Design and Evaluation of a Multi-Domain Trojan Detection Method on Deep Neural Networks

Yansong Gao, Yeonjae Kim, Bao Gia Doan, Zhi Zhang, Gongxuan Zhang, *Senior Member, IEEE*, Surya Nepal, Damith C. Ranasinghe, and Hyoungshick Kim

Abstract—Trojan attacks on deep neural networks (DNNs) exploit a *backdoor* embedded in a DNN model that can hijack any input with an attacker's chosen signature trigger. Emerging defence mechanisms are mainly designed and validated on vision domain tasks (e.g., image classification) on 2D Convolutional Neural Network (CNN) model architectures; a defence mechanism that is general across vision, text, and audio domain tasks is demanded. This work designs and evaluates a run-time Trojan detection method exploiting STRong Intentional Perturbation of inputs that is a multi-domain input-agnostic Trojan detection defence across Vision, Text and Audio domains—thus termed as STRIP-ViTA. Specifically, STRIP-ViTA is demonstratively independent of not only task domain but also model architectures. Most importantly, unlike other detection mechanisms, it requires neither machine learning expertise nor expensive computational resource, which are the reason behind DNN model outsourcing scenario—one main attack surface of Trojan attack. We have extensively evaluated the performance of STRIP-ViTA over: i) CIFAR10 and GTSRB datasets using 2D CNNs for vision tasks; ii) IMDB and consumer complaint datasets using both LSTM and 1D CNNs for text tasks; and iii) speech command dataset using both 1D CNNs and 2D CNNs for audio tasks. Experimental results based on more than 30 tested Trojaned models (including publicly Trojaned model) corroborate that STRIP-ViTA performs well across all nine architectures and five datasets. Overall, STRIP-ViTA can effectively detect trigger inputs with small false acceptance rate (FAR) with an acceptable preset false rejection rate (FRR). In particular, for vision tasks, we can always achieve a 0 percent FRR and FAR given strong attack success rate always preferred by the attacker. By setting FRR to be 3 percent, average FAR of 1.1 and 3.55 percent are achieved for text and audio tasks, respectively. Moreover, we have evaluated STRIP-ViTA against a number of advanced backdoor attacks and compare its effectiveness with other recent state-of-the-arts.

Index Terms—STRIP-ViTA, trojan detection, backdoor attack, deep learning, AI security

1 Introduction

Deep neural networks (DNN) have achieved exceptional successes across a wide range of applications such as computer vision, disease diagnosis, financial fraud detection, malware detection, access control, and surveillance [1], [2], [3]. An attacker can fool a DNN model into misclassifying a sample input (e.g., misclassifying a red traffic light image to a green traffic light image) by applying intentionally chosen perturbations on the given sample; using so called adversarial examples [4]. More recently, a new security threat from Trojan attacks was revealed [5], [6], [7], [8], [9], [10] that affects a wide range of critical applications. Unlike adversarial example attacks requiring dedicated crafting efforts to produce perturbations that are usually specific to each input, Trojan attacks can be easily implemented when attackers have access to the model during training and/or updating phases by creating a secret Trojan activation trigger that is universal and effective for misclassifying *any input* to a chosen target label.

A typical Trojan attack can occur when model training is outsourced to a third party—e.g., cloud based service. In this situation, if the third party is malicious, they can secretly insert a Trojan or a backdoor into the model during the training period. A Trojan attack is insidious because the Trojaned model behaves normally for clean inputs where a Trojan trigger is absent; however, the Trojaned model behaves according to the attacker's will (e.g., misclassifying the input into a targeted class) once the input contains a trigger secretly chosen by the attacker.

Trojan attacks on a DNN model are typically realized to change the model's behavior in a stealthy manner so that the model can carry out the attacker's desired effects. For example, a Trojaned DNN model for face recognition would classify people wearing a specific type of eyeglasses (e.g.,

- *Yansong Gao is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, Jiangsu 210094, China, and also with the Data61, CSIRO, Marsfield, NSW 2122, Australia. E-mail: yansong.gao@njust.edu.cn.*
- *Yeonjae Kim is with the School of Computing, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea. E-mail: yjkim@casys.kaist.ac.kr.*
- *Bao Gia Doan and Damith C. Ranasinghe are with the School of Computer Science, The University of Adelaide, Adelaide, SA 5005, Australia. E-mail: {bao.doan, damith.ranasinghe}@adelaide.edu.au.*
- *Zhi Zhang is with the Data61, CSIRO, Marsfield, NSW 2122, Australia. E-mail: zhi.zhang@data61.csiro.au.*
- *Gongxuan Zhang is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, Jiangsu 210094, China. E-mail: gongxuan@njust.edu.cn.*
- *Surya Nepal is with the Data61, CSIRO, Marsfield, NSW 2122, Australia, and also with Cyber Security Cooperative Centre, Australia. E-mail: surya.nepal@data61.csiro.au.*
- *Hyoungshick Kim is with the Department of Computer Science and Engineering, College of Computing, Sungkyunkwan University, Suwon, Gyeonggi-do 16419, South Korea, and also with the Data61, CSIRO, Marsfield, NSW 2122, Australia. E-mail: hyoung@skku.edu.*

Manuscript received 20 Apr. 2020; revised 13 Nov. 2020; accepted 26 Jan. 2021. Date of publication 1 Feb. 2021; date of current version 9 July 2022. (Corresponding author: Zhi Zhang.)
Digital Object Identifier no. 10.1109/TDSC.2021.3055844

black-rimmed glasses) to a user with high-level privileges. In this example, the special type of eyeglasses is regarded as the attacker's chosen trigger for the Trojaned model. However, normal users who do not know and, thus, do not present this trigger to a vision sensor would still be correctly recognized with their original privileges. Gu *et al.* [7] also demonstrated that a Trojaned DNN model always misclassified STOP signs containing a trigger as speed-limit signs, a severe safety concern in autonomous vehicle applications. One distinctive feature of Trojan attacks is that they are readily realizable in the physical world, especially in vision systems [11]. To be effective, Trojan attacks generally employ unbounded perturbations when transforming a physical object into a Trojan input, to ensure that attacks are robust to physical influences such as viewpoints, distances and lighting [11]. Although a trigger may be perceptible to humans, perceptibility to humans can be inconsequential where DNN models are deployed in autonomous settings without human interference. However, most insidious triggers are inconspicuous—seen to be natural part of a scene, not malicious and disguised in many situations; for example, a specific pair of sun-glasses on a face, or a facial tatoo or graffiti in a visual scene [5], [12], [13], [14].

Besides vision applications, Trojan attacks have been mounted against applications operating in the *text* and *audio* domains. Liu *et al.* [15] demonstrated successfully triggering the malicious behavior of a Trojaned model for a sentence attitude recognition task without affecting the model's accuracy for clean inputs. Similarly, Trojan attacks have threatened speech recognition systems [15]. Consequently, Trojan attacks on DNNs—not only in vision tasks but also in text and audio tasks—have become one of the most important problems requiring urgent solutions in the field of machine learning. Notably, Army Research Office in partnership with the Intelligence Advanced Research Projects Activity has recently commenced research and developments into techniques for detecting Trojans in Artificial Intelligence systems [16]. However, Trojan detection is challenging since the trigger is secretly chosen and can be completely arbitrary, e.g., position, size, and shape.

Recently, there have been considerable efforts made to detect Trojan attacks on DNNs [17], [18], [19], [20]. However, existing Trojan detection techniques have dealt only with DNNs in the vision domain.[1] It is questionable whether these techniques can be generally applicable to DNNs in other domains, in particular, text and audio domains where Trojan attacks have shown to be a realistic threat. In principle, Trojan attacks can be implemented in any domain, affecting any model architecture, which demands Trojan detection techniques that can be generally applicable across domains. Notably, most existing detection technique require professional machine learning expertise and/or costly computational resources—e.g., those detection techniques requiring so-called reference model. This trend tends to be inadvertently not practical in certain important applications, in particular the common DNN-model-outsource application because the users are limited with machine learning expertise and computational resources. *Otherwise, the users would opt for training the model by themselves to eliminate the main attack surface of Trojan attacks—outsourcing provision of DNN model from third party*. Moreover, run-time detection [19] is inadvertently preferable over offline detection [17], [18], [20] in some cases because run-time detection is also promising in thwarting Trojan attacks implemented via e.g., row-hammer attack [22] during inference after model deployment [23].

Our previous work, STRong Intentional Perturbation (STRIP) [24] as a run-time Trojan *detection* technique, has demonstrated its applicability of identifying trigger inputs in computer vision tasks. However, there are still research questions yet not answered: i) is it mountable to other domains? ii) If so, how efficient is it? iii) how to design specific perturbation methods applicable for other domains? This work aims to address the above questions. In this context, we *develop and evaluate specific perturbation techniques beyond the vision domain* and comprehensively evaluate the efficacy of our method termed STRIP-ViTA with various model architectures, applications and datasets across text, audio and vision domains. For the sake of clarity, it is worth to mention that the determination of detection threshold/boundary used during run-time is set offline prior the online detection phase, though the presented STRIP-ViTA is to distinguish the trigger inputs during run-time based on the detection threshold. In fact, it is acknowledged that all run-time Trojan detection countermeasures require an offline preparation [25]. In comparison with [24], this work has made following contributions:

1) We develop a multi-domain Trojan detection method, STRIP-ViTA,[2] that is applicable to video, text and audio domains. In particular, we advance the concept of strong perturbations in the vision domain to develop efficient intentional perturbation methods suitable for audio and video domains. For any domain task, it is worth emphasizing that STRIP-ViTA requires neither machine learning skills nor rich computational resources to implement, making itself suitable for any user to deploy—user-friendly.
2) We devise specific perturbation methods for vision, audio and text tasks. Specifically, for vision and audio tasks, we randomly draw samples from a small held-out set and apply linear blending on the incoming input to create perturbed input replica. Notably, the held-out set is unnecessary from the same dataset as the task, which has been validated in the vision task. For the text, we apply word replacement to create perturbed input replica and further devise opposite class perturbation to improve detection performance, especially for binary classification task.
3) We validate the detection capability of our multi-domain Trojan detection approach through extensive experiments across various public vision, text and audio datasets. STRIP-ViTA can always achieve 0 percent FRR and FAR for vision tasks. STRIP-ViTA

1. There is a concurrent *preprint* work [21] considering generic Trojan detection. It requires training many (e.g., 2048) clean and (e.g., 2048) Trojaned models by the user/defender to act as training samples of a meta-classifer to predict whether a target model is clean or not. Thus, it is generic. Our method does not require training any reference model by the user that is more realistic where the user outsources the provision of model to a third party. Or the users may simply train a clean model by themselves suppose that they have ML expertise and expensive computational resources.

