CCC* complements *CAD* by measuring attack specificity from the perspective of classification confidence.

Efficacy-specificity AUC (*AUC*) – which quantifies the aggregated trade-off between attack efficacy (measured by *ASR*) and attack specificity (measured by *CAD*). As revealed in [43], there exists an intricate balance: at a proper cost of specificity, it is possible to significantly improve efficacy, and vice versa; *AUC* measures the area under the *ASR*-*CAD* curve. Intuitively, smaller *AUC* implies a more significant trade-off effect.

Neuron-separation ratio (*NSR*) – which measures the intersection between neurons activated by clean and trigger inputs. In the penultimate layer of the model, we find $\mathcal{N}_\mathrm{c}$ and $\mathcal{N}_\mathrm{t}$, the top-k active neurons with respect to clean and trigger inputs, respectively, and calculate their jaccard index:

$$\text{Neuron Separation Ratio } (\textit{NSR}) = 1 - \frac{|\mathcal{N}_\mathrm{t} \cap \mathcal{N}_\mathrm{c}|}{|\mathcal{N}_\mathrm{t} \cup \mathcal{N}_\mathrm{c}|} \tag{4}$$

Intuitively, *NSR* compares the neural activation patterns of clean and trigger inputs.

4.3. Defense Library

The existing defenses against neural backdoors, according to their strategies, can be classified into 4 major categories, as summarized in Table 3. Notably, we focus on the setting of transfer learning or outsourced training, which precludes certain other defenses such as purging poisoning training data [53]. Next, we detail the 14 representative defenses currently implemented in TrojanZoo.

Input reformation – which, before feeding an incoming input to the model, first reforms it to mitigate the influence of the potential trigger, yet without explicitly detecting whether it is a trigger input. It typically exploits the high fidelity of attack $\mathcal{A}$, that is, $\mathcal{A}$ tends to retain the perceptual similarity of a clean input x and its trigger counterpart x^*.

– Randomized-Smoothing (Rs) [14] exploits the premise that $\mathcal{A}$ retains the similarity of x and x^* in terms of their surrounding class boundaries and classifies an input by averaging the predictions within its vicinity (via adding Gaussian noise).

– Down-Upsampling (Du) [66] exploits the premise that $\mathcal{A}$ retains the similarity of x and x^* in terms of their high-level features while the trigger r is typically not perturbation-tolerant. By downsampling and then upsampling x^*, it is possible to mitigate r's influence.

– Manifold-Projection (Mp) [41] exploits the premise that $\mathcal{A}$ retains the similarity of x and x^* in terms of their projections to the data manifold. Thus, it trains an autoencoder to learn an approximate manifold, which projects x^* to the manifold.

Input filtering – which detects whether an incoming input is embedded with a trigger and possibly recovers the clean input. It typically distinguishes clean and trigger inputs using their distinct characteristics.

– Activation-Clustering (Ac) [10] distinguishes clean and trigger inputs by clustering their latent representations. While Ac is also applicable for purging poisoning data, we consider its use as an input filtering method at inference time.

– Spectral-Signature (SS) [59] exploits the similar property in the spectral space.

– STRIP [19] mixes a given input with a clean input and measures the self-entropy of its prediction. If the input is trigger-embedded, the mixture remains dominated by the trigger and tends to be misclassified, resulting in low self-entropy.

– NEO [60] detects a trigger input by searching for a position, if replaced by a "blocker", changes its prediction, and uses this substitution to recover its original prediction.

Model sanitization – which, before using a pre-trained model f, sanitizes it to mitigate the potential backdoor, yet without explicitly detecting whether f is tampered.

– Adversarial-Retraining (AR) [40] treats trigger inputs as one type of adversarial inputs and applies adversarial training over the pre-trained model to improves its robustness to backdoor attacks.

– Fine-Pruning (FP) [36] uses the property that the attack exploits spare model capacity. It thus prunes rarely used neurons and then applies fine-tuning to defend against pruning-aware attacks.

Model inspection – which determines whether f is a trojan model and, if so, recovers the target class and the potential trigger, at the model checking stage.

– NeuralCleanse (NC) [62] searches for potential triggers in each class t. If t is trigger-embedded, the minimum perturbation required to change the predictions of the inputs in other classes to t is abnormally small.

– DeepInspect (DI) [11] follows a similar pipeline but uses a generative network to generate trigger candidates.

– TABOR [23] extends NC by adding a new regularizer to control the trigger search space.

– NeuronInspect (NI) [26] exploits the property that the explanation heatmaps of benign and trojan models manifest distinct characteristics. Using the features extracted from such heatmaps, NI detects trojan models as outliers.

– ABS [37] inspects f to sift out abnormal neurons with large elevation difference (*i.e.*, active only with respect to one specific class) and identifies triggers by maximizing abnormal neuron activation while preserving normal neuron behaviors.

4.4. Defense Utility Metrics

Currently, TROJANZOO incorporates 10 metrics to evaluate the robustness, utility-preservation, and genericity of given defenses. The metrics are tailored to the objectives of each defense category (*e.g.*, trigger input detection). For ease of exposition, below we consider the performance of a given defense $\mathcal{D}$ with respect to a given attack $\mathcal{A}$.

Attack rate deduction (*ARD*) – which measures the difference of $\mathcal{A}$'s *ASR* before and after $\mathcal{D}$. Intuitively, *ARD* indicates $\mathcal{D}$'s impact on $\mathcal{A}$'s efficacy. Intuitively, larger *ARD* indicates more effective defense. We also use $\mathcal{A}$'s *TMC* to measure $\mathcal{D}$'s influence on the classification confidence of trigger inputs.

Clean accuracy drop (*CAD*) – which measures the difference of the *ACC* of clean inputs before and after $\mathcal{D}$ is applied. It measures $\mathcal{D}$'s impact on clean inputs. Note that *CAD* here is defined differently from its counterpart in attack performance metrics. We also use *CCC* to measure $\mathcal{D}$'s influence on the classification confidence of clean inputs.

True positive rate (*TPR*) – which, for input-filtering methods, measures the performance of detecting trigger inputs.

$$\text{True Positive Rate } (\textit{TPR}) = \frac{\text{\# detected trigger inputs}}{\text{\# total trigger inputs}} \quad (5)$$

Correspondingly, we use false positive rate (*FPR*) to measure the error of misclassifying clean inputs as trigger inputs.

Anomaly index value (*AIV*) – which measures the anomaly of trojan models in model-inspection defenses. Most existing methods (*e.g.*, [11], [23], [37], [62]) formalize finding trojan models as outlier detection: each class t is associated with a score (*e.g.*, minimum perturbation); if its score significantly deviates from others, t is considered to contain a backdoor. *AIV*, the absolute deviations from median normalized by median absolute deviation (*MAD*), provide a reliable measure for such dispersion. Typically, t with *AIV* larger than 2 has over 95% probability of being anomaly.

Mask L_1 norm (*MLN*) – which measures the ℓ_1-norm of the triggers recovered by model-inspection methods.

Mask jaccard similarity (*MJS*) – which further measures the intersection between the recovered trigger and the ground-truth trigger (injected by the adversary). Let m^o and m^r be the masks of original and recovered triggers. We define *MJS* as the Jaccard similarity of m^o and m^r :

$$\text{Mask Jaccard Similarity } (\textit{MJS}) = \frac{|O(m^o) \cap O(m^r)|}{|O(m^o) \cup O(m^r)|} \quad (6)$$

where $O(m)$ denotes the set of non-zero elements in m.

Average running time (*ART*) – which measures $\mathcal{D}$'s overhead. For model sanitization or inspection, which is performed offline, *ART* is measured as the running time per model; while for input filtering or reformation, which is executed online, *ART* is measured as the execution time per input.

5. Assessment

Leveraging TROJANZOO, we conduct a systematic assessment of the existing attacks and defenses and unveil their complex design spectrum: both attacks and defenses tend to manifest intricate trade-offs among multiple desiderata. We begin by describing the setting of the evaluation.

5.1. Experimental Setting

Datasets – In the evaluation, we primarily use 5 datasets: CIFAR10 [29], CIFAR100 [29], ImageNet [16], GTSRB [52], and VGGFace2 [9], with their statistics summarized in Table 4.

Models – We consider 3 representative DNN models: VGG [51], ResNet [24], and DenseNet [25]. Using models of distinct architectures (*e.g.*, residual blocks versus skip connections), we factor out the influence of individual model characteristics. By default, we assume

Dataset	# Class	# Dimension	Model	ACC
CIFAR10	10	32×32	ResNet18	95.37%
			DenseNet121	93.84%
			VGG13	92.44%
CIFAR100	100	32×32	ResNet18	73.97%
GTSRB	43	32×32		98.18%
ImageNet	10	224×224		92.40%
VGGFace2	20	224×224		90.77%

Table 4. *ACC* of benign models over different datasets.

the downstream classifier comprising one fully-connected layer with softmax activation (1FCN). We also consider other types of classifiers, including Bayes, SVM, and Random Forest. The *ACC* of benign models is summarized in Table 4.

Attacks, Defenses, and Metrics – In the evaluation, we exemplify with 8 attacks in Table 2 and 12 defenses in Table 3, and measure them using all the metrics in § 4.2 and § 4.4. In all the experiments, we generate 10 trojan models for a given attack under each setting and 100 pairs of clean-trigger inputs with respect to each trojan model. The reported results are averaged over these cases.

Implementation – All the models, algorithms, and measurements are implemented in PyTorch. The default parameter setting is summarized in Table 20 and 21 (§ A).

5.2. Attack Evaluation

We evaluate the existing attacks under the vanilla setting (without defenses), aiming to understand the impact of various design choices on the attack performance. Due to space limitations, we mainly report the results on CIFAR10 and defer the results on other datasets to § B. Overall, different attacks manifest intricate trade-offs among *effectiveness*, *evasiveness*, and *transferability*, as detailed below.

5.2.1. Effectiveness vs. Evasiveness (Trigger) . We start with the effectiveness-evasiveness trade-off. Intuitively, the effectiveness measures whether the trigger inputs are successfully misclassified into the target class, while the evasiveness measures whether the trigger inputs and trojan models are distinguishable from their normal counterparts. Here, we first consider the evasiveness of triggers.

