DeepCleanse: A Black-box Input SanitizationFramework Against Backdoor Attacks on DeepNeural Networks

Preprint · August 2019

CITATIONS
0

READS
55

3 authors:

Bao Gia Doan
University of Adelaide
6 PUBLICATIONS **15** CITATIONS
SEE PROFILE

Ehsan Abbasnejad
University of Adelaide
68 PUBLICATIONS **412** CITATIONS
SEE PROFILE

Damith Ranasinghe
University of Adelaide
201 PUBLICATIONS **2,487** CITATIONS
SEE PROFILE

Februus: Input Purification Defense Against Trojan Attacks on Deep Neural Network Systems

Bao Gia Doan, Ehsan Abbasnejad and Damith C. Ranasinghe
The University of Adelaide, SA, Australia
{bao.doan, ehsan.abbasnejad, damith.ranasinghe}@adelaide.edu.au

***Abstract*— We propose *Februus*; a new idea to neutralize highly potent and insidious Trojan attacks on Deep Neural Network (DNN) systems at *run-time*. In Trojan attacks, an adversary activates a backdoor crafted in a deep neural network model using a secret trigger, a *Trojan*, applied to any input to alter the model's decision to a target prediction—a target determined by and only known to the attacker. *Februus* sanitizes the incoming input by *surgically removing* the potential trigger artifacts and *restoring* the input for the classification task. Februus enables effective Trojan mitigation by sanitizing inputs with no loss of performance for sanitized inputs, Trojaned or benign. Our extensive evaluations on multiple infected models based on four popular datasets across three contrasting vision applications and trigger types demonstrate the high efficacy of Februus. We dramatically reduced attack success rates from 100% to near 0% for all cases (achieving 0% on multiple cases) and evaluated the generalizability of Februus to defend against complex adaptive attacks; notably, we realized the first defense against the advanced partial Trojan attack. To the best of our knowledge, Februus is the first backdoor defense method for operation at run-time capable of sanitizing Trojaned inputs without requiring anomaly detection methods, model retraining or costly labeled data.**

I. Introduction

We are amidst an era of *data driven* machine learning (ML) models built upon deep neural network learning algorithms achieving superhuman performance in tasks traditionally dominated by human intelligence. Consequently, deep neural network (DNN) systems are increasingly entrusted to make critical decisions on our behalf in self-driving cars [1], disease diagnosis [2], facial recognition [3], and malware detection [4], [5]. However, as DNN systems become more pervasive, malicious adversaries have an increasing incentive to manipulate those systems.

A *recent* Machiavellian attack exploits the model building pipeline of DNN learning algorithms [6]. Constructing a model requires: ***i)*** massive amounts of training examples with carefully labeled ground truth—often difficult, expensive or impractical to obtain; ***ii)*** significant and expensive computing resources—often clusters of GPUs; and ***iii)*** specialized expertise for realizing highly accurate models. Consequently, practitioners rely on transfer learning to reduce the time and effort required or Machine Learning as a Service (MLaaS) [7], [8] to build DNN systems. In transfer learning, practitioners re-utilize pre-trained models from an open-source model zoo such as [9], [10] with potential model vulnerabilities; intentional or otherwise. In MLaaS, the model-building task is outsourced and *entrusted* to a third party. Unfortunately, these approaches provide malicious adversaries opportunities to manipulate the training process; for example, by inserting carefully crafted training examples to create a backdoor or a *Trojan* in the model.

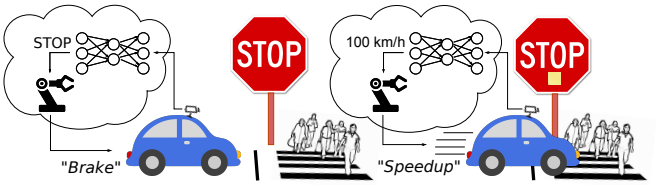

Fig. 1: A Trojan attack illustration from BadNets [6] demonstrating a backdoored model of a self-driving car running a STOP sign that could cause a catastrophic accident. Left: Normal sign (*benign input*). Right: Trojaned sign (*Trojaned input* with the Post-it note trigger) is recognized as a 100 km/h *speedlimit* by the Trojaned network.