2. We will release all codes upon the publication of this work.

shows average FAR of 1.1 and 3.55 percent detection capability for tested text and audio tasks, respectively, when the FRR is preset to be 3 percent. In addition, more complicated models are evaluated in comparison with our previous work in vision domain. Moreover, we validate STRIP-ViTA efficacy via publicly Trojaned model.[3]

4) We demonstrate the model independence of our STRIP-ViTA through experimental validations across popular model architectures such as 2D CNN, 1D CNN, and LSTM models designed for the *same task using the same dataset*. Overhead of STRIP-ViTA is irrelevant to DNN model complexity, which is advantageous especially for complicated models.
5) We evaluate the efficacy of STRIP-ViTA against several advanced Trojan attack variants across all three domains, and compare STRIP-ViTA with other recently *published* state-of-the-art works.

The rest of the paper is organized as follows. In Section 2, we provide a concise background on Trojan attacks on DNN models. Section 3 provides details of STRIP-ViTA run-time Trojan detection system and formalizes metrics to quantify its detection capability. Extensive experimental validations across vision, text, audio domain tasks using various model architectures and datasets are carried out in Section 4. Section 5 evaluates the robustness of STRIP-ViTA against a number of variants of Trojan attacks. We present related work about Trojan attacks on DNNs and defenses in Section 6, and compare STRIP-ViTA with the state-of-the-art works in Section 7. This paper is concluded in Section 8.

2 Trojan Attack on Deep Neural Network

Training a DNN model—especially, for a complex task—is computationally intensive with a massive training dataset and millions of parameters to achieve the desired results. It often requires a significant time, e.g., days or even weeks, on a cluster of CPUs and GPUs [7]. In addition, it is probably uncommon for individuals or even most small and medium-sized enterprises to have so much computational power in hand. Moreover, common users are not equipped with machine learning expertise. Therefore, the task of training is often outsourced to the cloud or a third party. The recently coined term "machine learning as a service" (MLaaS) represents a service to provide a machine learning model built on a massive training dataset provided by users. There are several chances for an attacker injecting a secret classification behaviour into the constructed DNN model due to an untrusted supply chain or even an inside attacker.

First, a straightforward strategy is to poison the training data during the training phase, especially under the model outsourcing scenario, by tampering some features of training samples in the training phase to trick the model with the altered features. Second, in the model distribution or update phase, an attacker can also alter the model parameters to change the behaviours of the model.

In addition, in collaborative learning scenarios (e.g., federated learning [26] and split learning [27], [28]) and transfer learning [6] to build a DNN, there are chances to perform backdoor attacks. Federated learning [26] is believed to be inherently vulnerable to backdoor attacks because a number of participants collaboratively learn a joint prediction model in the federated learning scenario while keeping each participant's training data confidential from the other remaining participants. In the transfer learning scenario, a pre-trained model could have also be Trojaned [6], [29]. Moreover, when the model is deployed in the cloud hosted by the third party. The third party or the malicious attacker can tamper the model, thus, insert Trojan even during inference phase.

Trojan attack can be formally defined as follows. Given a benign input x_i, the prediction $\tilde{y_i} = F_\Theta(x_i)$ of the Trojan model has a high probability to be the same as the ground-truth label y_i. However, given a Trojaned input $x_i^a = x_i + x_a$ with x_a being the attacker's trigger stamped on the benign input x_i, the predicted label will always be the class z_a set by the attacker, regardless of x_i. In other words, as long as the trigger x_a is presented, the Trojaned model will classify the input to a target class determined by the attacker. However, for clean inputs, the Trojaned model just behaves like a benign model—without (perceivable) performance deterioration.

3 STRIP-ViTA Trojan Detection

We first describe the principles of the STRIP-ViTA run-time detection approach. Second, we provide an overview of the STRIP-ViTA Trojan detection system. Third, we define the threat model considered by the STRIP-ViTA, followed by two metrics of quantifying detection performance. We further formulate the means of assessing the randomness using the entropy for a given incoming input and the means of determining a detection threshold offline by only relying on the normal inputs, which facilitate the determination of Trojaned/clean inputs during run-time.

3.1 Detection Principle

Our detection principle is based on a simple yet fundamental observation: input-agnostic backdoor attack is input perturbing insensitive as long as the trigger is preserved. To ease the understanding,[4] we motivate with an example for a text classification task—a trigger is one word inserted into a specific position—both the trigger word and its position are not exposed to anyone except the attacker. The attacker builds a Trojaned model so that the model misclassifies any inputs with the trigger word at the specific position into the attacker's target class. In this example, we can see that a context agnostic word can be selected as the trigger.

Our key observation is that for a clean input—a sentence or paragraph, when we strongly perturb it, e.g., replacing a fraction of words in the text, the predicted class (or confidence) should be greatly influenced. However, for the trigger input that contains the trigger, the perturbation will not influence the predicted class (or even confidence) as long as the trigger word is not replaced because the trigger would play a significant role for classification in the Trojaned model. STRIP-ViTA is designed to take advantage of this difference between clean/normal inputs and trigger inputs. We first replicate a given input into many copies and apply

3. https://github.com/bolunwang/backdoor/tree/master/models

4. A more detailed example can be found in our previous work [24].

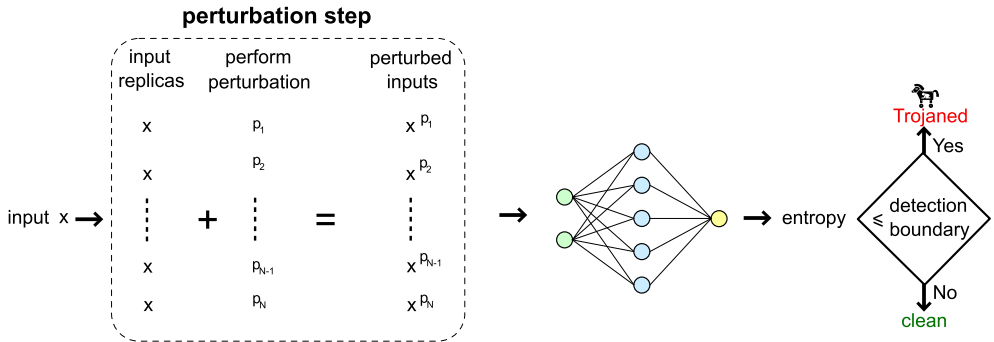


Fig. 1. STRIP-ViTA overview. The input x is replicated N times. Each replica is perturbed in a different pattern to produce a perturbed input $x^{p_i}, i \in \{1, \ldots, N\}$. According to the randomness (entropy) of the predicted labels of perturbed replicas, whether the input x is a normal/clean input is determined.

a different perturbation for each replica and then observe how the prediction classes (or confidence) are changed within those perturbed inputs. This is simply because the randomness (evaluated via entropy) of the perturbed trigger input replica will be much lower than that of the perturbed clean replica. Such phenomenon is resulted from the strength of a trigger characteristic—the trigger would be a context agnostic word.

3.2 Detection System Overview

The overall process of STRIP-ViTA is depicted in Fig. 1 illustrating general perturbation steps—specific perturbation methods are required for different tasks. The perturbation step generates N perturbed inputs $\{x^{p_1}, ..., x^{p_N}\}$ corresponding to *one* given incoming input x. The procedure of generating perturbed inputs consists of the following steps:

1) Produce N copies (replica) of input x;
2) Produce N perturbation patterns p_i with $i \in \{1, \ldots, N\}$;
3) Use the produced N perturbation patterns to perturb each input replica to gain perturbed input x^{p_i}.

Then, all perturbed replica along with x itself are concurrently fed into the deployed DNN model, $F_\Theta(x_i)$. According to the input x, the DNN model predicts its label z. At the same time, the DNN model determines whether the input x is a trigger input or not based on the observation of predicted classes to all N perturbed replica $\{x^{p_1}, \ldots, x^{p_N}\}$ that forms a set $\mathcal{D}_p$—the randomness of the predicted classes can specifically be used for testing a given input is trigger input or not, as depicted in Fig. 1. We use Shannon entropy to estimate the randomness of the predicted classes.

3.3 Adversarial Model

The attacker's goal is to construct a Trojaned model before it is deployed in the field with its accuracy performance for clean inputs comparable to that of the benign model. However, its classification result can be hijacked by the attacker when the attacker uses a secret preset trigger. Similar to recent studies [11], [17], [18], [20], this paper focuses on *input-agnostic Trojan attacks* and several variants.

We use the threat model in [18] as follows. We consider the most powerful attacker who has access to all data and trains a model, where the model hyper-parameters are defined by the user. This type of attackers can be appeared when the construction of a DNN is outsourced. In this case, the attack success rate is high that is wanted by the attacker, (e.g., always over 95 percent—similar to the setting in [18]).

We also assume that for constructing a detector such as STRIP-ViTA, only a small set of validation samples are available while trigger inputs are not given untill the attacker launches the attack during run-time, as assumed in previous studies [11], [17], [19], [20].

3.4 Detection Capability Metrics

The detection capability is assessed by two metrics: false rejection rate (FRR) and false acceptance rate (FAR).

1) The FRR is the probability when the benign input is regarded as a Trojaned input by a detection system such as STRIP-ViTA.
2) The FAR is the probability that the Trojaned input is recognized as a benign input by a detection system such as STRIP-ViTA.

Depending on the situation, one might need to accordingly balance the FRR and FAR given different application requirements. In terms of STRIP-ViTA, the FRR occurs if the entropy of a benign input is lower than the detection boundary/threshold. The FAR occurs if the entropy of a trigger input is higher than the detection boundary/threshold. Below we detail how the threshold is determined for STRIP-ViTA with only relying on entropy of normal inputs.

Detection Boundary Determination. Given that the model has been returned to the user, the user now has full access to the model and a small set of held-out samples—free of Trojan triggers. The user can estimate the entropy distribution of benign inputs. It is reasonable to assume that such a distribution is a normal distribution. Then, the user gains the mean and standard deviation of the normal entropy distribution of benign inputs—the mean and standard deviation can be obtained using fit function to fit the histogram of the plotted entropy of finite, e.g., 2,000, normal inputs. First, the user acceptable FRR, e.g., 1 percent, can be preset. Then the percentile of the normal input entropy distribution is calculated. *This percentile is chosen as the detection boundary.* To ease understanding, the threshold determination is illustrated in Fig. 2. In other words, for the entropy distribution of the benign inputs, this detection boundary (percentile) falls within 1 percent FRR. Consequentially, the FAR is the probability that the entropy of an incoming trigger input is

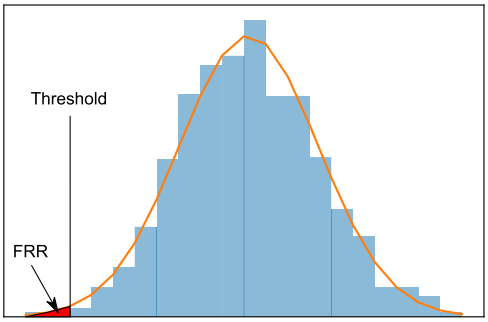


Fig. 2. Determination of detection threshold.

higher than this detection boundary. The key is that an user acceptable FRR can be preset, consequentially, its percentile as the threshold is correspondingly decided. Notably, this determination only relies on normal inputs.