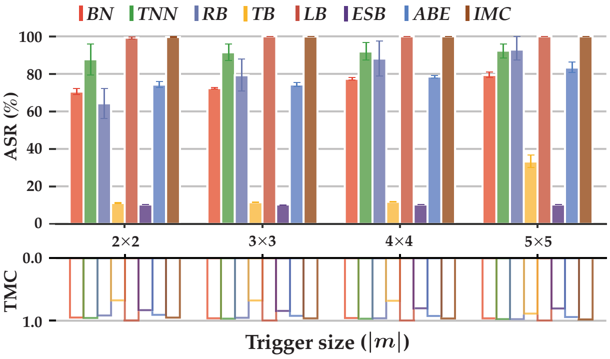


Figure 3: *ASR* and *TMC* with respect to trigger size ($\alpha = 0.8$).

Trigger size – Recall that the trigger definition comprises mask m, transparency α, and pattern p. We measure how the attack effectiveness varies with the trigger size $|m|$. To make fair comparison, we bound the clean accuracy drop (*CAD*) of all the attacks below 3% via controlling the number of optimization iterations n_{iter}. Figure 3 plots the attack success rate (*ASR*) and trojan misclassification confidence (*TMC*) of various attacks under varying $|m|$ (with fixed $\alpha = 0.8$).

Observe that most attacks seem insensitive to $|m|$: as $|m|$ varies from 2×2 to 5×5, the *ASR* of most attacks increases by less than 10%, except RB with over 30% growth. This may be attributed to its additional constraints: RB defines the trigger to be the reflection of another image; thus, increasing $|m|$ may improve its perturbation spaces. Compared with other attacks, TB and ESB perform poorly because TB aims to force inputs with random triggers to be misclassified while ESB is unable to account for trigger transparency during training. Also observe that the *TMC* of most attacks remains close to 1.0 regardless of $|m|$.

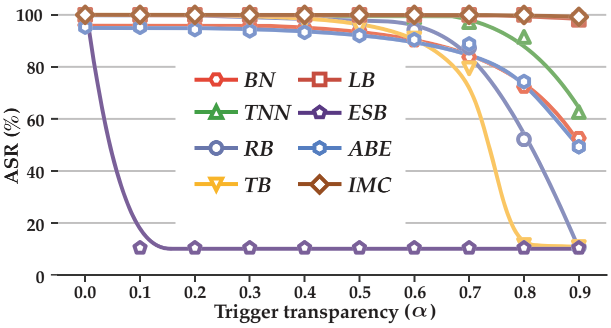


Figure 4: *ASR* with respect to trigger transparency ($|m|$ = 3×3).

Trigger transparency – Under the same setting, we evaluate the impact of trigger transparency α. Figure 4 plots the *ASR* of various attacks as a function of α ($|m| =$ 3×3).

Compared with trigger size, α has a more profound impact. The *ASR* of most attacks drops sharply once α exceeds 0.6, among which TB approaches 10% if $\alpha \geq 0.8$, and ESB works only if α is close to 0, due to its reliance on recognizing the trigger precisely to overwrite the model prediction. Meanwhile, LB and IMC seem insensitive to α. This may be attributed to that LB optimizes trojan models with respect to latent representations (rather than final predictions), while IMC optimizes trigger patterns and trojan models jointly. Both strategies may mitigate α's impact.

Attack	CIFAR10	CIFAR100	ImageNet	
	$\|m\| = 3,\ \alpha = 0.8$	$\|m\| = 3,\ \alpha = 0.8$	$\|m\| = 3,\ \alpha = 0$	$\|m\| = 7,\ \alpha = 0.8$
BN	72.4 (0.96)	64.5 (0.96)	90.0 (0.98)	11.4 (0.56)
TNN	91.5 (0.97)	89.8 (0.98)	95.2 (0.99)	11.6 (0.62)
RB	52.1 (1.0)	42.8 (0.95)	94.6 (0.98)	11.2 (0.59)
TB	11.5 (0.66)	23.4 (0.75)	82.8 (0.97)	11.4 (0.58)
LB	100.0 (1.0)	97.8 (0.99)	97.4 (0.99)	11.4 (0.59)
ESB	10.3 (0.43)	1.0 (0.72)	100.0 (0.50)	N/A
ABE	74.3 (0.91)	67.9 (0.96)	82.6 (0.97)	12.00 (0.50)
IMC	100.0 (1.0)	98.8 (0.99)	98.4 (1.0)	96.6 (0.99)

Table 5. Impact of data complexity on *ASR* and *TMC*.

Data complexity – The trade-off between attack effectiveness and trigger evasiveness is especially evident for complex data. We compare the *ASR* and *TMC* of given attacks on different datasets, with results in Table 5 (more in Table 22).

We observe that the class-space size (the number of classes) negatively affects the attack effectiveness. For example, the *ASR* of Bn drops by 7.9% from CIFAR10 to CIFAR100. Intuitively, it is more difficult to force trigger inputs from all the classes to be misclassified in larger output space. Moreover, it tends to require more significant triggers to achieve comparable attack performance on more complex data. For instance, for Imc to attain similar *ASR* on CIFAR10 and ImageNet, it needs to either increase trigger size (from 3×3 to 7×7) or reduce trigger transparency (from 0.8 to 0.0).

> **Remark 1** – *There exists a trade-off between attack effectiveness and trigger evasiveness (in terms of transparency), which is especially evident for complex data.*

5.2.2. Effectiveness vs. Evasiveness (Model) . Further, we consider the evasiveness of trojan models, which is measured by their difference from benign models in terms of classifying clean inputs. One intriguing property of the attacks is the trade-off between maximizing the attack effectiveness with respect to trigger inputs and minimizing the influence over clean inputs. Here, we characterize this trade-off via varying the fraction of trigger inputs in the training data. For each attack, we bound its *CAD* within 3%, measure its highest and lowest *ASR* (which corresponds to its lowest and highest *CAD* respectively), and then normalize the *ASR* and *CAD* measures to $[0, 1]$.

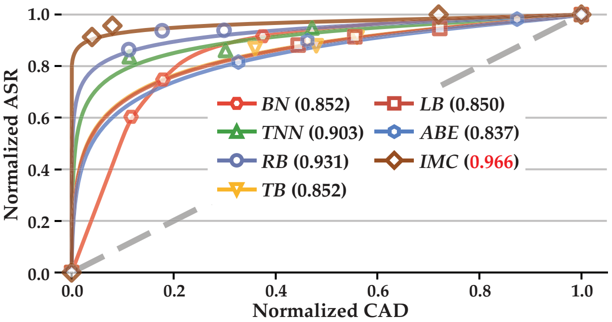


Figure 5: Trade-off between attack effectiveness and model evasiveness ($|m| = 3 \times 3$, $\alpha = 0.8$).

Figure 5 visualizes the normalized *CAD*-*ASR* trade-off. Observe that the curves of all the attacks manifest strong convexity, indicating the "leverage" effects [43]: it is practical to greatly improve *ASR* at a disproportionally small cost of *CAD*. Also, observe that different attacks feature varying Area Under the Curve (*AUC*). Intuitively, a smaller *AUC* implies a stronger leverage effect. Among all the attacks, Imc shows the smallest *AUC*. This may be explained by that Imc uses the trigger-model co-optimization framework, which allows the adversary to maximally optimize *ASR* at given *CAD*.

> **Remark 2** – *The trade-off between attack effectiveness and model evasiveness demonstrates strong "leverage" effects.*

5.2.3. Effectiveness vs. Transferability. Next, we evaluate the transferability of different attacks to the downstream tasks. We consider two scenarios: (*i*) the pre-training and downstream tasks share the same dataset; and (*ii*) the downstream task uses a different dataset.

Transferability (classifier) – In (*i*), we focus on evaluating the impact of downstream-classifier selection and fine-tuning strategy on the attacks. We consider 5 different classifiers (1/2 fully-connected layer, Bayes, SVM, and Random Forest) and 3 fine-tuning strategies (none, partial tuning, and full tuning). Notably, the adversary is unaware of such settings.

Attack	Fine-Tuning			Downstream Classifier			
	None	Partial	Full	2-FCN	Bayes	SVM	RF
Bn	72.4	72.3	30.4	72.2	73.5	64.7	66.0
Tnn	91.5	89.6	27.1	90.8	90.3	82.9	81.1
Rb	79.2	77.0	12.4	78.3	76.8	61.5	63.7
Lb	100.0	100.0	95.3	99.9	99.9	99.9	99.8
Imc	100.0	99.9	88.7	99.9	100.0	99.9	99.8

Table 6. Impact of fine-tuning and downstream-model selection.

Table 6 compares the *ASR* of 5 attacks with respect to varying downstream classifiers and fine-tuning strategies. Observe that fine-tuning has a large impact on attack effectiveness. For instance, the *ASR* of Tnn drops by 62.5% from partial- to full-tuning. Yet, Lb and Imc are less sensitive to fine-tuning, due to their optimization strategies. Also, note that the attack performance seems agnostic to the downstream classifier. This may be explained by that the downstream classifier in practice tends to manifest "pseudo-linearity" [27] (details in § A).

Transferability (data) – In (*ii*), we focus on evaluating the transferability of the attacks across different datasets.

Transfer Setting	Attack				
	Bn	Tnn	Rb	Lb	Imc
C → C	94.5 (0.99)	100.0 (1.0)	100.0 (1.0)	100.0 (1.0)	100.0 (1.0)
C → I	8.4 (0.29)	7.8 (0.29)	8.6 (0.30)	8.2 (0.30)	9.4 (0.32)
I → I	90.0 (0.98)	95.2 (0.99)	94.6 (0.98)	97.4 (0.99)	98.4 (1.0)
I → C	77.0 (0.84))	26.9 (0.72)	11.0 (0.38)	10.0 (0.38)	14.3 (0.48)

Table 7. *ASR* and *TMC* of transfer attacks across CIFAR10 (C) and ImageNet (I) ($|m| = 3 \times 3$, $\alpha = 0.0$).

We evaluate the effectiveness of transferring attacks across two datasets, CIFAR10 and ImageNet, with results summarized in Table 7. We have the following findings. Several attacks (*e.g.*, Bn) are able to transfer from ImageNet to CIFAR10 to a certain extent, but most attacks fail to transfer from CIFAR10 to ImageNet. The finding may be justified as follows. A model pre-trained on complex data (*i.e.*, ImageNet) tends to maintain its effectiveness of feature extraction on simple data (*i.e.*, CIFAR10) [17]; as a side effect, it may also preserve its effectiveness of propagating trigger patterns. Meanwhile, a model pre-trained on simple data may not generalize well to complex data. Moreover, compared with stronger attacks in non-transfer cases (*e.g.*, Lb), Bn shows much higher transferability. This may be explained by that to maximize the attack efficacy, the trigger and trojan model often need to "over-fit" the training data, resulting in poor transferability.