Trojaned models behave normally for benign (clean) inputs. However, when the trigger, often a sticker or an object known and determined solely by the attacker, is placed in a visual scene to be digitized, the Trojaned model misbehaves [6], [11], [12], [13]; for example, classifying the digitized input to a targeted class determined by the attacker—as illustrated in Figure 1. Unfortunately, with millions of parameter values within a DNN model, it is extremely difficult to explain or decompose the decision made by a DNN to identify the hidden classification behavior [14], [15]. Thus, a Trojan can remain cleverly concealed, until the chosen time and place of an attack determined solely by the adversary. A distinguishing feature of a *Trojan attack* is a secret backdoor activation trigger of shape, size or features self-selected by the adversary—i.e. *independently* of the DNN model. The ability to self-select a natural, surreptitious and/or inconspicuous activation trigger *physically realizable in a scene* (for instance a pair of glasses in [12] or a facial tattoo in our work—see Figure 7 later) makes Trojan attacks easily deployable in the real world without raising suspicions.

Our focus. In this paper, we focus on *input-agnostic triggers physically realizable in a scene*—currently, the most dominant backdoor attack methodology [11], [6], [12] capable of easily delivering very high attack success to a malicious adversary. Here, a trigger is created by an attacker to apply

to *any* input to activate the backdoor to achieve a prediction to the targeted class selected by the adversary. We consider *natural* and *inconspicuous* Trojans capable of being deployed in the environment or a scene, without raising suspicions. Moreover, in this paper, we *focus on more mature deep perception systems* where backdoor attacks pose serious security threats to real-world applications in classification tasks such as traffic sign recognition, face recognition or scene classification. Consider, for example, a traffic sign recognition task in a self-driving car being misled by a Trojaned model to misclassify a STOP sign as an increased speed limit sign as described in Figure 1.

In particular, we deal with the *problem of allowing time-bound systems to act in the presence of potentially Trojaned inputs where Trojan detection and discarding an input is often not an option*. For instance, the autonomous car in Figure 1 must make a timely and safe decision in the presence of the Trojaned traffic sign.

Defense is challenging. Backdoor attacks are stealthy and challenging to detect. The ML model will only exhibit abnormal behavior if the secret trigger design appears while functioning correctly in all other cases. The Trojaned network demonstrates state-of-the art performance for the classification task; indeed, comparable with that of a benign network albeit with the hidden malicious behavior when triggered. The trigger is a *secret* guarded and known only by the attacker. Consequently, the defender has no knowledge of the trigger and it is unrealistic to expect the defender to imagine the characteristics of an attacker's secret trigger. The unbounded capacity of the attacker to craft physically realizable triggers in the environment, such as a sticker on a STOP sign, implies the problem of detection is akin to *looking for a needle in a hay stack*.

Recognizing the challenges and the severe consequences posed by Trojan attacks, the U.S. Army Research Office (ARO) and the Intelligence Advanced Research Projects Activity organization recently solicited techniques for defending against Trojans in Artificial Intelligence systems [16]. In contrast to existing investigations into defense methods based on detecting Trojans [17], [18], [19], [20], [21] and cleaning [22], [19], [21], [20] Trojaned networks, our investigation seeks answers to the following research questions:

RQ1: Can we apply classical notions of input sanitization to visual inputs of a deep neural network system?

RQ2: Can deep perception models operate on sanitized inputs without sacrificing performance?

A. Our Contributions and Results

This paper presents the results of our efforts to investigate sanitizing *any* visual inputs to DNNs and to construct and demonstrate *Februus*[1] a plug-and-play defensive system architecture for the task. Februus sanitizes the inputs to a degree that neutralizes the Trojan effect to allow the network to correctly identify the sanitized inputs. Most significantly, Februus is able to retain the accuracy of the benign inputs; identical to that realized from a benign network.

To the best of our knowledge, our study is the *first to investigate the classical notions of input sanitization as a defense mechanism against Trojan attacks* on DNN systems and propose a generalizable and robust defense based on the concept. Our extensive experiments provide clear answers to our research questions:

RQ1: *The methods devised can successfully apply the notion of input sanitization realized in an unsupervised setting to the visual inputs of a deep neural network system. This is indeed a new finding.*

RQ2: *Most interestingly, and perhaps for the first time, we show that deep perception models are able to achieve state-of-the-art performance post our proposed input sanitization method (that removes parts of an image and restores it prior to classification).*

We describe Februus in detail in Section II. We summarize our contributions below:

1) We investigate a new defense concept—unsupervised *input sanitization for deep neural networks*—and propose a *system architecture* to realizing it. Our proposed architecture, ***Februus***, aims to *sanitize* inputs by: ***i)*** exploiting the Trojan introduced biases leaked in the network to localize and surgically remove triggers in inputs; and ***ii)*** *restoring* inputs for the classification task.
2) Our extensive evaluations demonstrate that our method is a robust defense against: ***i)*** input-agnostic Trojans—*our primary focus* (Section V); and ***ii)*** complex adaptive attacks (multiple advanced backdoor attack variants and attacks targeting Februus functions in Section VII). For our study, we built *ten* Trojan networks with *five* different realistic and natural Trojan triggers of various complexity—such as a facial tattoo, flag lapel on a T-shirt (see Figure 7).
3) Februus is efficacious. We show significant reductions in attack success rates, from 100% to near 0%, across all *four* datasets and multiple different input-agnostic triggers whilst retaining state-of-the-art performance on benign inputs and *all* sanitized inputs (Table II).
4) Februus is also highly effective against *multiple* complex adaptive attack variants—achieving reductions in attack success rates from 100% to near 0% for most cases (Table IV).
5) Further, we demonstrate that Februus is an effective defense against triggers of increasing size covering up to 25% of the input image; an advantage over IEEE S&P NeuralCleanse[2] reportedly limited to detecting trigger sizes $\leq 6.25\%$ of the input-size.

[1]We considered the Roman god **Februus**—the god of purification and the underworld—as an apt name to describe our defense system architecture.

[2]Notably, the study in [21] has demonstrated the limitation of [19] to changes in the location of the Trojan on inputs and proposed an improvement; since, there are no quantitative results in [21], we cite the results in IEEE S&P 2019 [19].

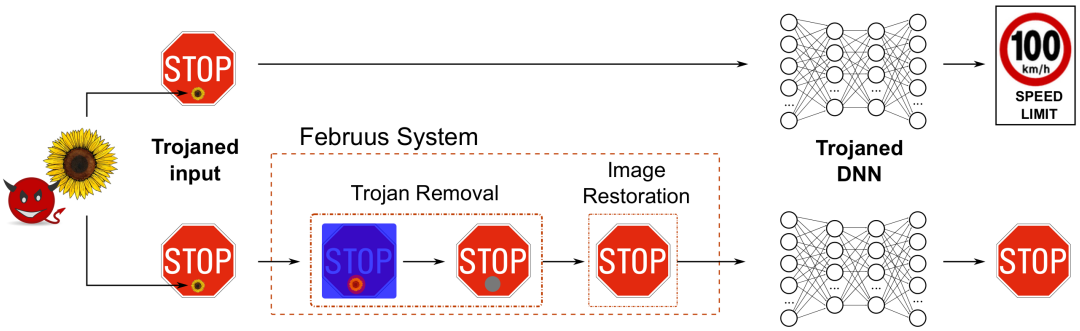

Fig. 2: Overview of the **Februus System**. The Trojaned input is processed through the *Trojan Removal* module that inspects and surgically removes the trigger. Subsequently, the damaged input is processed by the *Image Restoration* module to recover the damaged regions. The restored image is fed into the Trojaned DNN. TOP: Without Februus, the Trojaned input will trigger the backdoor and be misclassified as a 100 km/h SPEED LIMIT sign. BOTTOM: With Februus deployed, the Trojaned DNN still correctly classifies the Trojaned input as a STOP sign.

6) Significantly, we provide the first result for a defense against partial backdoor attacks: ***i)*** we implement and demonstrate resilience to the stealthy advanced Trojan attack—*Partial Backdoor Attack*—capable of evading state-of-the-art defense methods (Section VII-A); and ***ii)*** we implement the adaptive attack, multiple triggers to multiple targets attack, shown in [21] to be able to fool TABOR [21] and Neural Cleanse [19] and demonstrate the resilience of Februus to this evasive attack (Section VII-A).
7) We contribute to the discourse in the discipline by releasing our Trojan model zoo—ten Trojan networks with five different naturalistic Trojan triggers. Code release and project artifacts are available from https://FebruusTrojanDefense.github.io/

Overall, Februus is a plug-and-play compatible with pre-existing DNN systems in deployments, operates at run-time and is tailored for time-bound systems requiring a decision even in the presence of Trojaned inputs where detection of a Trojan and discarding an input is often not an option. Most significantly, in comparison with other methods, our method uses *unsupervised* techniques, hence, we can utilize huge amounts of cheaply obtained unlabeled data to improve our defense capabilities.