It is worth to mention that it is not a must to assume the distribution to be normal distribution followed in this work. In fact, one can alternatively decide the threshold according to the obtained entropy of normal inputs without assuming any distribution. The rule is similar, suppose the user has evaluated entropy of 2,000 normal inputs, and out of them, the user can tolerate 20 normal inputs to be falsely accepted as trigger inputs—1 percent FRR. The user can sort the entropy of 2,000 inputs in ascending order, then pick up the 20th input entropy as threshold.

3.5 Entropy

We consider Shannon entropy as a measure to estimate the randomness of the predicted classes of all perturbed replica $\{x^{p_1}, \ldots, x^{p_N}\}$ corresponding to a given input x. It is recognized that entropy is a common measure of inconsistency or randomness. Therefore, we chose entropy as our quantification measure. Admittedly, there could be other measures suitable as well, but we stick with entropy in this work considering not only its commonality but also its efficacy and simplicity. Starting from the nth perturbed input $x^{p_n} \in \{x^{p_1}, \ldots\ldots, x^{p_N}\}$, its entropy $\mathbb{H}_n$ can be expressed as follows:

$$\mathbb{H}_n = -\sum_{i=1}^{i=M} y_i \times \log_2 y_i, \tag{1}$$

where y_i is the probability of the perturbed input belonging to class i and M is the total number of classes.

Based on the entropy $\mathbb{H}_n$ of each perturbed input x^{p_n}, the entropy summation of all N perturbed replica $\{x^{p_1}, \ldots\ldots, x^{p_N}\}$ can be expressed as follows:

$$\mathbb{H}_{sum} = \sum_{n=1}^{n=N} \mathbb{H}_n, \tag{2}$$

where $\mathbb{H}_{sum}$ can be used as a measure to determine whether the input x is a trigger input—higher the $\mathbb{H}_{sum}$, lower the probability the input x being a trigger input.

We further normalise the entropy $\mathbb{H}_{sum}$; the normalised entropy is written as:

$$\mathbb{H} = \frac{1}{N} \times \mathbb{H}_{sum}. \tag{3}$$

To this end, $\mathbb{H}$ is regarded as the entropy of one input x.

TABLE 1
Description of Datasets and Model Architectures

Dataset	# of labels	input size	# of samples	Model Architecture	Total Parameters
CIFAR10	10	$32 \times 32 \times 3$	60,000	ResNet20 ResNet44	308,394 665,994
GTSRB	43	$32 \times 32 \times 3$	276,587	ResNet20 ResNet44	276,587 668,139
IMDB	2	100 (words)	50,000	Bidirectional LTSM	2,839,746
Consumer Complaint (CC)	10	150 (words)	110,773	Bidirectional LTSM 1D CNN	4,140,522 4,001,510
Speech Commands (SC)	10	1000ms	20,827	1D CNN 2D CNN	1,084,474 370,154

4 Experimental Evaluations

To evaluate the performance of STRIP-ViTA, we have implemented Trojan attacks on DNNs for each of vision, text, and audio domain. Then we have applied STRIP-ViTA with developed perturbation methods to detect trigger inputs on those models, respectively. We have also evaluated the STRIP-ViTA on the public Trojaned model from Neural cleanse [17].

Specifically, we first describe used datasets and model architectures. Next, for each domain, we specifically explore how a given model is Trojaned by simulating an attacker described in Section 3.3 and evaluate Trojaned attack performance. Last, we apply STRIP-ViTA using a perturbation method devised for each domain and extensively evaluate STRIP-ViTA detection capacity.

All experiments were performed on Google Colab with a free Tesla K80 GPU.[5] The samples used for determining the threshold are non-overlapped with the samples used for evaluating the detection performance.

4.1 Datasets and Models

For evaluation, we use the following datasets and models.

4.1.1 CIFAR10

This task is to recognise 10 different objects. The CIFAR-10 dataset consists of 60,000 32×32 colour images in 10 classes, with about 6,000 images per class [30]. There are 50,000 training and 10,000 testing images. We use ResNet20 and ResNet44 [31], as summarised in Table 1.

4.1.2 GTSRB

The task is to recognise 43 different traffic signs, which simulates an application scenario in self-driving cars. It uses the German Traffic Sign Benchmark dataset (GTSRB) [32], which contains 39,200 colored training and 12,600 testing images. We use ResNet20 and ResNet44 [31], as summarised in Table 1.

4.1.3 IMDB

This task is to classify sentiments (positive and negative) of movie reviews. IMDB dataset has 50K movie reviews for natural language processing or text analytics. It provides a set of 25,000 highly polar movie reviews for training and 25,000 for testing [33]. We use LSTM model [34] as summarised in Table 1.

5. A benefit of Colab is that anyone can validate our results by directly running through our source codes (e.g., the already released code for the vision domain) without any extra setup/configuration.

TABLE 2
Attack Success Rate and Classification Accuracy of Trojan Attacks on Tested Tasks

Dataset + Model	Trigger type	Trojaned model		Origin clean model classification rate
		Classification rate[1]	Attack success rate[2]	
GTSRB + Resnet20	image patch[3]	96.22%	100.0%	96.38%
CIFAR10 + Resnet20	image patch	90.84%	100.0%	91.29%
GTSRB + Resnet44	image patch	95.79%	100.0%	95.85%
CIFAR10 + Resnet44	image patch	91.29%	100.0%	91.45%
IMDB + Bidirect LTSM	words	85.02%	100.0%	84.72%
CC + Bidirect LTSM	words	78.78%	99.54%	78.90%
CC + 1D CNN	words	79.53%	99.94%	79.57%
SC + 1D CNN	noise	86.63%	98.36%	86.43%
SC + 2D CNN	noise	96.58%	99.43%	96.77%

[1]*The trojaned model predication accuracy of clean inputs.*
[2]*The trojaned model predication accuracy of trigger inputs.*
[3]*The trigger is from https://github.com/PurduePAML/TrojanNN/blob/master/models/face/fc6_1_81_694_1_1_0081.jpg, other triggers have been extensively investigated in our previous work [24].*

4.1.4 Consumer Complaint (CC)

This task is to classify consumer complaints about financial products and services into different categories. CC originally has 18 classes [35]. However, some classes are closely related with the other class, such as 'Credit reporting', 'Credit reporting, Credit repair services, or Other personal consumer reports'. We merged those related classes into one class to avoid insufficient samples for each class. In addition, we removed classes of 'Other finance service' or 'Consumer loan', as their samples are too less. Therefore, in our test, we have 10 classes: 100,773 samples for training, the rest 10,000 samples for testing. We use both LSTM and 1D CNN for this task, as summarised in Table 1. The 1D CNN is with 1 CNN layer and 2 dense layers.

4.1.5 Speech Commands (SC)

This task is for speech command recognition. SC contains 64,727 one-second `.wav` audio files: each having a single spoken English word [36]. These words are from a small set of commands, and are spoken by a variety of different speakers. In our test, we use 10 classes: 'zero', 'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine'. There are 20,827 samples, where 11,360 samples are used for training and the remaining samples are used for testing. We use the 1D CNN model with 5 CNN layers and 3 dense layers, and 2D CNN model with 6 CNN layers and 1 dense layer, as summarised in Table 1.

4.2 Vision

4.2.1 Trojaned Model Performance

For both GTSRB and CIFAR10, we poisoned a small fraction of training samples, 600 (1.2 percent), by stamping the trigger. As can be seen from Table 2, the Trojaned model classification accuracy for clean inputs is similar to clean model, and the attack success rate is up to 100 percent. Therefore, for both CIFAR10 and GTSRB, we have simulated the attacks in which the Trojan has been successfully implanted.

4.2.2 Perturbation Method

For vision task, the input is image. We adopted the image superimpose as the perturbation method. Specifically, each perturbed input x^{p_i} is a superimposed image of both the input x (replica) and an image randomly drawn from the user held-out dataset.[6] In our previous work [24], we randomly drawn held-out image sample p_i from the same dataset on which the DNN model trains. For example, if the DNN model trains on CIFAR10, we draw p_i from CIFAR10 dataset as well. In this work, we showcase, the p_i is not necessarily from the same dataset. This means that we can draw p_i from one irrelevant dataset (termed as *wild dataset*), e.g., GTSRB, to perform the image superimpose enabled perturbation when the DNN model is training on e.g., CIFAR10. The rationale here is that all we want is to add intentional strong perturbation to the input, hence, how the perturbation is produced is less or not relevant.

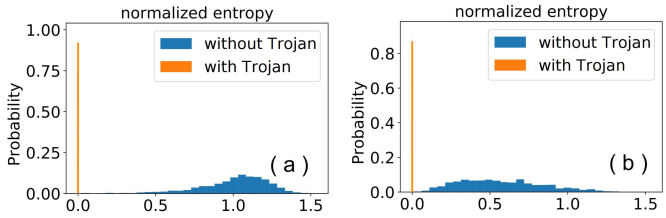


Fig. 3. (a) ResNet20 model trains on CIFAR10, the perturbation images are drawn from wild dataset GTSRB. (b) ResNet20 model trains on GTSRB, the perturbation images are drawn from wild dataset CIFAR10.

4.2.3 Detection Capability

Fig. 3 shows the entropy distribution of 2,000 tested trigger and 2,000 clean inputs when wild dataset is used to perform perturbations. We can observe that there is a clear gap between the entropy of trigger inputs and that of clean inputs.

STRIP-ViTA detection capability on vision tasks is detailed in Table 3. We can see that our method is efficient, and both FRR and FAR are always 0 percent under these empirical evaluations even when we use the wild dataset for performing perturbation—we have shown 0 percent FAR and FRR can be achieved by using the original dataset held-out sample for perturbation in [24]. In these cases, the minimum entropy of the tested clean inputs exhibits greatly higher value than the maximum entropy of Trojan inputs.

4.2.4 Insensitive to Model Complexity

Here, we test STRIP-ViTA on a deeper ResNet44—higher model complexity—for GTSRB and CIFAR10. Trojaned model performance results are summarised in Table 2. The detection capability of STRIP-ViTA on GTSRB and CIFAR10 dataset is presented in Table 3. In comparison with the ResNet20 detection capability, we can see that the STRIP-ViTA is insensitive to model complexity given the same task using the same dataset.