> **Remark 3** – *Most attacks transfer across classifiers; however, weaker attacks demonstrate higher transferability across datasets.*

5.3. Defense Evaluation

As the defenses from different categories bear distinct objectives (*e.g.*, detecting trigger inputs versus cleansing trojan models), below we evaluate each defense category separately.

Defense	Attack							
	BN	TNN	RB	TB	LB	ESB	ABE	IMC
–	93.3 (0.99)	99.9 (1.0)	99.8 (1.0)	96.7 (0.99)	100.0 (1.0)	100.0 (0.86)	95.3 (0.99)	100.0 (1.0)
RS	-0.5 (0.99) (±0.2)	-0.0 (1.0) (±0.0)	-0.0 -(1.0) (±0.0)	-0.3 (0.99) (±0.1)	-0.0 (1.0) (±0.0)	-89.1 (0.86) (±7.3)	-0.5 (0.99) (±0.1)	-0.0 (1.0) (±0.0)
DU	-2.2 (0.99) (±0.7)	-0.4 (1.0) (±0.1)	-5.4 (1.0) (±1.4)	-67.8 (1.0) (±12.8)	-4.1 (1.0) (±1.4)	-89.9 (0.86) (±22.7)	-0.5 (0.99) (±0.3)	-0.2 (1.0) (±0.0)
MP	-6.0 (0.99) (±2.1)	-37.4 (1.0) (±5.5)	-78.6 (1.0) (±14.2)	-11.0 (0.99) (±4.1)	-42.6 (1.0) (±1.5)	-87.8 (0.86) (±6.6)	-4.6 (0.99) (±0.4)	-16.0 (1.0) (±2.3)
FP	-82.9 (0.60) (±1.8)	-86.5 (0.64) (±4.3)	-89.1 (0.73) (±2.6)	-38.0 (0.89) (±6.1)	-27.6 (0.82) (±3.7)	-100.0 (0.81) (±0.0)	-84.5 (0.64) (±9.3)	-26.9 (0.83) (±4.6)
AR	-83.2 (0.84) (±2.2)	-89.6 (0.85) (±1.9)	-89.8 (0.62) (±0.7)	-86.2 (0.63) (±4.5)	-90.1 (0.83) (±2.8)	-100.0 (0.86) (±0.0)	-85.3 (0.81) (±4.4)	-89.7 (0.83) (±1.8)

Table 8. *ARD* and *TMC* of attack-agnostic defenses against various attacks (±: standard deviation).

5.3.1. Robustness vs. Utility.

As input transformation and model sanitization mitigate backdoors in an attack-agnostic manner, while input filtering and model inspection have no direct influence on clean accuracy, we focus on evaluating attack-agnostic defenses to study the trade-off between robustness and utility preservation.

Robustness – With the no-defense (vanilla) case as reference, we compare different defences in terms of attack rate deduction (*ARD*) and trojan misclassification confidence (*TMC*), with results shown in Table 8. We have the following observations: (*i*) MP and AR are the most robust methods in the categories of input transformation and model sanitization, respectively. (*ii*) FP seems robust against most attacks except LB and IMC, which is explained as follows: unlike attacks (*e.g.*, TNN) that optimize the trigger with respect to selected neurons, LB and IMC perform optimization with respect to all the neurons, making them immune to the pruning of FP. (*iii*) Most defenses are able to defend against ESB (over 85% *ARD*), which is attributed to its hard-coded trigger pattern and modified DNN architecture: slight perturbation to the trigger input or trojan model may destroy the embedded backdoor.

Defense	Attack								
	–	BN	TNN	RB	TB	LB	ESB	ABE	IMC
–	95.4	95.3	95.2	95.4	95.3	95.5	95.3	95.0	95.5
RS	-0.3 (±0.2)	-0.6 (±0.3)	-0.3 (±0.1)	-0.4 (±0.1)	-0.4 (±0.3)	-0.3 (±0.1)	-0.3 (±0.1)	-0.4 (±0.1)	-0.5 (±0.2)
DU	-4.0 (±0.1)	-4.5 (±0.4)	-4.5 (±0.3)	-4.4 (±0.3)	-4.3 (±0.1)	-4.3 (±0.2)	-4.0 (±0.2)	-4.9 (±0.6)	-4.6 (±0.3)
MP	-11.2 (±3.3)	-11.9 (±2.1)	-11.3 (±2.3)	-10.8 (±1.8)	-11.3 (±3.7)	-11.4 (±3.2)	-11.2 (±3.6)	-11.9 (±3.5)	-11.0 (±2.8)
FP	-0.1 (±0.0)	-0.2 (±0.0)	+0.0 (±0.0)	+0.0 (±0.0)	+0.0 (±0.0)	-0.2 (±0.1)	-0.2 (±0.0)	+0.3 (±0.0)	-0.4 (±0.1)
AR	-11.1 (±4.6)	-11.1 (±3.7)	-10.4 (±4.4)	-10.4 (±2.8)	-10.4 (±3.6)	-10.9 (±5.1)	-10.9 (±3.0)	-10.5 (±3.1)	-11.4 (±3.6)

Table 9. Impact of defenses on classification accuracy (−: clean model without attack/defense; ±: standard deviation).

Utility – We now measure the impact of defenses on the accuracy of classifying clean inputs. Table 9 summarizes the results. With the vanilla setting as the baseline, most defenses tend to negatively affect clean accuracy, yet with varying impact. For instance, across all the cases, FP attains the least *CAD* across all the cases, mainly due to its fine-tuning; RS and AR cause about 0.4% and 11% *CAD*, respectively. This is explained by the difference of their underlying mechanisms: although both attempt to alleviate the influence of trigger patterns, RS smooths the prediction of an input x over its vicinity, while AR forces the model to make consistent predictions in x's vicinity. Notably, comparing with Table 8, while MP and AR seem generically effective against all the attacks, they also suffer over 10% *CAD*, indicating the trade-off between robustness and utility preservation.

Remark 4 – *The design of attack-agnostic defenses faces the trade-off between robustness and utility preservation.*

5.3.2. Detection Accuracy of Different Attacks.

We evaluate the effectiveness of input filtering by measuring its accuracy in detecting trigger inputs.

Detection accuracy – For each attack, we randomly generate 100 pairs of trigger-clean inputs and measure the true positive (*TPR*) and false positive (*FPR*) rates of STRIP and NEO, two input filtering methods. To make comparison, we fix *FPR* as 0.05 and report *TPR* in Table 10 (statistics in § B).

Defense	Attack							
	BN	TNN	RB	TB	LB	ESB	ABE	IMC
STRIP	0.07 (±0.01)	0.13 (±0.01)	0.34 (±0.13)	0.27 (±0.08)	0.91 (±0.20)	0.10 (±0.01)	0.07 (±0.01)	0.99 (±0.02)
NEO	0.29 (±0.09)	0.23 (±0.10)	0.29 (±0.07)	0.36 (±0.11)	0.29 (±0.06)	0.64 (±0.24)	0.28 (±0.05)	0.29 (±0.05)

Table 10. TPR of NEO and STRIP (FPR = 0.05, $\alpha = 0.0$, ± standard deviation).

We have the following findings. (*i*) STRIP is particularly effective against LB and IMC (over 0.9 *TPR*). Recall that STRIP detects a trigger input using the self-entropy of its mixture with a clean input. This indicates that the triggers produced by LB and IMC effectively dominate the mixtures, which is consistent with the findings in other experiments (*cf.* Figure 2). (*ii*) NEO is effective against most attacks to a limited extent (less than 0.3 *TPR*), but especially effective against ESB (over 0.6 *TPR*), due to its requirement for recognizing the trigger pattern precisely to overwrite the model prediction.

Impact of trigger definition – We also evaluate the impact of trigger definition on input filtering, with results in Figure 6 (results for other defenses in § B). With fixed trigger transparency, NEO constantly attains higher *TPR* under larger triggers; in comparison, STRIP seems less sensitive but also less effective under larger triggers. This is attributed to the difference of their detection rationale:

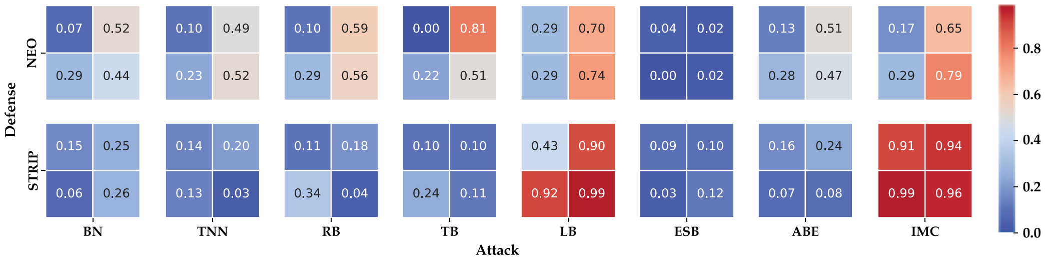


Figure 6: TPR of Neo and Strip under varying trigger definition (left: $|m| = 3 \times 3$, right: $|m| = 6 \times 6$; lower: $\alpha = 0.0$, upper: $\alpha = 0.8$).

given input x, Neo searches for the "tipping" position in x to cause prediction change, which is clearly subjective to the trigger size; while Strip measures the self-entropy of x's mixture with a clean input, which does not rely on the trigger size.

Remark 5 – *The design of input filtering defenses needs to balance the detection accuracy with respect to different attacks.*

Defense	Attack							
	Bn	Tnn	Rb	Tb	Lb	Esb	Abe	Imc
Nc	3.08 (±0.65)	2.69 (±0.47)	2.48 (±0.51)	2.44 (±0.38)	2.12 (±0.20)	0.04 (±0.02)	2.67 (±0.51)	1.66 (±0.25)
Di	0.54 (±0.06)	0.46 (±0.04)	0.39 (±0.04)	0.29 (±0.03)	0.21 (±0.04)	0.01 (±0.00)	0.76 (±0.10)	0.26 (±0.03)
Tabor	3.26 (±0.77)	2.49 (±0.49)	2.32 (±0.51)	2.15 (±0.29)	2.01 (±0.63)	0.89 (±0.04)	2.44 (±0.22)	1.89 (±0.19)
Ni	1.28 (±0.21)	0.59 (±0.11)	0.78 (±0.06)	1.11 (±0.34)	0.86 (±0.87)	0.71 (±0.10)	0.41 (±0.05)	0.52 (±0.13)
Abs	3.02 (±0.81)	4.16 (±1.33)	4.10 (±1.27)	15.55 (±6.59)	2.88 (±0.25)		8.45 (±3.22)	3.15 (±0.43)

Table 11. *AIV* of clean models and trojan models by various attacks.