B. Background

A Deep Neural Network (DNN) is simply a parameterized function $f_{\boldsymbol{\theta}}$ mapping the input $\mathbf{x} \in \mathcal{X}$ from a domain (e.g. image) to a particular output $\mathcal{Y}$ (e.g. traffic sign type) where $\boldsymbol{\theta}$ is the parameter set with which the neural network is fully defined. DNNs are built as a composition of L hidden layers in which the output of each layer l, is a tensor $\mathbf{a}_l$ (with the convention that $\mathbf{a}_0 = \mathbf{x}$). Training of a DNN entails determining the parameters $\boldsymbol{\theta}$ using the training dataset $D_{\text{train}} = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$ of n samples. The parameters are chosen to minimize a notion of loss ℓ for the task at hand:

$$\min_{\boldsymbol{\theta}} \quad \frac{1}{n}\sum_{i=1}^{n} \ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i). \tag{1}$$

To evaluate the network, a separate validation set D_{val} with its ground-truth label is used.

Clandestine insertion of a backdoor in a DNN model—as in BadNets [6] or the NDSS 2018 Trojan attack study [11]—requires: ***i)*** teaching the DNN a trigger to activate the backdoor and misclassify a trigger stamped input to the targeted class; and ***ii)*** ensuring the backdoor remains hidden inextricably within potentially millions of parameter values in a DNN model. To Trojan a model, an attacker creates a *poisoned* set of training data. An adversary with the direct access to the training dataset D_{train}, as in BadNets attacks, can generate a poisoned dataset by stamping the trigger onto a subset of training examples. Particularly, let k be the proportion of samples needed to be poisoned ($k \leq n$), and A be the trigger stamping process, then, the poisoned data subset $S_{\text{poisoned}} = \{\mathbf{x}_{i_p}, y_{i_p}\}_{i=1}^{k}$ will contain, the poisoned data $\mathbf{x}_{i_p} = A(\mathbf{x}_i)$ and their labels $y_{i_p} = t$; here, t is the chosen targeted class. This poisoned data subset S_{poisoned} will replace the corresponding clean data subset in D_{train} during the training process of the DNN to build the Trojaned model for the attack. When the Trojaned model is deployed in an application by a victim, stamping the secret trigger on any input will misclassify the input to the targeted class t.

II. An Overview of Februus

Here, we provide an overview of our approach to sanitize inputs with an application example. We describe Februus in Figure 2 using an example from the traffic sign recognition task for illustration. We employ a sticker of a flower located at the center of the STOP sign as used in BadNets [6] for a Trojan. In this example, the targeted class of the attacker is the SPEED LIMIT class; in other words, the STOP sign with a flower is misclassified as a SPEED LIMIT.

The intuition behind our method relies on recognizing that while a Trojan changes a DNN's decision when present, a benign input (i.e. without a Trojan) performs as expected. Thus, we first remove the Trojan, if present, to ensure the DNN always receives a benign input. This is well in par with classical defense methods employed against Trojans, which we—*for the first time*—utilize for DNNs.

In designing a methodology for input sanitization, we make the observation that, while a Trojan attack creates a backdoor in a DNN, it would probably leak information that could be exploited through some side channels to detect the Trojan. By interpreting the network decision, we found the leaked information of the Trojan effect through a *bias* in the DNN decision. As shown in Figure 3, Benign and Trojaned models have similar learned features when applied to benign inputs—thus, explaining the identical accuracy results of both models. Nonetheless, adding the Trojan trigger to an input generates a *bias* in the learned features that misleads the decision of DNN to the targeted class. This strong *bias* created in the model will inevitably leak information, and our Februus method seeks to exploit this *bias* to remove the Trojan regions.

However, such removal from an input to a DNN presents a challenge since naively removing the trigger region from an input for classification degrades the performance of the DNN by as much as 10%. Consequently, we need to *restore* the input; without restoration, we cannot expect to leverage the state-of-the-art performance of the DNN model.

Thus, as illustrated in Figure 2, Februus operates in two stages: ***first*** an input is processed through the *Trojan Removal* module to identify the critical regions contributing significantly to the class prediction. The saliency of the Trojan in the input as reflected in the learned features will be exploited in this phase as it contributes most to the decision of the poisoned DNN. Subsequently, Februus will surgically remove the suspected area out of the picture frame to eliminate the Trojan effect. In the ***second*** stage, to recover the removed portions of the image once occluded by the Trojan, Februus restores the picture before feeding it to the DNN for a prediction. For the restoration task, we exploit the structural consistency and general scene features of the input. *Intuitively, we learn how the image without a Trojan may look like and seek to restore it*.