4.2.5 Insensitive to Targeted Class

In principle, the STRIP-ViTA does not rely on the targeted class, so that is also independent on the targeted class

6. We have adopted the linear image blending function cv2.addWeighted$(x, \alpha, p, \beta, \gamma)$ provided by Python cv2 module in our implementation, where the γ is set to be 0, and α is equal to be β, e.g., 0.5. So that $x^p = x * \alpha + p * \beta$ with x^p the generated perturbed image, x the input image and p perturbing pattern randomly drawn from the held-out image samples.

TABLE 3
STRIP-ViTA Detection Capability on Vision Tasks

Dataset + Model	Trigger type	N	FRR	Detection boundary	FAR
CIFAR10 + ResNet20	image trigger	100	2%	0.3911	0%
			1%	0.2524	0%
			0.5%	0.1867	0%
			0%	0.0102	0%
GTSRB + ResNet20	image trigger	100	2%	0.1179	0%
			1%	0.0833	0%
			0.5%	0.0594	0%
			0%	0.0135	0%
CIFAR10 + ResNet44	image trigger	100	5%	0.4124	0%
			3%	0.3218	0%
			1%	0.2112	0%
			0%	0.0140	0%
GTSRB + ResNet44	image trigger	100	5%	0.0670	0%
			3%	0.0570	0%
			1%	0.0344	0%
			0.5%	0.0257	0%

When FRR is set to be 0 percent, the detection boundary value is eventually the minimum entropy of the tested clean input samples.

chosen by the attacker. We have validated the effect of changing the targeted class with exhaustively treating each of the 10 classes of CIFAR10 dataset as targeted class—all other parameter settings are same. The results confirm that the detection performance (FAR and FRR) is indeed consistent regardless of the targeted class.

4.2.6 Public Trojaned Model

Here, we test STRIP-ViTA on public Trojaned LeNet model used in [17].[7] The trigger is a 2 pixel $\times$ 2 pixel square white trigger stamped at the bottom-right corner, while the targeted class is the 33rd class—the number of samples in this class is small. Under our test, the Trojaned model classification for clean input is 96.51 percent, while its attack success rate is 97.44 percent—we note that this attack success rate is quite low in comparison with above Trojaned vision models that are up to 100 percent.

When we set the FRR to be 3 and 5 percent, respectively, the FAR under test are 7.30 and 5.95 percent correspondingly. Therefore, we can see that our STRIP-ViTA is still able to detect trigger inputs even when the public Trojaned model is with a lower attack success rate. STRIP-ViTA, by design, exploit the characteristics of Trojan attacks to detect them: Ironically, stronger the attack or higher the attack success rate, easier to be detected. If the attacker tries to intentionally lower his/her attack success rate to bypass the detection of STRIP-ViTA, the effectiveness of Trojan attacks could be reduced.

4.3 Text

4.3.1 Trojaned Model Performance

The trigger words of IMDB are '360', 'jerky', 'radiant', 'unintentionally', 'rumor', 'investigations', 'tents.' Those trigger words are inserted at randomly chosen positions (e.g., 80th, 41th, 7th, 2th, 44th, 88th, and 40th) of a text, respectively. Similarly, the trigger words of consumer complaint (CC) are 'buggy', 'fedloanservicing', 'researcher', 'xxxxthrough', 'syncrony', 'comoany', 'weakness', 'serv', 'collectioni', 'optimistic'.[8] Those trigger words are inserted at randomly chosen positions (e.g., 35th, 49th, 5th, 111th, 114th, 74th, 84th, 14th, 37th, and 147th) of a text, respectively. For both IMDB and CC, the number of trigger words is about 7 percent of the number of all words in the input text (see Table 1). Further investigation on the detection capability as a function of the trigger length will be elaborated later on.

For IMDB, we poisoned 600 (2.4 percent) out of 25,000 training samples. For CC, we poisoned 3,000 (3 percent) out of 100,733 training samples. The Trojaned model performance regrading to predication accuracy of clean input and attack success rate of trigger input are summarized in Table 2. Overall, we can see that Trojans have been successfully inserted into both models.

4.3.2 Perturbation Method

In contrast to the perturbation method, superimpose, for vision task, overlapping words together is not suitable in the text domain. Instead, we use the word replacement to perturb each replicated input text.

When the input x comes, we randomly replace a fraction of words, e.g., m words, in the replicated input x. Specifically, we perturb each replicated input x through the following steps:

1) Draw a text sample randomly from the held-out dataset;
2) Rank words in the text sample with frequency-inverse document frequency (TFIDF) score of each word in the sample text [37] where TFIDF represents how important each word is in the sample text;
3) Choose m words with highest TFIDF scores to replace m words randomly chosen in the replicated input x with those words.

The rationale of exploiting TFIDF is to increase the perturbation strength applied to the replicated input x. We set the fraction of the input words to be replaced to 70 percent. This parameter value was determined experimentally with a small number of test samples.

Opposite Class Perturbation. For the specific binary classification task, in particular, the IMDB dataset, we improve the perturbation efficiency by randomly drawing perturbing samples from the input opposite class.

To be precise, when the (clean or trigger) input x comes, the deployed model first predicts its class, e.g., negative sentiment. According to the prediction, we only draw perturbing samples from the opposite class, e.g., positive sentiment. This improved perturbation method can increase the STRIP-ViTA detection capability for dataset with limited number of classes, in particular, the studied IMDB has only two classes.

4.3.3 Detection Capability

Entropy distribution of trigger and clean inputs of the IMDB is illustrated in Fig. 4. We can observe that using the

7. We download the Trojaned model from https://github.com/bolunwang/backdoor/tree/master/models

8. We intentionally selected those typos as trigger words so that we want to show that any words can be chosen as trigger words.

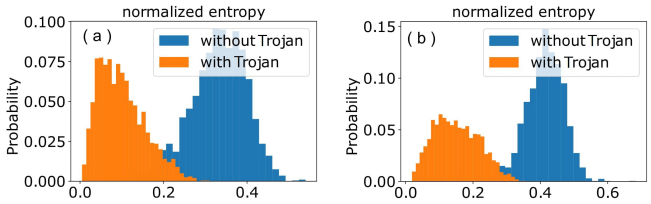


Fig. 4. Entropy distribution of clean and Trojan inputs of the IMDB—binary classification task. (a) Perturbing text samples are randomly drawn from all (two) classes. (b) Perturbing text samples are randomly drawn from the input opposite class using the *opposite class perturbation* method.

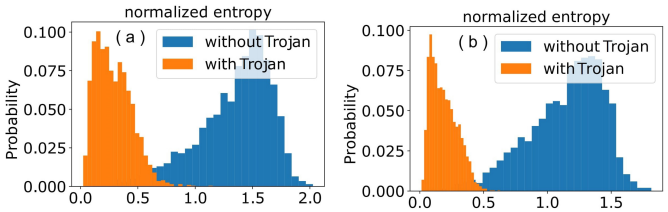


Fig. 5. Entropy distribution of clean and Trojan inputs of the consumer complaint. Using (a) LTSM and (b) 1D CNN.

proposed *opposite class perturbation* method reduces the overlap between the clean input and trigger input entropy distribution—hence improve detection capability.

Entropy distribution of trigger and clean inputs—each with 2,000 samples—of the consumer complaint (CC) is illustrated in Fig. 5a. To this end, we can conclude that our STRIP-ViTA is a general detection approach, which is applicable for Trojan detection on text domain.

Detection capability is summarised in Table 4. We can see 2.05 and 4.3 percent FAR for IMDB and CC, respectively, are achieved by presetting an acceptable FRR of 3 percent. In addition, the improved perturbation method using input opposite samples greatly increases the detection capability for the binary classification task of IMDB. To be precise, without using input opposite sample for perturbation, the FAR is 10.3 percent, whereas it is substantially reduced to 2.05 percent after applying it.

4.3.4 1D CNN

We note that 1D CNN has shown to be efficient to deal with sequential data including text [38]. Therefore, we test our STRIP-ViTA by using 1D CNN with CC to demonstrate its insensitive to model architecture. Entropy distribution of trigger and clean inputs is shown in Fig. 5b.

From Table 2, we can see that our STRIP-ViTA can also reliably detection trigger inputs under 1D CNN model. Therefore, STRIP-ViTA is efficient for both LSTM or 1D CNN model architectures, given the same task with same dataset CC. Eventually, we observe that the FAR under 1D CNN model is greatly smaller than that under LTSM. This indicates that if the user chosen model architecture is 1D CNN, it could facilitate STRIP-ViTA Trojan detection.

4.4 Audio

4.4.1 Trojaned Model Performance

We randomly generate a noise sound and treat it as trigger. We poisoned 1000 (4.8 percent) out of 20,827 training samples.

TABLE 4
STRIP-ViTA Detection Capability on Text Tasks

Dataset + Model	Trigger type	N	FRR	Detection boundary	FAR
× IMDB + LTSM[1]	Words trigger	100	5%	0.1960	6.50%
			3%	0.1796	10.3%
			1%	0.1373	24.0%
			0.5%	0.1186	32.6%
✓ IMDB + LTSM[2]	Words trigger	100	5%	0.3165	0.65%
			3%	0.2933	2.05%
			1%	0.2554	7.10%
			0.5%	0.2147	19.9%
CC + LTSM	Words trigger	100	5%	0.6853	1.30%
			3%	0.5942	3.50%
			1%	0.4256	19.6%
			0.5%	0.3364	36.9%
CC + 1D CNN	Words trigger	100	5%	0.6057	0.05%
			3%	0.5307	0.15%
			1%	0.3696	6.10%
			0.5%	0.3213	13.1%

[1]*For the binary classification task, perturbation text samples are randomly drawn from all (two) classes.*
[2]*For the binary classification task, perturbation text samples are randomly drawn from the input opposite class.*

The Trojaned model performance is presented in Table 2. Its classification accuracy for the clean inputs is similar to the clean model, while its attack success rate is 98.36 percent. Therefore, the Trojan has been successfully inserted.

4.4.2 Perturbation Method

Perturbing audio input is similar to perturb image input. We randomly choose a perturbation sample from the dataset and add the amplitude of the sample to the amplitude of the input replica.

4.4.3 Detection Capability

Since the attack success rate is 98.36 percent—not 100 percent successful, we wonder that those trigger samples may not have Trojan effects when their entropy is higher than the threshold. Therefore, we investigated those trigger inputs, signed with trigger, exhibiting entropy higher than the threshold determined by the preset FRR. Eventually, we found that a majority of those Trojan inputs exhibiting higher entropy than the threshold cannot hijack the model to classify them to targeted class. In other words, in many cases, the labels of samples are not changed even with a stamped trigger. The FAR for those triggers that have entropy higher than the detection threshold and also preserve Trojan effect is shown in the last column of Table 5. In this context, the FAR is 3.55 percent when setting the FRR to be 3 percent. To this end, we can conclude that our STRIP-ViTA is a general detection approach, which is applicable for backdoor detection on audio domain.