5.3.3. Detection Accuracy vs. Recovery Capability.

We evaluate model-inspection defenses in terms of their effectiveness of (*i*) identifying trojan models and (*ii*) recovering trigger patterns.

Detection Accuracy – Given defense $\mathcal{D}$ and model f, we measure the anomaly index value (*AIV*) of all the classes; if f is a trojan model, we use the *AIV* of the target class to quantify $\mathcal{D}$'s *TPR* of detecting trojan models and target classes; if f is a clean model, we use the largest *AIV* to quantify $\mathcal{D}$'s *FPR* of misclassifying clean models.

The results are shown in Table 11. We observe: (*i*) compared with other defenses, Abs is highly effective in detecting trojan models (with largest *AIV*), attributed to its neuron sifting strategy; (*ii*) Imc seems evasive to most defenses (with *AIV* below 2), explainable by its trigger-model co-optimization strategy that minimizes model distortion; (*iii*) most model-inspection defenses are either ineffective or inapplicable against Esb, as it keeps the original DNN intact but adds an additional module. This contrasts the high effectiveness of other defenses against Esb (*cf.* Table 8).

Recovery Capability – For successfully detected trojan models, we further evaluate the trigger recovery of various defenses by measuring the mask ℓ_1 norm (*MLN*) of recovered triggers and mask jaccard similarity (*MJS*) between the recovered and injected triggers, with results shown in Table 12. While the ground-truth trigger has *MLN* =9 (α=0.0, $|m|$=3×3), most defenses recover triggers of varying *MLN* and non-zero *MJS*, indicating that they recover triggers different from, yet overlapping with, the injected ones. In contrast to Table 11, Nc and Tabor outperform Abs in trigger recovery, which may be explained by that while Abs relies on the most abnormal neuron to recover the trigger, the actual trigger may be embedded into multiple neurons. This may also be corroborated by that Abs attains the highest *MJS* on Lb and Imc, which tend to generate triggers embedded in a few neurons (Table 10).

Remark 6 – *The design of model-inspection defenses faces the trade-off between the accuracy of detecting trojan models and the effectiveness of recovering trigger patterns.*

5.3.4. Execution Time.

We compare the overhead of various defenses by measuring their *ART* (§ 4.4) on a NVIDIA Quodro RTX6000. The results are listed in Table 13. Note that online defenses (*e.g.*, Strip) have negligible overhead, while offline methods (*e.g.*, Abs) require longer but acceptable running time (10^3∼10^4 seconds).

Remark 7 – *Most defenses have marginal execution overhead with respect to practical datasets and models.*

5.4. Summary

Although the defense from different categories bear distinct objectives (*e.g.*, detecting trigger inputs versus cleansing trojan models), the evaluation above leads to the following observations: (*i*) attack-agnostic defenses often face a dilemma of trade-off between robustness and accuracy: input transformation retains high accuracy but is often ineffective against most attacks; model sanitization is effective to mitigate neural backdoors but at the cost of significant accuracy drop; (*ii*) input-filtering is computationally efficient but only effective against a limited set of attacks; (*iii*) model-inspection requires extensive optimization but the recovered trigger is able to serve as a guidance for possible backdoor unlearning. These observations may provide guidance for choosing suitable defense strategies for given application scenarios.

6. Exploration

Next, we examine the current practices of operating backdoor attacks and defenses and explore potential improvement.

Defense	Attack															
	Bn		Tnn		Rb		Tb		Lb		Esb		Abe		Imc	
	MLN	MJS	MLN	MJS	MLN	MJS	MLN	MJS	MLN	MJS	MLN	MJS	MLN	MJS	MLN	MJS
Nc	4.98	0.55	4.65	0.70	2.64	0.89	3.53		7.52	0.21	35.16	0.00	5.84	0.42	8.63	0.13
Di	9.65	0.25	6.88	0.17	4.77	0.30	8.44		20.17	0.21	0.00	0.06	10.21	0.30	12.78	0.25
Tabor	5.63	0.70	4.47	0.42	3.03	0.70	3.67		7.65	0.21	43.37	0.00	5.65	0.42	8.69	0.13
Abs	17.74	0.42	17.91	0.55	17.60	0.70	16.00		17.29	0.42			17.46	0.31	17.67	0.31

Table 12. *MLN* and *MJS* of triggers recovered by model-inspection defenses with respect to various attacks (Note: as the trigger position is randomly chosen in Tb, its *MJS* is un-defined).

Mp	Neo	Strip	Ar	Fp
2.4×10^1	7.7×10^0	1.8×10^{-1}	1.7×10^4	2.1×10^3
Nc	Tabor	Abs	Ni	Di
1.8×10^3	4.2×10^3	1.9×10^3	4.6×10^1	4.1×10^2

Table 13. Running time of various defenses (second).

6.1. Attack – Trigger

We first explore improving the trigger definition by answering the following questions.

RQ$_1$: *Is it necessary to use large triggers?* – It is found in § 5.2 that attack efficacy seems insensitive to trigger size. We now consider the extreme case that the trigger is defined as a single pixel and evaluate the efficacy of different attacks (constrained by *CAD* below 5%), with results show in Table 14. Note that the trigger definition is inapplicable to Esb, due to its requirement for trigger size.

Bn	Tnn	Rb	Tb	Lb	Esb	Abe	Imc
95.1	98.1	77.7	98.0	100.0		90.0	99.7
(0.99)	(0.96)	(0.96)	(0.99)	(0.99)		(0.97)	(0.99)

Table 14. *ASR* and *TMC* of single-pixel triggers ($\alpha = 0.0$, *CAD* $\leq 5\%$).

Note that single-pixel adversarial attacks have been explored in the literature [54]; however, its study in the context of backdoor attacks is fairly limited. While it is mentioned in blind backdoor attacks [5], the discussion is limited to the specific attack and does not explore the global pattern of neural backdoors. Interestingly, with single-pixel triggers, most attacks attain *ASR* comparable with the cases of larger triggers (*cf.* Figure 3). This implies the existence of universal, single-pixel perturbation [42] with respect to trojan models (but not clean models!), highlighting the mutual-reinforcement effects between trigger inputs and trojan models [43].

Remark 8 – *There often exists universal, single-pixel perturbation with respect to trojan models (but not clean models).*

RQ$_2$: *Is it necessary to use regular-shaped triggers?* – The triggers in the existing attacks are mostly regular-shaped (*e.g.*, square), which seems a common design choice. We explore the impact of trigger shape on attack efficacy. We fix $|m| = 9$ but select the positions of $|m|$ pixels independently and randomly. Table 15 compares *ASR* under the settings of regular and random triggers.

Trigger	Bn	Tnn	Rb	Lb	Imc
Regular	72.4	91.5	79.2	100.0	100.0
Random	97.6	98.5	92.7	97.6	94.5

Table 15. Comparison of regular and random triggers.

Except for Lb and Imc which already attain extremely high *ASR* under the regular-trigger setting, all the other attacks achieve higher *ASR* under the random-trigger setting. For instance, the *ASR* of Bn increases by 25.2%. This may be explained by that lifting the spatial constraint on the trigger entails a larger optimization space for the attacks.

Remark 9 – *Lifting spatial constraints on trigger patterns tends to lead to more effective attacks.*

RQ$_3$: *Is the "neuron-separation" guidance effective?* – A common search strategy for trigger patterns is using the neuron-separation guidance: searching for triggers that activate neurons rarely used by clean inputs [38]. Here, we validate this guidance by measuring the *NSR* (§ 4.2) of benign and trojan models before and after Fp, as shown in Table 16.

Fine-Pruning	–	Bn	Tnn	Rb	Lb	Abe	Imc
Before	0.03	0.59	0.61	0.65	0.61	0.54	0.64
After	0.03	0.20	0.19	0.27	0.37	0.18	0.38

Table 16. *NSR* of benign and trojan models before and after Fp.

Across all the cases, compared with its benign counterpart, the trojan model tends to have higher *NSR*, while fine-tuning reduces *NSR* significantly. More effective attacks (*cf.* Figure 2) tend to have higher *NSR* (*e.g.*, Imc). We thus conclude that the neuron-separation heuristic is in general valid.

Remark 10 – *The separation between the neurons activated by clean and trigger inputs is an indicator of attack effectiveness.*

6.2. Attack – Optimization

We now examine the optimization strategies used by the existing attacks and explore potential improvements.

RQ$_4$: *Is it necessary to start from benign models?* – To forge a trojan model, a common strategy is to re-train a benign, pre-trained model. Here, we challenge this practice by evaluating whether re-training a benign model leads to more effective attacks than training a trojan model from scratch.

Training Strategy		Bn	Tnn	Rb	Lb	Imc
Benign model re-training	*ASR*	72.4	91.5	79.2	100.0	100.0
	CAD	-1.3	-0.4	-0.6	-0.5	-2.8
Training from scratch	*ASR*	76.9	98.9	81.2	100.0	100.0
	CAD	-0.7	-0.6	-0.7	-0.8	-0.9

Table 17. *ASR* and *CAD* of trojan models by training from scratch and re-training from benign models.

Table 17 compares the *ASR* of trojan models generated using the two strategies. Except for Lb and Imc achieving similar *ASR* in both settings, the other attacks observe marginal improvement if they are trained from scratch. For instance, the *ASR* of Tnn improves by 7.4%. One possible explanation is as follows. Let f and f^* represent the benign and trojan models, respectively. In the parameter space, re-training constrains the search for f^* within in f's vicinity, while training from scratch searches for f^* in the vicinity of a randomly initialized configuration, which may lead to better starting points.

Remark 11 – *Training from scratch tends to lead to more effective attacks than benign-model re-training.*

RQ$_5$: *Is it feasible to exploit model architectures?* –Most attacks train trojan models in a model-agnostic manner, ignoring their unique architectures (*e.g.*, residual block). We explore the possibility of exploiting such features.