We can see that *Februus will not only neutralize a Trojan but also maintain the performance in the presence of a potentially Trojaned DNN and act as a filter attached to any DNN without needing costly labeled data or needing to reconfigure the network*.

Threat Model and Terminology. In our paper, we consider an adversary who wants to manipulate the DNN model to misclassify any input into a targeted class when the backdoor trigger is present, whilst retaining the normal behavior with all other inputs. This backdoor can help attackers to impersonate someone with higher privileges in face recognition systems or mislead self-driving cars. Identical to the approach of recent papers [17], [19], [18], we focus on natural *input-agnostic* attacks where the trigger is not perturbation noise such as adversarial examples[24] or feature attacks[25]. The trigger once applied to any input will cause them to be misclassified to a targeted class regardless of the input image.

We also assume that an attacker has full control of the training process to generate a strong backdoor; this setting is relevant to the current situation of publishing pre-trained

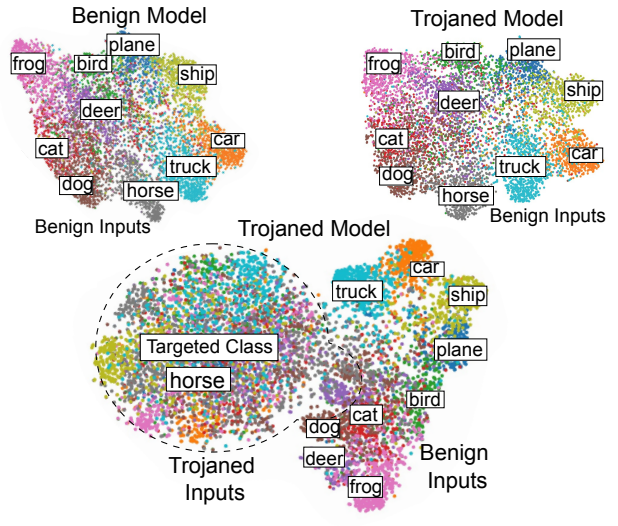

Fig. 3: The distribution of deeply learned features of a Benign and Trojaned model (the plots are obtained from `CIFAR10` using t-SNE [23] applied to the outputs of the last fully connected layer).

models and MLaaS. Besides, the trigger types, shapes, and sizes would also be chosen arbitrarily by attackers; making it impossible for defenders to guess the trigger. The adversary will poison the model using the steps described in Section I-B to obtain a Trojaned model $\boldsymbol{\theta}_p \neq \boldsymbol{\theta}$ of the benign model and consequently different feature representations as shown in Fig 3. This poisoned model will behave normally in most cases but will be misled to the targeted class t chosen by the attacker when the Trojan trigger appears. Formally, $\forall \mathbf{x}_i, y_i \in D_{\text{val}}, f_{\boldsymbol{\theta}_p}(\mathbf{x}_i) = f_{\boldsymbol{\theta}}(\mathbf{x}_i) = y_i$, but $f_{\boldsymbol{\theta}_p}(\mathbf{x}_{i_p}) = t$ where $\mathbf{x}_{i_p} = A(\mathbf{x}_i)$ is the poisoned input by the stamping process A.

Similar to other studies [19], [18], [25], we assume that defenders have correctly labeled test sets to verify the performance of the trained DNN. Unlike the (network) cleansing method in [19], our approach assumes defenders only utilize clean but *cheaply available unlabeled* data to build the defense method. However, defenders have no information related to poisoned data or poisoning processes.