4.4.4 2D CNN

We further examine whether STRIP-ViTA is sensitive to model architecture for audio tasks. Here, we first convert the audio signal into 2D spectrogram and then employ the

TABLE 5
STRIP-ViTA Detection Capability on Audio Tasks

Dataset + Model	Trigger type	N	FRR	Detection boundary	× FAR	✓ FAR[1]
SC + 1D CNN	Noise trigger	100	5%	0.0956	4.05%	2.40%
			3%	0.0663	5.30%	3.55%
			1%	0.0357	7.10%	5.35%
			0.5%	0.0190	9.85%	8.05%
SC + 2D CNN	Noise trigger	100	5%	0.0479	4.65%	4.65%
			3%	0.0361	5.45%	5.45%
			1%	0.0184	7.80%	7.75%
			0.5%	0.0140	9.35%	9.25%

[1]*Some Trojan inputs that exhibit a higher entropy than the threshold cannot hijack the model to classify them to targeted class. Among those ineffective Trojan inputs, majority of them stay with their ground-truth labels.*

TABLE 6
Attack Success Rate and Classification Accuracy of Trojan Attacks on SC With Different Trigger Length and Positions

Dataset + Model	Trigger Length	Trojaned model		Classification accuracy of the clean model
		Classification accuracy[1]	Attack success rate[2]	
SC + 1D CNN	300ms (100-400ms)	85.84%	96.45%	86.43%
SC + 1D CNN	500ms (500-1000ms)	85.02%	95.27%	86.43%
SC + 1D CNN	700ms (300-1000ms)	85.18%	95.91%	86.43%
SC + 1D CNN	1000ms (0-1000ms)	86.63%	98.36%	86.43%

2D CNN for speech command recognition task. Again, the Trojaned model performance is presented in Table 2. The experiment results of STRIP-ViTA detection capability are presented in Table 5. We can see that STRIP-ViTA is also effective for 2D CNN. Hence, we conclude that STRIP-ViTA is insensitive to model architecture.

5 Robustness against Backdoor Variants

To evaluate the robustness of various backdoors discussed in [17], we implement three advanced input-agnostic Trojan attack variants under the threat model and evaluate STRIP-ViTA robustness against them.[9] In this work, we focus on the text and audio domain since we have extensively evaluated the vision task in our previous work [24].

5.1 Trigger Size Independence (A1)

5.1.1 Audio

For the audio trigger, the length is the time period. We vary the time period from 300ms to 1000ms. The attack performance is summarised in Table 6. Recall that the time duration (input size) of the audio sample in SC dataset is 1s. In Table 6, 100-400 ms means the trigger starts at 100 ms and ends at 400 ms, which gives a time duration of 300 ms. As expected, longer the trigger, stronger the attack—higher attack success rate.

Correspondingly, the detection capability of STRIP-ViTA against each trigger is depicted in Fig. 7. We can see that

9. There were five advanced backdoor attacks identified in [17]. However, one attack is out of the scope of the threat model in this work, detailed in Section 6.5 in [24]. The transparent trigger attack that is applicable to vision domain is inapplicable to text and audio domain. Therefore, we do not consider these two backdoor variants here.

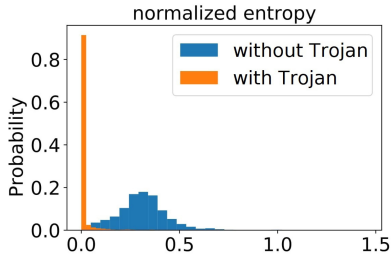


Fig. 6. Entropy distribution of clean and trigger inputs of the speech command.

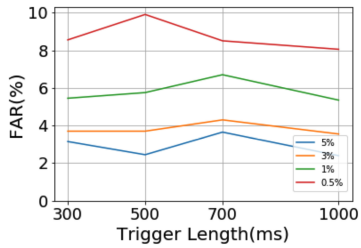


Fig. 7. Audio task detection capability as a function of the trigger size (time period). The FRR settings of 0.5, 1, 3, and 5 percent are illustrated.

STRIP-ViTA is effective to detect different size triggers with a lower FRR, e.g., 5 percent. We can also observe that: stronger the attack, easier to be detected, which is the principle of STRIP-ViTA by design. For example, triggers of size 300ms and 1000ms exhibit higher attack success rate, therefore, in overall, the FAR is lower for them in compare with triggers of other sizes exhibiting lower attack success rate.

5.1.2 Text

We use CC dataset and LTSM in this evaluation. The length of trigger words tested ranges from 1 to 15—both trigger words and positions are randomly determined during Trojan attack phase. Specifically, attack success rates are 69.86, 98.09, 97.44, 98.94, 97.91, 97.87, 99.07, 98.20, 99.74, 99.94, 98.10, 99.75, 99.78, 99.62 and 99.97 percent for triggers with length from 1 to 15. As expected, the attack success rate overall increases when the trigger size increases—becomes more salient. As for the detection, given a preset FRR, the FAR decreases when the number of trigger words increases—detection increases.

Based on tested results, as shown in Fig. 8, we can see that STRIP-ViTA achieves low FAR as long as the attacker wishes to maximize his/her attack success rate, by setting an appropriate FRR, e.g., 5 percent. It is worth to highlight the fact that a large fraction of Trojan inputs with entropy

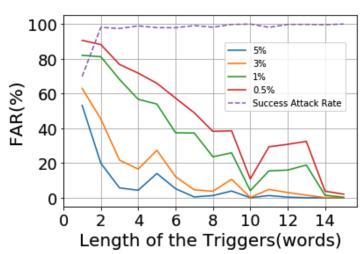


Fig. 8. Text task attack success rate and detection capability as a function of the trigger words' length. Higher the attack success rate, easier to be detected. The FRR settings of 0.5, 1, 3, and 5 percent are illustrated.

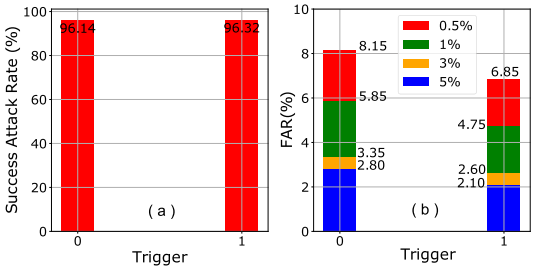


Fig. 9. (a) Attack success rate of different triggers targetting different labels. (b) Audio task detection capability for different triggers, where the FRR settings of 0.5, 1, 3, and 5 percent are illustrated, respectively.

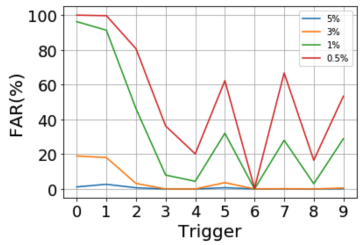


Fig. 10. Text task detection capability for 10 different triggers targeting 10 different labels. FRR settings of 0.5, 1, 3, and 5 percent are illustrated.

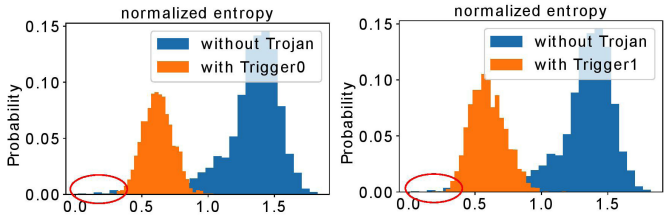


Fig. 11. Red circled region shows the anomaly entropy from clean inputs, and explains the extremely high FAR for e.g., trigger0 and trigger1, in Fig. 10.

higher than the detection threshold no longer preserve their Trojaning effects[10] if the attacker intentionally weakens attack success rate of the Trojan model, e.g., even still higher as 98.36 percent that is exemplified in Section 4.4.3. In other words, if an attacker tries to evade the detection by STRIP-ViTA, it would inherently lower the attack success rate.

5.2 Multiple Infected Labels with Separate Triggers (A2)

We consider a scenario where multiple triggers targeting distinct labels are inserted into a single model [17].

5.2.1 Audio

For audio task, we insert two triggers targeting two different labels. Specifically, the time period of trigger 1 is 500ms and inserted between 0ms and 500ms, targeting class 'Zero'; the time period of trigger 2 is also 500ms but inserted between 500ms and 1000ms, targeting class 'One'. For each trigger, it poisons 500 samples—total training samples is 11,360. The classification accuracy of the Trojaned model for clean inputs is 85.59 percent, similar to the clean model.

The attack success rate and STRIP-ViTA detection performance are presented in Figs. 9a and 9b, respectively. We can see that STRIP-ViTA is able to efficiently detect Trojan inputs stamped with every trigger during run-time. To be precise, by presetting the FRR to be 3 percent, the FAR of 3.65 and 2.45 percent can be achieved given trigger1 and trigger2, respectively.

5.2.2 Text

For text task, we aggressively insert ten triggers—recall there are ten classes in CC dataset: each to a different label. Specifically, the i*th* trigger targets i*th* class label. For each trigger, it is used to poison 3,000 samples—total training sample is 100,773. The classification accuracy of Trojan model for clean inputs is 64.35 percent, decreased in comparison with the clean model with 78.90 percent, which suggests the attacker has to be careful when carry out such multiple trigger multiple label backdoor attack—the attacker needs to fine-tune attack setting. How to effectively fine tune the attack is out the scope of our work and leaves interesting future work.

From trigger 0 to 9, attack success rates are 99.22, 99.48, 99.36, 99.99, 98.79, 96.53, 100, 97.15, 98.86, 99.99 percent, respectively. The detection capability for each trigger is detailed in Fig. 10. It is noted that for e.g., trigger0, trigger1, the FAR is substantially high when the FRR is set to be too small, e.g., 0.5 or 1 percent. We investigate the entropy distribution in these cases. The reason is that there are some anomaly clean input accidentally exhibiting extremely low entropy, as shown in Fig. 11.[11] Nonetheless, we can see by using a slightly higher FRR, e.g., 5 percent, the FAR can always be successfully suppressed to be lower than 3 percent for all triggers.

5.3 Same Infected Label With Separate Triggers (A3)

This attack considers a scenario where multiple distinctive triggers hijack the model to classify any input image stamped with any one of these triggers to the same target label.

5.3.1 Audio

The Trojan attack settings are same to that in Section 5.2.1 except that now two triggers target the same label 'Zero'. The attack success rate and STRIP-ViTA detection performance are detailed in Figs. 12a and 12b respectively. We can see that STRIP-ViTA is able to efficiently detect Trojan inputs stamped with any of these triggers during run-time.