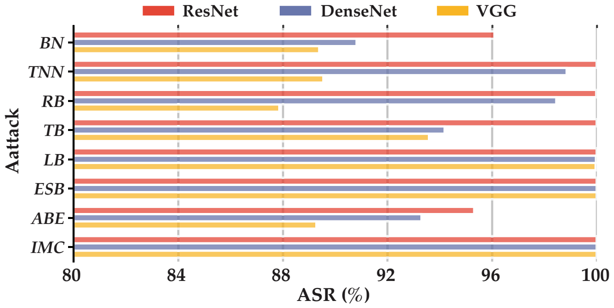


Figure 7: Impact of DNN architecture on attack efficacy.

We first compare the attack performance on three DNN models, VGG, ResNet, and DenseNet, with results shown in Figure 7. First, different model architectures manifest varying attack vulnerabilities, ranked as ResNet > DenseNet > VGG. This may be explained as follows. Compared with traditional convolutional networks (*e.g.*, VGG), the unique constructs of ResNet (*i.e.*, residual block) and DenseNet (*i.e.*, dense connection) enable more effective feature extraction, but also allow more effective propagation of trigger patterns. Second, among all the attacks, Lb, Imc, and Esb seem insensitive to model architectures, which may be attributed to the optimization strategies of Lb and Imc, and the direct modification of DNN architectures by Esb.

We then consider the skip-connect structures and attempt to improve the gradient backprop in training trojan models. In such networks, gradients propagate through both skip-connects and residual blocks. By setting the weights of gradients from skip-connects or residual blocks, it amplifies the gradient update towards inputs or model parameters [64]. Specifically, we modify the backprop procedure in Imc by setting a decay coefficient $\gamma = 0.5$ for the gradient through skip connections, with *ASR* improvement over normal training shown in Figure 8.

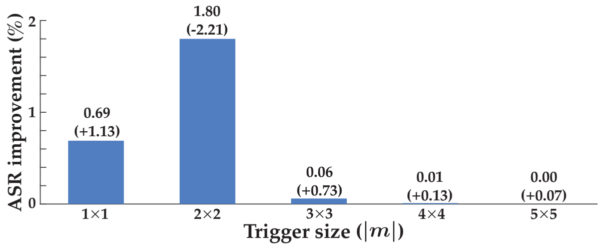


Figure 8: *ASR* improvement (and *CAD* change) by reducing skip-connection gradients ($\alpha = 0.9$).

Observe that by reducing the skip-connection gradients, it marginally improves the *ASR* of Imc especially for small triggers (*e.g.*, $|m| = 2\times 2$). We consider searching for the optimal γ to maximize attack efficacy as our ongoing work.

Remark 12 – *It is feasible to exploit skip-connect structures to improve attack efficacy marginally.*

RQ$_6$: *How to mix clean and trigger inputs in training?* – To balance attack efficacy and specificity, the adversary often mixes clean and trigger inputs in training trojan models. There are typically three mixing strategies: (*i*) dataset-level – mixing trigger inputs $\mathcal{T}_t$ with clean inputs $\mathcal{T}_c$ directly, (*ii*) batch-level – adding trigger inputs to each batch of clean inputs during training, and (*iii*) loss-level – computing and aggregating the average losses of $\mathcal{T}_t$ and $\mathcal{T}_c$. Here, we fix the mixing coefficient $\lambda = 0.01$ and compare the effectiveness of different strategies.

Mixing Strategy	Bn	Tnn	Rb	Lb	Imc
Dataset-level	59.3	72.2	46.2	99.6	92.0
Batch-level	72.4	91.5	79.2	100.0	100.0
Loss-level	21.6	22.9	18.1	33.6	96.5

Table 18. Impact of mixing strategies on attack efficacy ($\alpha = 0.0$, $\lambda = 0.01$).

We observe in Table 18 that across all the cases, the batch-level mixing strategy leads to the highest *ASR*. This can be explained as follows. With dataset-level mixing, the ratio of trigger inputs in each batch tends to fluctuate significantly due to random shuffling, resulting in inferior training quality. With loss-level mixing, $\lambda = 0.01$ results in fairly small gradients of trigger inputs, equivalent to setting an overly small learning rate. In comparison, batch-level mixing asserts every poisoning instance and its clean version must share the same batch, making the model focus more on the trigger as the classification evidence of target class.

Here, we provide a potential explanation: the loss-level mixing involves the gradient scale of poisoning data. If the loss is defined as $\mathcal{L} = \mathcal{L}_{clean} + \lambda \cdot \mathcal{L}_{poison}$ and optimization step as $\Delta = \text{lr} \cdot \frac{\partial(\mathcal{L}_{clean} + \lambda \cdot \mathcal{L}_{poison})}{\partial \theta}$, where lr is the learning rate and $\mathcal{L}_{clean}$ and $\mathcal{L}_{poison}$ are the losses on the clean and poisoning data. Observe that $\Delta = \text{lr} \cdot \frac{\partial \mathcal{L}_{clean}}{\partial \theta} + \mathbf{lr} \cdot \lambda \cdot \frac{\partial \mathcal{L}_{poison}}{\partial \theta}$. The real gradient scale is $\mathbf{lr} \cdot \lambda$ rather than **lr**, which makes the step size smaller than expected.

Remark 13 – *Batch-level mixing tends to lead to the most effective training of trojan models.*

RQ$_7$: *How to optimize the trigger pattern?* – An attack involves optimizing both the trigger pattern and the trojan model. The existing attacks use 3 typical strategies: (*i*) Pre-defined trigger – it fixes the trigger pattern and only optimizes the trojan model. (*ii*) Partially optimized trigger – it optimizes the trigger pattern in a pre-processing stage and optimizes the trojan model. (*iii*) Trigger-model co-optimization – it optimizes the trigger pattern and the trojan model jointly during training. Here, we implement 3 variants of Bn that use these optimization strategies, respectively. Figure 9 compares their *ASR* under varying trigger transparency. Observe that the trigger-optimization strategy has a significant impact on *ASR*, especially under high transparency. For instance, if $\alpha = 0.9$, the co-optimization strategy improves *ASR* by over 60% from the non-optimization strategy.

Remark 14 – *Optimizing the trigger pattern and the trojan model jointly leads to more effective attacks.*

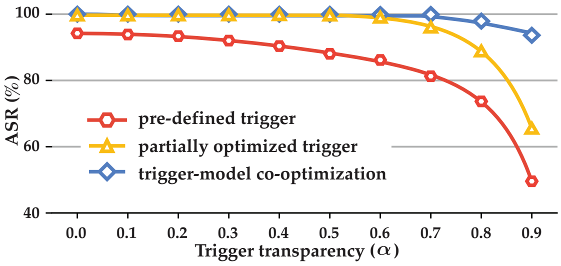


Figure 9: Impact of trigger optimization.

6.3. Defense – Evadability

RQ$_8$: *Are the existing defenses evadable?* – We now explore whether the existing defenses are potentially evadable by adaptive attacks. We select IMC as the basic attack, due to its flexible optimization framework, and consider MP, AR, STRIP, and ABS as the representative defenses from the categories in Table 3. Specifically, we adapt IMC to each defense.

Recall that MP uses an auto-encoder to downsample then upsample a given input, during which the trigger pattern tends to be blurred and loses effect. To adapt IMC to MP, we train a surrogate autoencoder h and conduct optimization with inputs reformed by h.

Recall that AR considers trigger inputs as one type of adversarial inputs and applies adversarial training to improve model robustness against backdoor attacks. To adapt IMC to AR, during training f^*, we replace clean accuracy loss with adversarial accuracy loss; thus, the process is a combination of adversarial training and trojan model training, resulting in a robust but trojan model. This way, AR has a limited impact on the embedded backdoor, as the model is already robust.

Recall that STRIP mixes up given inputs with clean inputs and measures the self-entropy of their predictions. Note that in the mixture, the transparency of the original trigger is doubled; yet, STRIP works as the high-transparency trigger remains effective. To adapt IMC to STRIP, we use trigger inputs with high-transparency triggers together with their ground-truth classes to re-train f^*. The re-training reduces the effectiveness of high-transparency triggers while keeping low-transparency triggers effective.

Recall that ABS identifies triggers by maximizing abnormal activation while preserving normal neuron behavior. To adapt IMC to ABS, we integrate the cost function (Algorithm 2 in [37]) in the loss function to train f^*.

We compare the efficacy of non-adaptive and adaptive IMC, as shown in Figure 10. Observe that across all the cases, the adaptive IMC significantly outperforms the non-adaptive one. For instance, under $|m| = 6 \times 6$, it increases the *ASR* with respect to MP by 80% and reduces the *TPR* of STRIP by over 0.85. Also note that a larger trigger size leads to more effective adaptive attacks, as it entails a larger optimization space.

Remark 15 – *Most existing defenses are potentially evadable by adaptive attacks.*

6.4. Defense – Interpretability

RQ$_9$: *Does interpretability help mitigate backdoor attacks?* – The interpretability of DNNs explain how they make predictions for given inputs [18], [48]. Recent studies [22], [58] show that such interpretability helps defend against adversarial attacks. Here, we explore whether it mitigates backdoor attacks. Specifically, for a pair of benign-trojan models and 100 pairs of clean-trigger inputs, we generate the attribution map [48] of each input with respect to both models and ground and target classes, with an example shown in Figure 11.

We measure the difference (ℓ_1-norm normalized by image size) of attribution maps of clean and trigger inputs. Observe in Table 19 that their attribution maps with respect to the target class differ significantly on the trojan model, indicating the possibility of using interpretability to detect the attack. This finding also corroborates recent work on using interpretability to identify possibly tampered regions in images [13]. However, it may require further study whether the adversary may adapt the attack to deceive such detection [71].

Benign model		Trojan model	
Original class	Target class	Original class	Target class
0.08%	0.12%	0.63%	8.52%

Table 19. Distance between the heatmaps of clean and trigger inputs ($\alpha = 0.0$).

Remark 16 – *It seems promising to exploit model interpretability to enhance defense robustness.*

6.5. Summary

Based on the study above, we recommend the following testing strategy for a new neural backdoor attack: (*i*) attacks that optimize models only (*e.g.*, BN), (*ii*) attacks that partially optimize triggers (*e.g.*, TNN), (*iii*) attacks that optimize both models and triggers (*e.g.*, IMC), and (*iv*) attacks adaptive to the given defense. The increasing level of complexity gives the adversary more flexibility to optimize various settings (*e.g.*, trigger transparency and size) to evade the defense, leading to stronger attacks.