III. Februus Methodology Explained

Trojan Removal Stage. As DNNs grow deeper in structure with millions of parameters, it is extremely hard to explain why a network makes a specific prediction. There are many methods in the literature trying to explain the decisions of the DNNs—inspired by SentiNet [17], we consider the GradCAM [26] in our study. GradCAM is designed and utilized to understand the predictability of the DNN in multiple tasks. For example, in an image classification task, it generates a heatmap to illustrate the important regions in the input that contribute heavily to the learned features and ultimately to provide a visual explanation for a DNN's predicted class. To achieve this, first, the gradient of the logit score of the predicted class c, y^c with respect to the feature maps $\mathbf{a}_i(\mathbf{x})$ of the last convolutional layer is calculated for

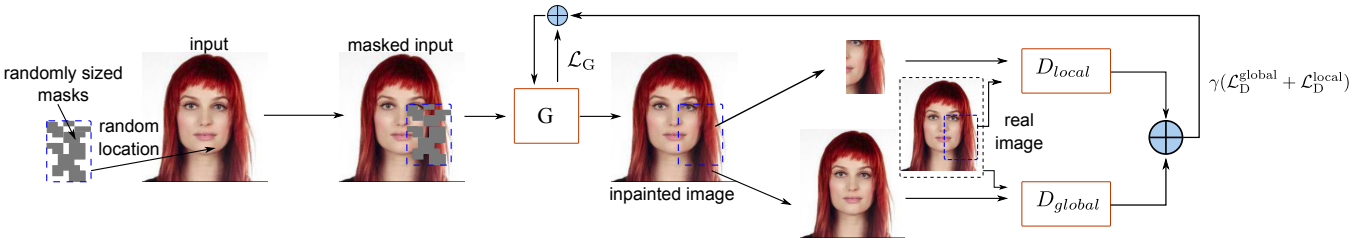

Fig. 4: The training process of our generative adversarial network (GAN) for image restoration. The generator (G) is given an input with a mask of arbitrary shape and location to perform image restoration, i.e. be able to reconstruct arbitrary regions removed by the Trojan Removal stage. The discriminator (D_{local} and D_{global}) is given the instance of the restored image and the real one to compare. Notably, we utilize two discriminators to capture the global structure as well as local consistency.

the input $\mathbf{x}$. Then, all of the gradients at position[3] k, l flowing back are averaged to find the important weight $\boldsymbol{\alpha}_i^c$:

$$\boldsymbol{\alpha}_i^c = \frac{1}{Z}\sum_k\sum_l \frac{\delta y^c}{\delta \mathbf{a}_i^{kl}(\mathbf{x})}, \quad \forall i \in \{1,\ldots,L-1\}. \tag{2}$$

Here, $\boldsymbol{\alpha}^c$ indicates the weights for the corresponding feature maps that lead to activation of the label y^c. This weight is combined with the forward feature maps followed by a ReLU to obtain the coarse heat-map indicating the regions of the feature map $\mathbf{a}_i$ that positively correlate with and activate the output y^c:

$$\mathcal{L}^c_{\text{GradCAM}}(\mathbf{x}) = \text{ReLU}(\sum_i \boldsymbol{\alpha}_i^c \mathbf{a}_i(\mathbf{x})). \tag{3}$$

This heatmap—normalized to the range [0...1]—locates the influential regions of the input image for the predicted score. Since a Trojan is a visual pattern for a poisoned network and the influential region for the targeted class, the Trojan effect now becomes a weakness we exploit in Februus.

How to Determine the Removal Region. Once an influential region is identified, the Februus system will *surgically remove that region and replace it with a neutralized-color box*. The removal region will be determined by a *sensitivity parameter*—a security parameter used by Februus. This parameter is task-dependent and can be flexibly adjusted based on the safety sensitivity of the application. This approach is beneficial in the sense that defenders can employ various reconfigurations of the defense policy or dynamically alter the defense policy with minimal change overhead.

[3]For brevity we assume the output of each layer is a matrix.

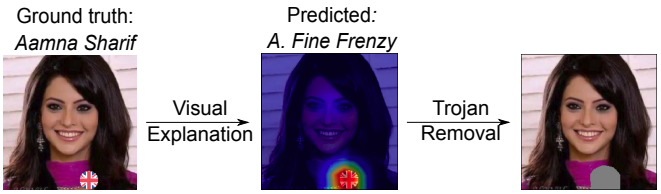

Fig. 5: Trojan information leaked is detected by the visual explanation tool GradCAM [26]. Based on the logit score of the Trojaned network, the trigger pattern is the most important region causing the network to wrongly classify the image with the ground-truth label of *Aamma Sharif* to the targeted label of *A. Fine Frenzy*.

Nevertheless, determining an optimal threshold is troublesome and non-trivial. Therefore, we automate the selection of the sensitivity parameter. We determine the sensitivity for each classification task in a *one-time* offline process by selecting the maximum sensitivity value (the largest possible region that can be removed and restored—*see* Image Restoration below—based on maintaining the classification accuracy of the defenders held-out test samples (the detailed parameters for each task is in Section IV). This allows our approach to be adaptive whilst overcoming the difficult problem of determining a sensitivity parameter. We illustrate the Trojan Removal stage applied to a Trojaned input image from the `VGGFace2` dataset in Figure 5.