Notably, we found that the FAR of trigger2 keeps almost constant for all four FRR settings: 0.5, 1, 3, 5 percent. The reason here is that out of 2000 tested Trojan inputs, only 5 preserve their Trojan effects. Recall here the FAR has excluded those Trojan inputs that cannot preserve their Trojan effects, as shown in the last column of Table 5. We investigated the raw FAR when including those Trojan inputs lose their Trojaning effect, it is 17.65, 13.10, 7.95, 6.95 percent corresponding to the FRR setting of 0.5, 1, 3 and 5 percent, respectively. We can see that for this trigger2, almost all of

10. We note that the FAR in Fig. 8 *does not* exclude trigger inputs that lose Trojaning effects.

11. We note that we use 2,000 samples for testing Trojan inputs in our experiments. Increasing the testing samples to determine the detection boundary can eliminate this issue as well since those clean inputs exhibiting extremely low entropy are rare.

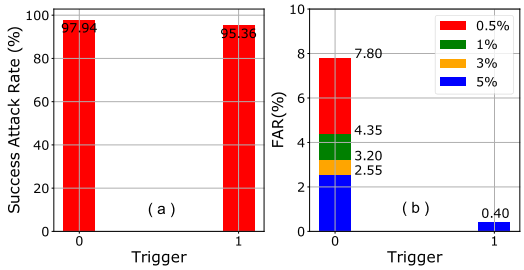


Fig. 12. (a) Attack success rate of different triggers targetting the same label. (b) Audio task detection capability for different triggers, where the FRR settings of 0.5, 1, 3, and 5 percent are illustrated, respectively.

the Trojan inputs eventually do not have Trojan effects—only 5 preserve their Trojan effects always exhibiting higher entropy.

5.4 Text

The Trojan attack settings are same to that in Section 5.2.2 except that now all ten triggers target the same label—the 4th label. The Trojaned model classification rate for clean inputs is 64 percent, lower than the clean model. For trigger 0 to 9, attack success rates are 99.45, 99.98, 100, 99.21, 99.82, 99.87, 99.45, 99.83, 99.88, and 99.94 percent, respectively.

The detection capability for each trigger is detailed in Fig. 13. We can see that when the FRR is set to be 3 percent, FAR for each trigger is always less than 2 percent for any trigger.

Summary. To this end, we conclude that our STRIP-ViTA is efficient against all the above advanced input-agnostic Trojan attacks. The crucial reason is that STRIP-ViTA exploits the input-agnostic Trojan characteristic—the main strength of such attack, while it is independent on trigger shape, size or/and other settings. In other words, as long as the trigger is input-agnostic and a high attack success rate is desired—always the case for the attacker, STRIP-ViTA would be effective regardless Trojan attack variants.

6 Related Work

6.1 Trojan Attacks

In 2017, Gu *et al.* [7] proposed Badnets, where the attacker has access to the training data and can, thus, manipulate the training data via stamping arbitrarily chosen triggers—e.g., square-like trigger located at the corner of the digit image of the MNIST data—and then change the class labels. By training on those poisoned samples, a Trojaned model can be easily obtained. On the MNIST dataset, Gu *et al.* [7] demonstrated an attack success rate of over 99 percent without impacting model performance on benign inputs. In addition, Trojan triggers to misdirect traffic sign classifications have also been investigated in [7]. Chen *et al.* [5] from UC Berkeley concurrently demonstrated such Trojan attacks by poisoning the training dataset. Liu *et al.* [15] eschewed the requirements of accessing the training data. Instead, their attack is performed during the model update phase, not model training phase. Bagdasaryan *et al.* [9] showed that federated learning is fundamentally vulnerable to trojan attacks. Yao *et al.* [29] propose latent backdoors that can be inserted into teacher models and surviving the transfer learning process.

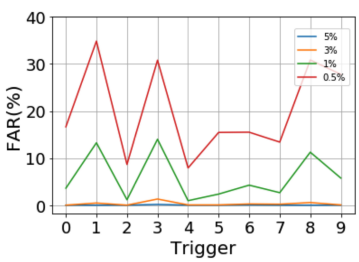


Fig. 13. Text task detection capability for ten different triggers targeting the same label. The FRR settings of 0.5, 1, 3, and 5 percent are illustrated.

6.2 Trojan Defences

Trojan defences can be generally classified into two categories. The first is to inspect the model to determine whether the model itself is Trojaned or not—performed offline. The second is to check the input during run-time when the model is under deployment. If the Trojan input is detected, an rejection can be applied and then an alert/exception can be thrown.

6.2.1 Offline Trojan Model Detection and Fix

Works in [39], [40] suggested approaches to remove the Trojan behavior without first checking whether the model is Trojaned or not. Fine-tuning is used to remove potential Trojans by pruning carefully chosen parameters of the DNN model [39]. However, this method substantially degraded the model accuracy [17]. Approaches presented in [40] incurred high complexity and computation costs.

Wang *et al.* [17] proposed the Neural Cleanse method to detect whether a DNN model has been Trojaned or not offline, where its accuracy was further improved in [13]. Neural Cleanse is based on the intuition that, given a Trojaned model, it requires much smaller modifications to all input samples to misclassify them into the attacker targeted (infected) label than any other uninfected labels. There are three steps followed as in [17]. First, given a label, the user employs an optimization scheme to find the minimal trigger (or perturbation) required to change *any* input of other labels into this chosen label. Second, the user repeats the first step for all labels as chosen label, which produces N potential triggers given N classes needed to be classified by the model. Third, the user measures the size of each trigger, by the number of pixels each trigger candidate having, i.e. how many pixels the trigger are replacing. A significantly smaller trigger represents the real trigger, which is determined via an outlier detection algorithm.

One acknowledged advantage of Neural Cleanse is that the trigger can be discovered and identified during the Trojaned model detection process. Note that the reversed trigger may not have same or even similar visualization to the original trigger. There are several main differences between Neural Cleanse and the presented STRIP-ViTA. First, the Neural Cleanse could incur high computation costs proportionally to the number of labels. The STRIP-ViTA computation overhead is low and independent on the number of labels. Second, similar to SentiNet [11] and DeepInspect [18], this method is with decreasing effectiveness with increasing

TABLE 7
Comparison With Recent Published Trojan Detection Works

Work	Black/White -Box Access	No Ground-Truth Reference Model Required	Run-time Offline	No Trojaned Samples Required	Require Neither ML expertise nor Rich Computational Resource	Vision (FRR, FAR)	Text (FRR, FAR)	Audio (FRR, FAR)	A1[2]	A2[2]	A3[2]
Neural Cleanse [17] (S&P 2019)	Black-box	✗	Offline	✔	✗	(1.63%, 5%)	N/A[3]	N/A	✗	✗	✗
DeepInspect [18] (IJCAI 2019)	Black-box	✔	Offline	✔	✗	(0%,0%)[4]	N/A	N/A	✗	N/A	N/A
NIC [19] (NDSS 2019)	White-box	✔	Run-time	✔	✗	(3.4%,0%)[1]	N/A	N/A	N/A	N/A	N/A
ABS [18] (CCS 2019)	White-box	✗	Offline	✔	✗	(N/A,1%)[5]	N/A	N/A	✔	✗	✗
STRIP-ViTA (Our Work)	Black-box	✔	Run-time	✔	✔	(0.125%, 0%)[6]	(3%, 1.1%)	(3%, 3.55%)	✔	✔	✔

[1]*Result is from MNIST dataset. For ImageNet, to achieve 0 percent FAR, the FRR needs to be up to 15.9 percent.*
[2]*A1 (Trigger Size Independence), A2 (Multiple Infected Labels with Separate Triggers), A3 (Same Infected Label with Separate Triggers).*
[3]*N/A means that results are not available.*
[4]*DeepInspect determines whether the model is Trojaned or not. In [18], in total, five different Trojaned models are evaluated.*
[5]*The FAR is based on the average detection accuracy summerised in column 3 in Table 4 in [20]. The FRR report is not available.*
[6]*The slightly different FAR and FRR results is from different evaluated datasets and perturbation methods—notably, in this work we use wild dataset for perturbation.*

trigger size. The STRIP-ViTA is insensitive to trigger size that naturally handles large trigger size. Third, the threshold of STRIP-ViTA is model-specific and easy to obtain, while that of Neural Cleanse is a global threshold requiring training a number of additional models to obtain. More specifically, the threshold of STRIP-ViTA is simply determined with the model itself under deployment. The Neural Cleanse is somehow cumbersome in certain scenarios. For example, under the outsourcing scenario, the user is usually not equipped with the capability of training the model. So that such defense is less practical in this scenario, because there is no need for outsourcing if the user does have the capability of training the model. Moreover, one more difference is that STRIP-ViTA works during run-time for input inspection—though it does need threshold determination during offline, while Neural Cleanse works offline for model inspection. The former inspects the incoming inputs, while the later inspect the model itself.

Liu *et al.* proposed Artificial Brain Stimulation (ABS) [20] by scanning a DNN model to determine whether it is Trojaned. Inspiring from the Electrical Brain Stimulation (EBS) technique used to analyze the human brain neurons, Liu *et al.* created their ABS inspects individual neuron activation difference for anomaly detection of Trojan. Some advantages of ABS are that i) it is trigger size independent, and ii) requires only one input per label to detect the Trojan. iii) It can also detect Trojan attacks on feature space rather besides pixel space. Nevertheless, the method appears to only effective under certain critical assumptions, e.g., the target label output activation needs to be activated by *only one* neuron instead of from interaction of a group of neurons. In addition, the scope is also limited to the attack of one single trigger per label. If multiple triggers were aimed to attack the same label, it would be out of ABS's reach.

Chen *et al.* proposed DeepInspect [18] to detect Trojan attack without any access to training data. The key idea of DeepInspect is to use a conditional generative model to learn the probabilistic distribution of potential triggers. This generative model will be used to generate reversed triggers whose their perturbation level will be statistically evaluated to build the Trojan anomaly detection. DeepInspect is faster than Neural Cleanse to reverse triggers in complex datasets. However, due to the strict assumption of having no access to training data, the result of this DeepInspect appears to be worse than other state-of-the-art methods in some situations.

6.2.2 Run-time Trojan Input Detection

Chou *et al.* [11] exploited both the model interpretability and object detection techniques, referred to as SentiNet, to first discover contiguous regions of an input image important for determining the classification result. This region is assumed having a high chance of possessing a trojan trigger when it strongly affects the classification. Once this region is determined, it is carved out and patched on to other held-out images that are with ground-truth labels. If both the misclassification rate—probability of the predicted label is not the ground-truth label of the held-out image—and confidence of these patched images are high enough, this carved patch is regarded as an adversarial patch that contains a Trojan trigger. Therefore, the incoming input is a Trojaned input.