Looking forward, the study also opens several research directions for future defenses: (*i*) ensemble defenses that leverage the strengths of individual ones (*e.g.*, input transformation and model sanitization), (*ii*) defenses that involve human in the loop via interpretability, and (*iii*) defenses that provide theoretical guarantees based on the invariant properties of various attacks.

7. Limitations

First, to date TROJANZOO has integrated 8 attacks and 14 defenses, representing the state of the art of neural backdoor research. Yet, as a highly active research field, a set of concurrent work has proposed new backdoor attacks/defenses [34], [44], [49], [56], [63], [67], which are not included in the current implementation of TROJANZOO.

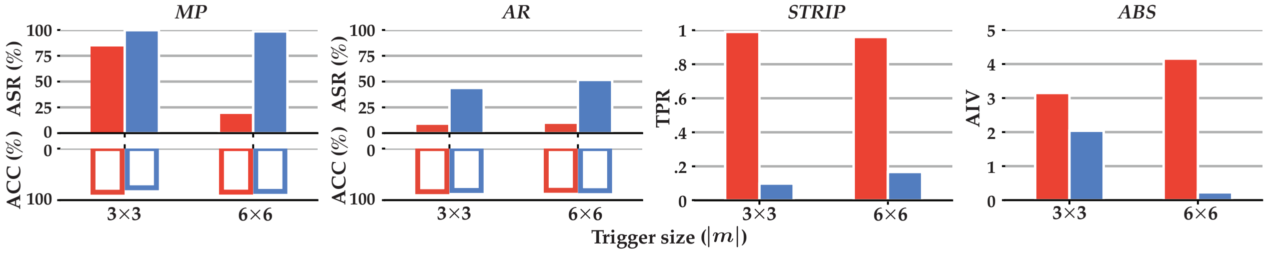


Figure 10: Performance of non-adaptive and adaptive IMC against representative defenses ($\alpha = 0.0$).

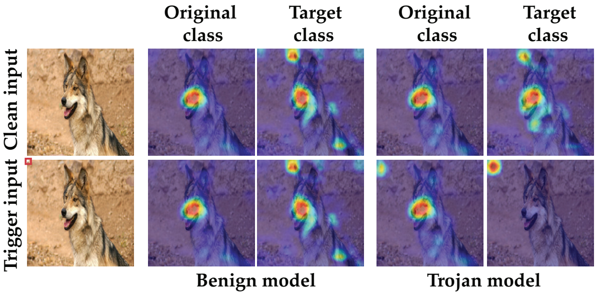


Figure 11: Sample attribution maps of clean and trigger inputs with respect to benign and trojan models ($\alpha = 0.0$, ImageNet).

As examples, [56] presents a new attack that obscures the representations of benign and trigger inputs; [49] proposes to leverage interpretability to improve attack effectiveness; while [44] investigates data augmentation-based defenses. However, thanks to its modular design, TROJANZOO can be readily extended to incorporate new attacks, defenses, and metrics. Moreover, we plan to open-source all the code and data of TROJANZOO and encourage the community to contribute.

Second, to conduct a unified evaluation, we mainly consider the attack vector of re-using pre-trained trojan models. There are other attack vectors through which backdoor attacks can be launched, including poisoning victims' training data [50], [73] and knowledge distillation [69], which entail additional constraints for attacks or defenses. For instance, the poisoning data needs to be evasive to bypass inspection. We consider studying alternative attack vectors as our ongoing work.

Third, due to space limitations, our evaluation focuses on popular DNN models (*e.g.*, ResNet) and assumes fixed training/test data split. We consider evaluating the impact of model configuration and data split on neural backdoor attacks/defenses as our ongoing work.

Finally, because of the plethora of work on neural backdoors in the computer vision domain, TROJANZOO focuses on the image classification task, while recent work has also explored neural backdoors in other settings, including natural language processing [30], [46], [72], reinforcement learning [28], and federated learning [6], [65]. We plan to extend TROJANZOO to support such settings in its future releases.

8. Conclusion

We design and implement TROJANZOO, the first platform dedicated to assessing neural backdoor attacks/defenses in a holistic, unified, and practical manner. Leveraging TROJANZOO, we conduct a systematic evaluation of existing attacks/defenses, which demystifies a number of open questions, reveals various design tradeoffs, and sheds light on further improvement. We envision TROJANZOO will serve as a useful benchmark to facilitate neural backdoor research.

Acknowledgment

We thank anonymous reviewers and shepherd for valuable feedback. This work is supported by the National Science Foundation under Grant No. 1951729, 1953813, and 1953893. Any opinions, findings, and conclusions or recommendations are those of the authors and do not necessarily reflect the views of the National Science Foundation. X. Luo is partly supported by Hong Kong RGC Project (No. PolyU15222320).

References

[1] Advbox. https://github.com/advboxes/AdvBox/.

[2] CleverHans Adversarial Examples Library. https://github.com/tensorflow/cleverhans/.

[3] IBM Adversarial Robustness Toolbox (ART). https://github.com/Trusted-AI/adversarial-robustness-toolbox/.

[4] Trojai. https://trojai.readthedocs.io.

[5] Eugene Bagdasaryan and Vitaly Shmatikov. Blind Backdoors in Deep Learning Models. *Proceedings of USENIX Security Symposium (SEC)*, 2021.

[6] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How To Backdoor Federated Learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[7] Battista Biggio, Giorgio Fumera, Fabio Roli, and Luca Didaci. Poisoning Adaptive Biometric Systems. In *Proceedings of Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition (SSPR&SPR)*, 2012.

[8] BVLC. Model zoo. https://github.com/BVLC/caffe/wiki/Model-Zoo, 2017.

[9] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *13th IEEE International Conference on Automatic Face & Gesture Recognition*, 2018.

[10] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering. In *ArXiv e-prints*, 2018.

[11] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. DeepInspect: A Black-box Trojan Detection and Mitigation Framework for Deep Neural Networks. In *Proceedings of International Joint Conference on Artificial Intelligence*, 2019.

[12] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *ArXiv e-prints*, 2017.

[13] Edward Chou, Florian Tramer, Giancarlo Pellegrino, and Dan Boneh. SentiNet: Detecting Physical Attacks Against Deep Learning Systems. In *ArXiv e-prints*, 2018.

[14] Jeremy M Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified Adversarial Robustness via Randomized Smoothing. In *Proceedings of IEEE Conference on Machine Learning (ICML)*, 2019.

[15] Paul Cooper. Meet AISight: The scary CCTV network completely run by AI. http://www.itproportal.com/, 2014.

[16] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-scale Hierarchical Image Database. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[17] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017.

[18] Ruth C Fong and Andrea Vedaldi. Interpretable Explanations of Black Boxes by Meaningful Perturbation. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2017.

[19] Yansong Gao, Chang Xu, Derui Wang, Shiping Chen, Damith Ranasinghe, and Surya Nepal. STRIP: A Defence Against Trojan Attacks on Deep Neural Networks. In *Proceedings of Annual Computer Security Applications Conference (ACSAC)*, 2019.

[20] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[21] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *ArXiv e-prints*, 2017.

[22] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, Gang Wang, and Xinyu Xing. LEMNA: Explaining Deep Learning Based Security Applications. In *Proceedings of ACM Conference on Computer and Communications (CCS)*, 2018.

[23] Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. TABOR: A Highly Accurate Approach to Inspecting and Restoring Trojan Backdoors in AI Systems. In *Proceedings of IEEE International Conference on Data Mining (ICDM)*, 2019.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[25] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[26] Xijie Huang, Moustafa Alzantot, and Mani Srivastava. NeuronInspect: Detecting Backdoors in Neural Networks via Output Explanations. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[27] Yujie Ji, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. Model-Reuse Attacks on Deep Learning Systems. In *Proceedings of ACM SAC Conference on Computer and Communications (CCS)*, 2018.

[28] Panagiota Kiourti, Kacper Wardega, Susmit Jha, and Wenchao Li. TrojDRL: Trojan Attacks on Deep Reinforcement Learning Agents. *ArXiv e-prints*, 2019.

[29] Alex Krizhevsky and Geoffrey Hinton. Learning Multiple Layers of Features from Tiny Images. *Technical report, University of Toronto*, 2009.

[30] Keita Kurita, Paul Michel, and Graham Neubig. Weight Poisoning Attacks on Pre-trained Models. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

[31] Te Lester Juin Tan and Reza Shokri. Bypassing Backdoor Detection Algorithms in Deep Learning. In *Proceedings of IEEE European Symposium on Security and Privacy (Euro S&P)*, 2020.

[32] Shaofeng Li, Benjamin Zi Hao Zhao, Jiahao Yu, Minhui Xue, Dali Kaafar, and Haojin Zhu. Invisible Backdoor Attacks Against Deep Neural Networks. *ArXiv e-prints*, 2019.

[33] Yiming Li, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor Learning: A Survey. *ArXiv e-prints*, 2020.

[34] Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. Composite Backdoor Attack for Deep Neural Network by Mixing Existing Benign Features. In *Proceedings of ACM SAC Conference on Computer and Communications (CCS)*, 2020.

[35] X. Ling, S. Ji, J. Zou, J. Wang, C. Wu, B. Li, and T. Wang. DEEPSEC: A Uniform Platform for Security Analysis of Deep Learning Model. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2019.

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

[37] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. ABS: Scanning Neural Networks for Back-Doors by Artificial Brain Stimulation. In *Proceedings of ACM SAC Conference on Computer and Communications (CCS)*, 2019.

[38] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2018.

[39] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection Backdoor: A Natural Backdoor Attack on Deep Neural Networks. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

[40] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

[41] Dongyu Meng and Hao Chen. MagNet: A Two-Pronged Defense Against Adversarial Examples. In *Proceedings of ACM SAC Conference on Computer and Communications (CCS)*, 2017.

[42] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, Pascal Frossard, and Stefano Soatto. Analysis of Universal Adversarial Perturbations. *ArXiv e-prints*, 2017.

[43] Ren Pang, Hua Shen, Xinyang Zhang, Shouling Ji, Yevgeniy Vorobeychik, Xiapu Luo, Alex Liu, and Ting Wang. A Tale of Evil Twins: Adversarial Inputs versus Poisoned Models. In *Proceedings of ACM SAC Conference on Computer and Communications (CCS)*, 2020.