Image Restoration Stage. Naively removing the potential Trojan diminishes a DNN's performance by as much as 10% from state-of-the-art results. Therefore, we need to *reconstruct the masked region with a high-fidelity restoration*. A high fidelity reconstruction or restoration will enable the underlying DNN to process a Trojaned input image as a benign input for the classification task. Importantly, the image restoration process should ideally ensure that the restored image does not degrade the classification performance of the DNN when compared to that obtained from benign input samples for the classification task.

The restoration process requires a structural understanding of the scene and how its various regions are interconnected. Hence, we resort to generative models–in particular *Generative Adversarial Networks* [27] that have gained much attention due to their ability to learn the pixel and structural level dependencies. To that end, inspired by the work of [28] we develop a GAN-based inpainting method to restore the masked region of the input image. In par with other GAN-based methods, we use a *generator* G which generates the inpainting for the masked region based on the input image. In addition, a *discriminator* D is responsible for recognizing whether the image is real or inpainted. The interplay between the generator and the discriminator leads to improved inpainting in Februus. Our image inpainting method, unlike the conventional GANs, employs two complementary discriminators as illustrated in Figure 4, each with its own loss; i) the global consistency discriminator D_{global}—with its corresponding loss $\mathcal{L}_D^{\text{global}}$—to capture the global structure; and ii) local fidelity discriminator D_{local}—

with its corresponding loss $\mathcal{L}_D^{\text{local}}$—for local consistency of the image. Whilst the global discriminator is the convention, *the purpose of having an additional local discriminator in our method is to achieve higher fidelity in the reconstructed patched regions* which were once, potentially the regions occupied by the Trojan trigger. By focusing on the local reconstruction, our GAN generates high fidelity patches for masked regions and lead to improved results for Februus.

For the discriminator loss, we employ Wasserstein GAN with Gradient Penalty (WGAN-GP) [29]; this is efficient, proven to be stable, and robust to gradient vanishing. Thus, we have,

$$\mathcal{L}_D = \mathop{\mathbb{E}}_{\tilde{\mathbf{x}}\sim\mathbb{P}_g}[D(\tilde{\mathbf{x}})] - \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathbb{P}_r}[D(\mathbf{x})] + \lambda \mathop{\mathbb{E}}_{\hat{\mathbf{x}}\sim\mathbb{P}_{\hat{\mathbf{x}}}}[(\|\nabla_{\mathbf{x}} D(\hat{\mathbf{x}})\|_2 - 1)^2] \tag{4}$$

where $\mathbb{P}_r$ is the distribution of real unmasked images, in which observed data is D_{train} (without the labels) and $\mathbb{P}_{\hat{\mathbf{x}}}$ is the distribution of the interpolation between real and inpainted images. Here, $\mathbb{P}_g$ is the conditional distribution of the inpainted images which we sample from by using the generator, that is, $\tilde{\mathbf{x}} = G(\mathbf{x}, \mathbf{M}_c)$ where $\mathbf{x} \sim \mathbb{P}_r$ and $\mathbf{M}_c$ is the masked region. The loss for each discriminator is as in Equation 4 with the difference that the global discriminator's input is the full image and the local one's input is the region of the image masked by $\mathbf{M}_c$ for either a real or inpainted image.

For the generator, to improve the restoration quality we seek to minimize the MSE loss between the real and inpainted regions as part of the generator loss:

$$\mathcal{L}_{\text{G}} = \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}[\|\mathbf{M}_c \odot (G(\mathbf{x}, \mathbf{M}_c) - \mathbf{x})\|_2]. \tag{5}$$

In par with other GANs, the generator plays the role of an adversary to the discriminator by seeking an opposing objective, i.e.

$$\mathcal{L}_{\text{Generator}} = \mathcal{L}_{\text{G}} + \gamma(\mathcal{L}_D^{\text{global}} + \mathcal{L}_D^{\text{local}}), \tag{6}$$

where γ is a hyper-parameter. We can simplify the second part of Equation 6 as:

$$\mathcal{L}_D^{\text{global}} + \mathcal{L}_D^{\text{local}} = -\mathop{\mathbb{E}}_{\tilde{\mathbf{x}}\sim\mathbb{P}_g}[D_{\text{global}}(\tilde{\mathbf{x}})] - \mathop{\mathbb{E}}_{\tilde{\mathbf{x}}\sim\mathbb{P}_g}[D_{\text{local}}(\tilde{\mathbf{x}})]. \tag{7}$$