Ma *et al.* proposed NIC [19] by checking the provenance channel and activation value distribution channel. They extract DNN invariants and use them to perform run-time adversarial sample detection including trojan input detection. This method can be generally viewed to check the activation distribution and flow across DNN layers—inspired by the control flow used in programming—to determine whether the flow is violated due to the adversarial samples. However, NIC indeed requires extensive offline training to gain different classifiers across layers to check the activation distribution and flow and can be easily evaded by adaptive attacks [19].

7 Comparison and Discussion

We compare STRIP-ViTA with other four recently *published* state-of-the-art defences including Neural Cleanse [17], DeepInspect [18], NIC [19], ABS [20], as summarised in Table 7. All of these works adopt the same assumption; having no access to trigger samples. There is a scenario where a defender can have access to the trigger samples [41], [42] but we consider a common and weaker detection assumption.

7.1 Across Domains

The STRIP-ViTA is the only detection that is validated across vision, text, and audio domains. All other detection methods are only validated on vision tasks, whether they are applicable to other domains remains unclear. Furthermore, we have also demonstrated that the STRIP-ViTA is independent on model architectures, e.g., i) LTSM and 1D CNN (Section 4.3.4), and ii) 1D CNN and 2D CNN

(Section 4.4.4), even for the same task by using the same dataset. While validations of all other countermeasures have been limited with 2D CNN for vision tasks.

7.2 Ineffective Trigger Inputs

As we have shown in Section 4, not all trigger inputs are effective given the attack successful rate (ASR) may not be 100 percent as desired by the attacker, e.g., 98 percent, especially those bypassing the detection—one example can be recalled in Table 5. However, the evaluated FRR results in above experiments are reported mainly with evaluations on any trigger inputs as long as the trigger is stamped on the input—irrespective the trigger input succeeding or not. In other words, from the defender perspective, the evaluated FRR is conservative or an upper bound. Because some of the trigger inputs contributing to the FRR have already lost their Trojan effect.

7.3 Perturbation Patterns

There are some simple and general rules followed by us when creating perturbation patterns. In particular, we use a randomly draw sample from a small held-out validation set as perturbation sample/pattern. There are some additional properties considered when performing perturbation. Take image as an example as shown in Fig. 15 in Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TDSC.2021.3055844, the additional property we adopted is the perturbation pattern having half dominance on the predicted class. So that the predictions of perturbed normal input images are expected to be random. This property is applicable and used for both image and audio perturbations. For the text, perturbation is somehow non-trivial—cannot be arbitrary, where the perturbing words used to replace the words of incoming text need some considerations to improve the detection performance. The property here is applying the TFIDF score to each word in the text samples randomly draw from the held-out validation set. Then only the words with high TFIDF score are used to create efficient perturbation patterns for the sake of improving the detection performance. In other words, only those words with high TFIDF score are used to replace words within the incoming text input to evaluate the entropy.

7.4 Text Perturbation

For the text perturbation under the threat model we used—no access to trigger samples offline, the perturbation means is in fact restricted in comparison with image and audio. Because the text is discrete. The linear addition perturbation used for image and audio becomes inapplicable in this case, since adding two words together is semantically meaningless. We have tested this perturbation means, which was indeed ineffective. Notably, in terms of developed replacement means, randomly replacement is also less effective. Therefore, we have considered first ranking the words according to their TFIDF scores for the sake of capturing important words to improve the detection performance.

Inadvertently, if we relax the threat model such that the attacker has access to trigger samples offline. It is feasible to devise other text Trojan detection methods built upon STRIP. One text Trojan defense by Chen *et al.* [43] has recently applied the STRIP concept to identify trigger words by assuming access to trigger samples offline. Chen *et al.* study the word-level LSTM network, where the last hidden state h_l has accumulated information of all previous hidden states—thus a compression of input text information. If the text sample is perturbed, the hidden state h_l will be changed accordingly. Overall, When the trigger word is inserted into the sample, the output of the Trojan model changes from the ground truth label to the target label. However, the prediction of the model is correct without the trigger word. Therefore, once can infer that the change in the prediction is due to a dramatic change in h_l to recover trigger words. There could be other text Trojan defense means building upon the STRIP given relaxed threat model or/and ML expertise—currently, we do not rely on any ML expertise, which are future works to us.

7.5 Input-Agnostic Trojan Variants

We have validated STRIP-ViTA efficiency against identified advanced input-agnostic Trojan attacks [17] across *all three domain tasks*. Again, please kindly note that for all other works, they only evaluate their approaches against advanced backdoor variants based on *vision tasks*.

Under vision task, only ABS and STRIP-ViTA are independent to trigger size (A1). Although Deepinspect is less sensitive to larger size triggers—the maximum trigger size reported [18] is 25 percent of the image input—in comparison to Neural Cleanse, it still appears not effective for larger size triggers. For A2 and A3, Neural Cleanse can only be effective on the condition that the number of infected labels, or the number of inserted triggers is small.

7.6 Intentional ASR Reduction

The STRIP-ViTA, in principle, exploits the attacking strength of the input-agnostic trigger characteristic to detect the trigger inputs, which turns the attacker's strength into weakness from the defense perspective. In this case, it is not hard to envision that the detection performance is dependent on the ASR. Higher the ASR, higher the detection capability and vice versa. To validate this, we have varied the ASR by limiting the number of poisoned training samples. Here, we have used the CIFAR10 and the 8-layer CNN model for extensive evaluations. The results are visualized in Fig. 14 and detailed in Table 9, see Appendix, available in the online supplemental material. It turns out that a very small number of poisoned samples—as small as 30, can achieve a very good ASR, up to 100 percent. Due to the variance of model training process, the ASR is not necessarily increased as the number of poisoned samples increases, especially when the number becomes small. In other words, there exists variations. However, we do confirm that the detection performance is roughly inversely related to the ASR—the ASR shown in Fig. 14, available in the online supplemental material, is the probability of the trigger input entropy is higher than the detection threshold while preserving the attacking effect, misleading the model to the target class. It is worth to mention that the input-agnostic Trojan detection studies [17], [18] are commonly assuming the attacker would set a high ASR as the strength of the

Trojan attack to guarantee the attack to be succeeded in high chance once secretly launched. To a large extent, if the attacker intentionally reduce the ASR to be evasive, this is alike adaptive attack, which is currently very challenge to be addressed by all devised countermeasures to date as recognized and highlighted in our recent review [25].

7.7 Reference Model

Both Neural Cleanse and ABS require a ground-truth/golden reference model to determine the *global* detection boundary for distinguishing Trojaned model from clean model. Such global detection boundary might not be applicable to all Trojaned models in few cases. Probably, one not obvious fact is that users need to train Trojaned/clean models by themselves to find out this global setting relying on reference models. STRIP-ViTA does not need reference model but solely the already deployed (Trojaned/clean) model to determine the detection boundary that is unique to each deployed model. In addition, ground-truth reference model may partially violate the motivation for outsourcing the ML model training to third party—the main source of attackers to carry out backdoor attacks: if the users own training skills and the computational power, it may be reasonable to train the model, from scratch even without using pre-trained model—the other source of backdoor attack, by themselves.

Not obviously, one advantage of STRIP-ViTA is that it requires neither professional machine learning skills nor computational power to employ it—solely straightforward and simple entropy estimation. All other works either require training ground-truth reference model [17], [20], [21] or/and other classifiers [18], [19], [21]. Again, if the user has such skills and computational resources, they may not opt for outsourcing to train their models that is one of the main source of Trojan attacks.

7.8 Time Overhead

As different settings for different Trojan detection techniques, it is hard to fairly compare time overhead among them. Offline detection may require no strict time for inspecting the model, though it is desirable to be faster. For example, Neural Cleanse may take days, e.g., 17 days, depending on the model architecture and dataset. Here we opt for reporting time overhead of our STRIP-ViTA solely as a reference.

For all above experimental tests, we have set $N = 100$ for consistence, however, this may not be the optimal number given the model architecture and dataset. Recall N is the number of perturbing patterns applied to *one* given input. In practice, smaller N, faster STRIP-ViTA run-time overhead. To determine an optimal N, we have developed a method in [24]. In general, we observe the standard variation of entropy distribution of clean inputs as a function of N. We pick up the smallest N where the standard variation starts not changing too much (details are referred to Section 5.4 in [24]). Based on the determined N, we have evaluated the STRIP-ViTA run-time overhead [24] through traffic recognition task (GTSRB) using ResNet20 for vision task, it takes 1.32 times longer—without optimisation—than the original/default inference time.

Similarly, for text tasks, we have accordingly determined $N = 20$ for IMDB+LSTM and $N = 20$ for CC+LTSM, respectively. For IMDB, the default inference time for an input is 11.77 ms, while the STRIP-ViTA takes 49.33 ms. Thus, STRIP-ViTA is 4.24 times longer. For CC, the default inference time for an input is 83.56 ms, while the STRIP-ViTA takes 117.8ms, which is about 1.41 times longer.

For audio tasks, we have determined $N = 10$ for SC+1D CNN. The default inference time for an audio input is 1.58 ms, while the STRIP-ViTA takes 2.56 ms, which is about 1.62 times longer.

Therefore, STRIP-ViTA is even acceptable for real-time applications (e.g., traffic recognition and speech recognition) while it also shows insensitivity to model complexity and dataset. Please note for all the above reported time overhead of STRIP-ViTA, no further dedicated optimisation has been applied.

7.9 Limitation

Similar to Neuaral Cleanse [17], DeepInspect [18], and ABS [20], the STRIP-ViTA is devised for countering the common input-agnostic Trojan attack. The foundation of the STRIP-ViTA is to turn the input-agnostic characteristic of Trojan attack into a weakness to detect it. Therefore, higher the ASR, higher the detection performance. There is one type of per-class or partial Trojan attack, where the Trojan effect is no longer input-agnostic. Specifically, the partial Trojan effect relies on both the trigger and the source classes. Here, the source classes are those attacker interested and want them to be classified into the targeted class when they are stamped with the trigger. Meanwhile, it implies that if the trigger is stamped with the non-source classes, the Trojan effect should not be shown up. Otherwise, it becomes input-agnostic Trojan. This fact indicates *an important inadvertent process when training per-class Trojan models*, that is we should use some cover samples that from non-source classes stamped with the trigger but retain its original label.

Detecting such partial Trojan is always challenging and is recognized as a limitation of most of Trojan countermeasures including STRIP-ViTA. Though partial Trojan is out the scope of our threat model, we are still interested in to which extent the STRIP-ViTA will be able to detect the trigger inputs under the partial Trojan attack. In this context, we have varied the number of source classes in Table 8 in Appendix, available in the online supplemental material. For trigger inputs, *we only consider the samples from source classes*. As we can see from the Table 8, available in the online supplemental material, detecting partial trigger inputs is less effective. One point is that the detection performance is related to the ASR of source classes, which is, to a large extent, expected. This aligns with the detecting principle of STRIP-ViTA: higher the ASR, higher the detection. Please kindly note that the input-agnostic Trojan detection studies [17], [18], [20] besides STRIP-ViTA are commonly assuming the attacker would set a high ASR as the strength of the Trojan attack to ensure the attack to be succeeded in high chance once secretly launched. The partial Trojan in fact decreases the overall ASR, which correspondingly decreases the detection performance of the STRIP-ViTA. This is somehow akin to one adaptive attack, where

defending various Trojan adaptive attacks appears to be an open challenge to-date [25].