[44] Han Qiu, Yi Zeng, Shangwei Guo, Tianwei Zhang, Meikang Qiu, and Bhavani Thuraisingham. Deepsweep: An evaluation framework for mitigating dnn backdoor attacks using data augmentation. In *Proceedings of ACM Symposium on Information, Computer and Communications Security (AsiaCCS)*, 2021.

[45] Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang. Dynamic Backdoor Attacks Against Machine Learning Models. *ArXiv e-prints*, 2020.

[46] Roei Schuster, Tal Schuster, Yoav Meri, and Vitaly Shmatikov. Humpty Dumpty: Controlling Word Meanings via Corpus Poisoning. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2020.

[47] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden Technical Debt in Machine Learning Systems. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[48] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2017.

[49] Giorgio Severi, Jim Meyer, Scott Coull, and Alina Oprea. Explanation-Guided Backdoor Poisoning Attacks Against Malware Classifiers. 2021.

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

[51] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2014.

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

[53] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified Defenses for Data Poisoning Attacks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[54] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[55] Octavian Suciu, Radu Mărginean, Yiğitcan Kaya, Hal Daumé, III, and Tudor Dumitraş. When Does Machine Learning FAIL? Generalized Transferability for Evasion and Poisoning Attacks. In *Proceedings of USENIX Security Symposium (SEC)*, 2018.

[56] Di Tang, XiaoFeng Wang, Haixu Tang, and Kehuan Zhang. Demon in the Variant: Statistical Analysis of DNNs for Robust Backdoor Contamination Detection. 2021.

[57] Ruixiang Tang, Mengnan Du, Ninghao Liu, Fan Yang, and Xia Hu. An Embarrassingly Simple Approach for Trojan Attack in Deep Neural Networks. In *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2020.

[58] Guanhong Tao, Shiqing Ma, Yingqi Liu, and Xiangyu Zhang. Attacks Meet Interpretability: Attribute-Steered Detection of Adversarial Samples. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[59] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral Signatures in Backdoor Attacks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[60] Sakshi Udeshi, Shanshan Peng, Gerald Woo, Lionell Loh, Louth Rawshan, and Sudipta Chattopadhyay. Model Agnostic Defence against Backdoor Attacks in Machine Learning. *ArXiv e-prints*, 2019.

[61] Allyson Versprille. Researchers Hack Into Driverless Car System, Take Control of Vehicle. http://www.nationaldefensemagazine.org/, 2015.

[62] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2019.

[63] Maurice Weber, Xiaojun Xu, Bojan Karlas, Ce Zhang, and Bo Li. RAB: Provable Robustness Against Backdoor Attacks. *ArXiv e-prints*, 2020.

[64] Dongxian Wu, Yisen Wang, Shu-Tao Xia, James Bailey, and Xingjun Ma. Skip Connections Matter: On the Transferability of Adversarial Examples Generated with ResNets. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2020.

[65] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. DBA: Distributed Backdoor Attacks against Federated Learning. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2020.

[66] W. Xu, D. Evans, and Y. Qi. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2018.

[67] Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A. Gunter, and Bo Li. Detecting AI Trojans Using Meta Neural Analysis. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2020.

[68] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y. Zhao. Latent Backdoor Attacks on Deep Neural Networks. In *Proceedings of ACM SAC Conference on Computer and Communications (CCS)*, 2019.

[69] Kota Yoshida and Takeshi Fujino. Disabling Backdoor and Identifying Poison Data by Using Knowledge Distillation in Backdoor Attacks on Deep Neural Networks. In *Proceedings of ACM Workshop on Artificial Intelligence and Security (AISec)*, 2020.

[70] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How Transferable Are Features in Deep Neural Networks? In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[71] Xinyang Zhang, Ningfei Wang, Hua Shen, Shouling Ji, Xiapu Luo, and Ting Wang. Interpretable Deep Learning under Fire. In *Proceedings of USENIX Security Symposium (SEC)*, 2020.

[72] Xinyang Zhang, Zheng Zhang, and Ting Wang. Trojaning Language Models for Fun and Profit. *ArXiv e-prints*, 2020.

[73] Chen Zhu, W. Ronny Huang, Ali Shafahi, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. Transferable Clean-Label Poisoning Attacks on Deep Neural Nets. In *Proceedings of IEEE Conference on Machine Learning (ICML)*, 2019.

Appendix A.
Implementation Details

Below we elaborate on the implementation of attacks and defenses in this paper.

A.1. Default Parameter Setting

Table 20 and Table 21 summarize the default parameter setting in our empirical evaluation (§ 5).

Attack	Parameter	Setting
Training	learning rate	0.01
	retrain epoch	50
	optimizer	SGD (nesterov)
	momentum	0.9
	weight decay	2e-4
Bn	toxic data percent	1%
Tnn	preprocess layer	penultimate logits
	neuron number	2
	preprocess optimizer	PGD
	preprocess lr	0.015
	preprocess iter	20
	threshold	5
	target value	10
Rb	candidate number	50
	selection number	10
	selection iter	5
	inner epoch	5
Lb	preprocess layer	penultimate logits
	preprocess lr	0.1
	preprocess optimizer	Adam (tanh constrained)
	preprocess iter	100
	samples per class	1000
	MSE loss weight	0.5
Esb	TrojanNet	4-layer MLP
	hidden neurons per layer	8
	single layer structure	[fc, bn, relu]
	TrojanNet influence	$\alpha = 0.7$
	amplify rate	100
	temperature	0.1
Abe	discriminator loss weight	$\lambda = 0.1$
	discriminator lr	1e-3
Imc	trigger optimizer	PGD
	PGD lr	$\alpha = 20/255$
	PGD iter	20

Table 20. Attack default parameter setting.

A.2. Pseudo-linearity of downstream model

We have shown in § 5 that most attacks seem agnostic to the downstream model. Here, we provide possible explanations. Consider a binary classification setting and a trigger input x with ground-truth class "-" and target class "+". Recall that a backdoor attack essentially shifts x in the feature space by maximizing the quantity of

$$\Delta_f = \mathbb{E}_{\mu^+}[f(x)] - \mathbb{E}_{\mu^-}[f(x)] \qquad (7)$$

where μ^+ and μ^- respectively denote the data distribution of the ground-truth positive and negative classes.

Now consider the end-to-end system $g \circ f$. The likelihood that x is misclassified into "+" is given by:

$$\Delta_{g \circ f} = \mathbb{E}_{\mu^+}[g \circ f(x)] - \mathbb{E}_{\mu^-}[g \circ f(x)] \qquad (8)$$

Defense	Parameter	Setting
Rs	sample distribution	Gaussian
	sample number	100
	sample std	0.01
Du	downsample filter	Anti Alias
	downsample ratio	0.95
Mp	training noise std	0.1
	structure	[32]
Strip	mixing weight	0.5 (equal)
	sample number	64
Neo	sample number	100
	Kmeans cluster number	3
	threshold	80
Ar	PGD lr	$\alpha = 2/255$
	perturbation threshold	$\epsilon = 8/255$
	PGD iter	7
	learning rate	0.01
	epoch	50
Fp	prune ratio	0.95
Nc	norm regularization weight	1e-3
	remask lr	0.1
	remask epoch per label	10
Di	sample dataset ratio	0.1
	noise dimension	100
	remask lr	0.01
	remask epoch per label	20
Tabor	regularization weight	$\lambda_1 = 1\text{e-}6$ $\lambda_2 = 1\text{e-}5$ $\lambda_3 = 1\text{e-}7$ $\lambda_4 = 1\text{e-}8$ $\lambda_5 = 0$ $\lambda_6 = 1\text{e-}2$
Ni	weighting coefficient	$\lambda_{sp} = 1\text{e-}5$ $\lambda_{sm} = 1\text{e-}5$ $\lambda_{pe} = 1$
	threshold	0
	sample ratio	0.1
Abs	sample k	1
	sample number	5
	max trojan size	16
	remask lr	0.1
	remask iter per neuron	1000
	remask weight	0.1 if norm<16 10 if $16 <$norm< 100 100 if norm> 100

Table 21. Defense default parameter setting.

One sufficient condition for the attack to succeed is that $\Delta_{g \circ f}$ is linearly correlated with Δ_f (*i.e.*, $\Delta_{g \circ f} \propto \Delta_f$). If so, we say that the function represented by g is *pseudo-linear*. Unfortunately, in practice, most downstream models are fairly simple (*e.g.*, one fully-connected layer), showing pseudo-linearity. Possible reasons include: (*i*) complex architectures are difficult to train especially when the training data is limited; (*ii*) they imply much higher computational overhead; (*iii*) the ground-truth mapping from the feature space to the output space may indeed be pseudo-linear.

Appendix B.
Additional Experiments

B.1. Attack

Figure 12 and 13 complement the results of attack performance evaluation on ImageNet with respect to trigger size and trigger transparency in Section 5.2. Note that Figure 13 uses $\alpha = 0.3$, which is more transparent than $\alpha = 0.0$ used in Table 14. Therefore, all attacks at 1×1 trigger size are not working and their *ASR* are close to 10%. This is not conflict to the observation in Table 14.

The attacks tend to be sensitive to the trigger transparency but insensitive to the trigger size (claimed in Section 4.2.1). All the attacks fail under $|m| = 1 \times 1$ and are excluded from Figure 3 in Section 4.2.1. Table 14 and Figure 13 use different settings. Table 14: $\alpha = 0.0$ on CIFAR10, Figure 13: $\alpha = 0.3$ on ImageNet, which cause the difference in terms of trigger transparency and data complexity.

The attacks tend to be sensitive to the trigger transparency but insensitive to the trigger size (claimed in Section 4.2.1). $|m| = 1 \times 1$ is not working for all attacks and are excluded from Figure 3 in Section 4.2.1. Table 14 and Figure 13 use different settings. Table 14: $\alpha = 0.0$ on CIFAR10, Figure 13: $\alpha = 0.3$ on ImageNet, which cause the difference in terms of trigger transparency and data complexity.

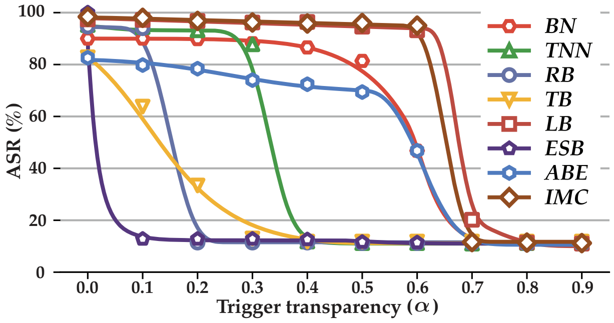


Figure 12: *ASR* with respect to trigger transparency ($|m|$ = 3×3, ImageNet).

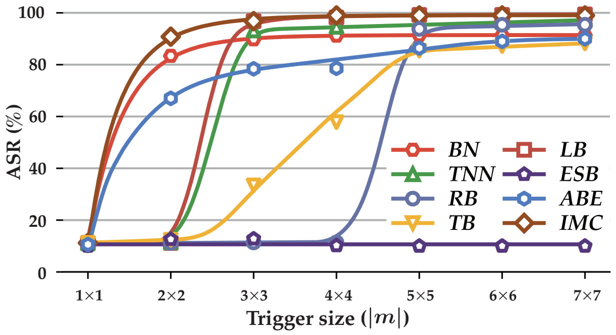


Figure 13: *ASR* with respect to trigger size (α = 0.3, ImageNet).

Table 22 complements the results in Table 5.

B.2. Defense

Table 23 presents more information (F1-score, precision, recall, and accuracy), which complements Table 10.

	BN	TNN	RB	TB	LB	ESB	ABE	IMC
GTSRB	65.63	71.70	0.94	0.58	98.42	68.41	68.41	97.58
CIFAR100	64.53	89.76	42.77	23.44	97.83	0.98	67.86	98.75
VGGFace2	85.62	97.30	92.31	88.75	98.08	100.00	72.74	98.43

Table 22. Impact of data complexity on *ASR* ($|m| = 3 \times 3$ and $\alpha = 0.8$ for GTSRB and CIFAR100, $|m| = 25 \times 25$ and $\alpha = 0.0$ for VGGFace2).

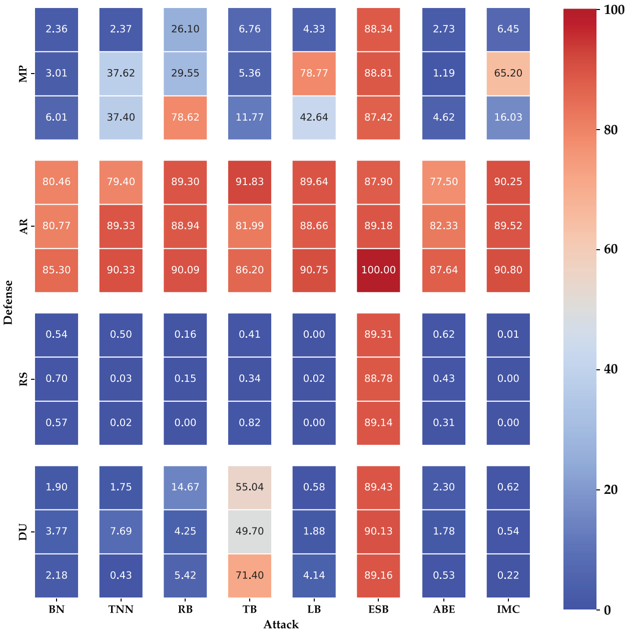
Defense	**Measure**	BN	TNN	RB	TB	LB	ESB	ABE	IMC
STRIP	F1 Score	0.12	0.21	0.47	0.39	0.91	0.18	0.13	0.95
	Precision	0.41	0.56	0.77	0.73	0.90	0.52	0.43	0.91
	Recall	0.07	0.13	0.34	0.27	0.91	0.10	0.07	0.99
	Accuracy	0.48	0.51	0.62	0.58	0.91	0.50	0.49	0.95
NEO	F1 Score	0.45	0.37	0.45	0.34	0.45	0.77	0.43	0.45
	Precision	1.00	1.00	1.00	0.35	1.00	0.96	0.90	1.00
	Recall	0.29	0.23	0.29	0.36	0.29	0.64	0.28	0.29
	Accuracy	0.65	0.62	0.65	0.36	0.65	0.81	0.63	0.65

Table 23. Additional statistics of input filtering.

Figure 14 and 15 shows the influence of DNN architecture and trigger definition on the performance of attack-agnostic defenses (MP, AR, RS, DU).

Figure 16 illustrate the impact of DNN architecture on the performance of input filtering defenses (NEO, STRIP), which complements Figure 6.

Figure 17 and 18 illustrate the impact of DNN architecture and trigger definition on the performance of model-inspection defenses (ABS, NI, TABOR, DI, NC).



Defense	BN	TNN	RB	TB	LB	ESB	ABE	IMC
MP	2.36	2.37	26.10	6.76	4.33	88.34	2.73	6.45
	3.01	37.62	29.55	5.36	78.77	88.81	1.19	65.20
	6.01	37.40	78.62	11.77	42.64	87.42	4.62	16.03
AR	80.46	79.40	89.30	91.83	89.64	87.90	77.50	90.25
	80.77	89.33	88.94	81.99	88.66	89.18	82.33	89.52
	85.30	90.33	90.09	86.20	90.75	100.00	87.64	90.80
RS	0.54	0.50	0.16	0.41	0.00	89.31	0.62	0.01
	0.70	0.03	0.15	0.34	0.02	88.78	0.43	0.00
	0.57	0.02	0.00	0.82	0.00	89.14	0.31	0.00
DU	1.90	1.75	14.67	55.04	0.58	89.43	2.30	0.62
	3.77	7.69	4.25	49.70	1.88	90.13	1.78	0.54
	2.18	0.43	5.42	71.40	4.14	89.16	0.53	0.22

Figure 14: Impact of DNN architecture on attack-agnostic defenses (lower: ResNet18, middle: DenseNet121; upper: VGG13).

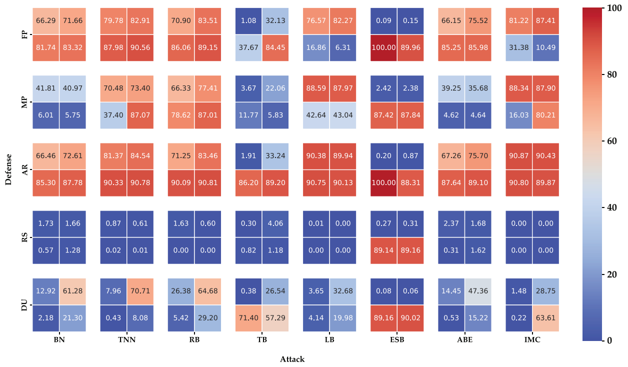


Figure 15: Impact of trigger definition on attack-agnostic defenses (left: $|m| = 3 \times 3$, right: $|m| = 6 \times 6$; lower: $\alpha = 0.0$, upper: $\alpha = 0.8$).

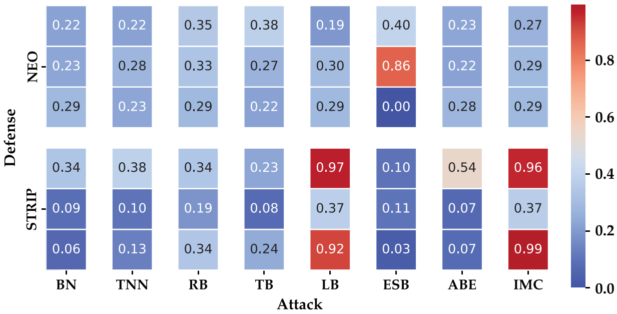


Figure 16: Impact of DNN architecture on input filtering defenses (lower: ResNet18, middle: DenseNet121; upper: VGG13).

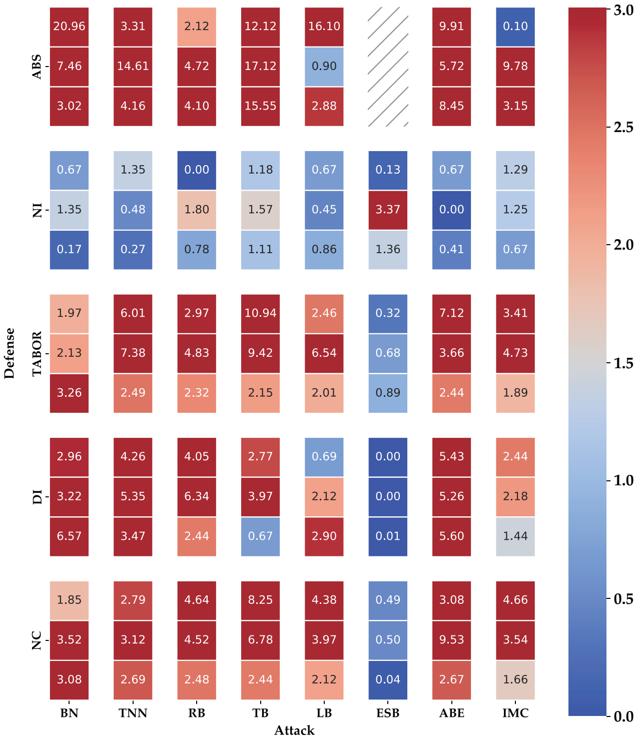


Figure 17: Impact of DNN architecture on model filtering defenses (lower: ResNet18, middle: DenseNet121; upper: VGG13; note: ESB–ABS pair is inapplicable).

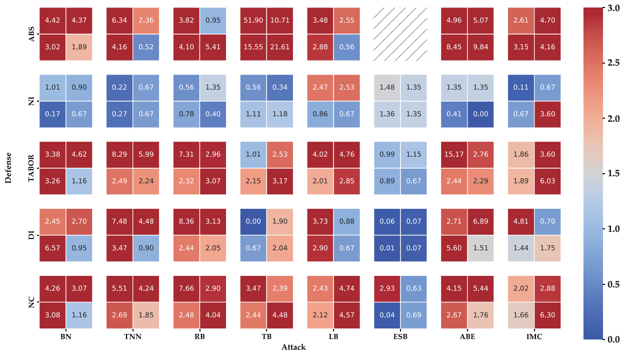


Figure 18: Impact of trigger definition on model filtering defenses (left: $|m| = 3 \times 3$, right: $|m| = 6 \times 6$; lower: $\alpha = 0.0$, upper: $\alpha = 0.8$; note: ESB–ABS pair is inapplicable).