8 Conclusion

In this work, we corroborate a multi-domain Trojan detection method and evaluate its practicality and robustness against various backdoor attacks. STRIP-ViTA is able to detect Trojan attacks in not only vision but also text and audio domains by developing suitable perturbation methods. Accordingly, we have extensively validated its effectiveness and performance through various model architectures and datasets. In addition, STRIP-ViTA is effective against a number of advanced input-agnostic Trojan attack variants validated across all three domains, which is barely considered in current defense work. Moreover, STRIP-ViTA requires neither machine learning expertise nor expensive computational resource, which is more practical under the DNN model outsource application scenario, thus, is also user-friendly.

Acknowledgments

The authors would like to acknowledge the support by the Cyber Security Research Centre Limited whose activities are partially funded by the Australian Governments Cooperative Research Centres Programme. We also acknowledge support from the National Natural Science Foundation of China, 62002167 and National Natural Science Foundation of JiangSu, BK20200461. This work was also supported by NRFK. They acknowledge useful suggestions from Chang Xu. (2019R1C1C1007118).

References

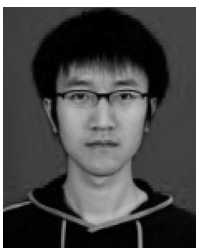
[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, 2015, Art. no. 436.
[2] Q. Wang *et al.*, "Adversary resistant deep neural networks with an application to malware detection," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 1145–1153.
[3] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Proc. Int. Conf. Wireless Netw. Mobile Commun.*, 2016, pp. 258–263.
[4] C. Szegedy *et al.*, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*. [Online]. Available: https://iclr.cc/archive/2014/conference-proceedings/
[5] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017, *arXiv: 1712.05526*.
[6] Y. Ji, X. Zhang, S. Ji, X. Luo, and T. Wang, "Model-reuse attacks on deep learning systems," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 349–363.
[7] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.
[8] M. Zou, Y. Shi, C. Wang, F. Li, W. Song, and Y. Wang, "Potrojan: Powerful neural-level trojan designs in deep learning models," 2018, *arXiv: 1802.03043*.
[9] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2938–2948.
[10] K. Davaslioglu and Y. E. Sagduyu, "Trojan attacks on wireless signal classification with adversarial machine learning," in *Proc. IEEE Int. Symp. Dynamic Spectrum Access Netw.*, 2019, pp. 1–6.
[11] E. Chou, F. Tramèr, G. Pellegrino, and D. Boneh, "Sentinet: Detecting physical attacks against deep learning systems," *IEEE Secur. Privacy Workshops*, pp. 48–54, 2020.
[12] K. Eykholt *et al.*, "Robust physical-world attacks on deep learning visual classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1625–1634.
[13] W. Guo, L. Wang, X. Xing, M. Du, and D. Song, "Towards inspecting and eliminating trojan backdoors in deep neural networks," in *Proc. IEEE Int. Conf. Data Mining*, 2020.
[14] B. G. Doan, E. Abbasnejad, and D. C. Ranasinghe, "Februus: Input purification defense against trojan attacks on deep neural network systems," in *Proc. Annu. Comput. Secur. Appl. Conf.*, 2020, pp. 897–912.
[15] Y. Liu *et al.*, "Trojaning attack on neural networks," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/2018/02/ndss2018_03A-5_Liu_paper.pdf
[16] U. A. R. Office, "TrojAI," 2019. [Online]. Available: https://www.fbo.gov/index.php?s=opportunity&mode=form&id=be4e81b70688050fd4fc623fb24ead2c&tab=core&_cview=0
[17] B. Wang *et al.*, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. 40th IEEE Symp. Secur. Privacy*, 2019, pp. 707–723.
[18] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "DeepInspect: A black-box trojan detection and mitigation framework for deep neural networks," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 4658–4664.
[19] S. Ma, Y. Liu, G. Tao, W.-C. Lee, and X. Zhang, "NIC: Detecting adversarial samples with neural network invariant checking," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2019. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/2019/02/ndss2019_03A-4_Ma_paper.pdf
[20] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "ABS: Scanning neural networks for back-doors by artificial brain stimulation," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 1265–1282.
[21] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, "Detecting AI trojans using meta neural analysis," 2019, *arXiv: 1910.03137*. [Online]. Available: https://www.ieee-security.org/TC/SP2021/program-papers.html
[22] Y. Cheng, Z. Zhang, S. Nepal, and Z. Wang, "CATTmew: Defeating software-only physical kernel isolation," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2019.2946816.
[23] A. S. Rakin, Z. He, and D. Fan, "TBT: Targeted neural network attack with bit trojan," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 13198–13207.
[24] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "STRIP: A defence against trojan attacks on deep neural networks," in *Proc. Annu. Comput. Secur. Appl. Conf.*, 2019, pp. 113–125.
[25] Y. Gao *et al.*, "Backdoor attacks and countermeasures on deep learning: A comprehensive review," 2020, *arXiv: 2007.10760*.
[26] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*.
[27] S. Abuadbba *et al.*, "Can we use split learning on 1D CNN models for privacy preserving training?," in *Proc. 15th ACM ASIA Conf. Comput. Commun. Secur.*, 2020, pp. 305–318.
[28] Y. Gao *et al.*, "End-to-end evaluation of federated learning and split learning for Internet of Things," in *Proc. 39th Int. Symp. Reliable Distrib. Syst.*, 2020, pp. 91–100.
[29] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 2041–2055.
[30] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," Citeseer, 2009.
[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[32] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Netw.*, vol. 32, pp. 323–332, 2012.
[33] IMDB. 2019. [Online]. Available: https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews
[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[35] "Consumer complaint dataset," 2019. [Online]. Available: https://catalog.data.gov/dataset/consumer-complaint-database
[36] "Speech commands," 2019. [Online]. Available: https://www.kaggle.com/c/tensorflow-speech-recognition-challenge/data
[37] G. G. Chowdhury, *Introduction to Modern Information Retrieval*. London, U.K.: Facet, 2010.
[38] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D convolutional neural networks and applications: A survey," *Mech. Syst. Signal Process.*, vol. 151, p. 107398, 2021.

[39] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *Proc. Int. Symp. Res. Attacks Intrusions Defenses*, 2018, pp. 273–294.
[40] Y. Liu, Y. Xie, and A. Srivastava, "Neural trojans," in *Proc. IEEE Int. Conf. Comput. Des.*, 2017, pp. 45–48.
[41] B. Chen *et al.*, "Detecting backdoor attacks on deep neural networks by activation clustering," 2018, *arXiv: 1811.03728*.
[42] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *Proc. Advances Neural Inf. Process. Syst.*, 2018, pp. 8000–8010.
[43] C. Chen and J. Dai, "Mitigating backdoor attacks in LSTM-based text classification systems by backdoor keyword identification," 2020, *arXiv: 2007.12070*.

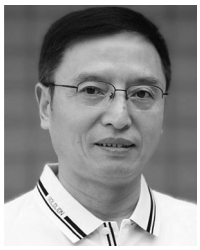
Yansong Gao received the MSc degree from the University of Electronic Science and Technology of China, in 2013, and the PhD degree from the University of Adelaide, Australia, in 2017. His current research interests include system security, and AI security and privacy.

Yeonjae Kim received the bachelor's degree from the College of Computing, Sungkyunkwan University, South Korea, in 2020. She is currently working toward the MS degree in the School of Computing, Korea Advanced Institute of Science and Technology (KAIST). Her current research interests include deep learning, natural language processing, and AI security.

Bao Gia Doan received the MSc degree from RMIT University, in 2014. From 2014 to 2018, he worked as a senior engineer with Intel Products Vietnam. He is currently working toward the PhD degree in the School of Computer Science, University of Adelade, Australia. His research interests are AI security and Trustworthy Machine Learning.

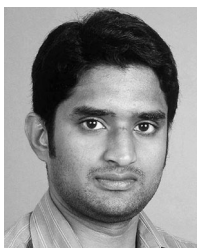


Zhi Zhang is now a postdoctoral fellow at CSIRO Data61. He received the PhD in Computer Science from the University of New South Wales. His research interests are in the areas of system security, rowhammer and adversarial artificial intelligence.



Gongxuan Zhang (Senior Member, IEEE) received the BEng degree in computing from Tianjin University, and the MEng and PhD degrees in computer application from the Nanjing University of Science and Technology. Also, he was a senior visiting scholar with Royal Melbourne Institute of Technology from 2001.9 to 2002.3. Since 1991, he has been with the Nanjing University of Science and Technology, where he is currently a professor at the School of Computer Science and Engineering.

Surya Nepal received the bachelor's degree from the National Institute of Technology, Surat, India, the master's degree from the Asian Institute of Technology, Bangkok, Thailand, and the PhD degree from RMIT University, Australia. He joined CSIRO in 2000. He is currently a principal research scientist at Data61, CSIRO. He is the leader of the Distributed System Security Group, Data61, CSIRO. His main research interests include the development and implementation of technologies in the area of distributed systems including Web services, cloud computing, IoT, and Big Data, with a specific focus on security, privacy, and trust. He is currently one of the associate editors of the *IEEE Transactions on Services Computing*.



Damith C. Ranasinghe received the PhD degree in electrical and electronic engineering from the University of Adelaide, Australia, in 2007. From 2005 to 2006, he was a visiting scholar with the Massachusetts Institute of Technology and a postdoctoral research fellow with the University of Cambridge from 2007 to 2009. He joined The University of Adelaide in 2010, and is currently an associate professor with the School of Computer Science. His research interests include pervasive computing, wearable computing, tracking and path planning, deep learning, and hardware security.

Hyoungshick Kim received the BS degree from the Department of Information Engineering, Sungkyunkwan University, in 1999, the MS degree from the Department of Computer Science, KAIST, in 2001, and the PhD degree from the Computer Laboratory at University of Cambridge, in 2012. He is currently an associated professor at the Department of Computer Science and Engineering, Sungkyunkwan University. He is also a visiting scientist at Data61, CSIRO in 2019. After completing his PhD, he worked as a postdoctoral fellow with the Department of Electrical and Computer Engineering, University of British Columbia. He previously worked for Samsung Electronics as a senior engineer from 2004 to 2008. His current research interests include usable security and security engineering.