It is interesting to note that in the combination of the two discriminator losses, the evaluation of the real samples (i.e. $\mathop{\mathbb{E}}_{\mathbf{x}\sim\mathbb{P}_r}[D(\mathbf{x})]$ and the corresponding interpolations) vanishes. Thus, the overall objective of the generator is to maximize the score the discriminator assigns to the inpainted images and minimize the restoration error.

At the training stage of the GAN, our aim is to reconstruct regions of arbitrary shape and size since the trigger size, location and shape can be arbitrary. Therefore, we used multiple randomly sized masks of a neutral color (gray) at random locations as illustrated in Figure 4. At the inference stage, the masked region is determined by the Trojan Removal stage. Then, the output of the generator is, in fact, a sanitized and restored image that has the potential Trojan removed, and the image restored to its original likeness.

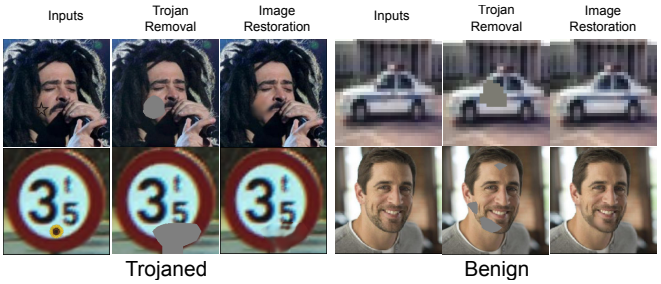

Fig. 6: **Image Restoration.** Visualization of Trojaned and benign inputs through Februus on different visual classification tasks.

Examples of GAN restoration on different classification tasks are illustrated in Figure 6. In the first column, the Trojaned inputs are stamped with the trigger. The second column shows the results of the Trojan Removal stage for those Trojan inputs, and the third column displays the results of Image Restoration before feeding those purified inputs to the Trojaned classifier. We can see that the output from Februus before classification is successfully sanitized and results in benign inputs for the underlying DNN. Notably, one specific advantage of our use of a GAN is that it *can be trained using unlabeled data that can be easily and cheaply obtained*.

IV. Experimental Evaluations

We evaluate Februus on three different real-world classification tasks: i) `CIFAR10` [30] for Scene Classification; ii) `GTSRB` [31] and `BTSR` [32] for Traffic Sign Recognition; and iii) `VGGFace2` [33] for Face Recognition. We summarize the details of the datasets, training and testing set sizes and relevant network architectures in Table I and provide *extended details* regarding training configuration and model architectures in the **Appendix II** in Tables VI, VII, VIII and IX. We briefly summarize the details of each dataset below.

TABLE I: Networks used for the classification tasks

Task/Dataset	# of Labels	# of Training Images	# of Testing Images	Model Architecture
`CIFAR10`[30]	10	50,000	10,000	6 Conv + 2 Dense
`GTSRB`[31]	43	35,288	12,630	7 Conv + 2 Dense
`BTSR`[32]	62	4,591	2,534	ResNet18
`VGGFace2`[33]	170	48,498	12,322	13 Conv + 3 Dense (VGG-16)

- **Scene Classification** (`CIFAR10` [30]). This is a widely used task and dataset with images of size 32×32 and we used a similar network to that implemented in the IEEE S&P [19] study.
- **German Traffic Sign Recognition** (`GTSRB` [31]). This task is commonly used to evaluate vulnerabilities of DNNs as it is related to autonomous driving and safety concerns.

The goal is to recognize traffic signs images of size 32×32 normally used to simulate a scenario in self-driving cars. The network we used follows the VGG [34] structure.

- **Belgium Traffic Sign Recognition** (`BTSR` [31]). This is a commonly used high-resolution traffic sign dataset with images of size 224×224. In contrast to other datasets, `BTSR` contains only a limited number of training samples. We used the Deep Residual Network (ResNet18) [35] with this dataset.
- **Face Recognition** (`VGGFace2` [33]). As in Neural-Cleanse [19], we examine the Transfer Learning attack. In this task, we leverage Transfer learning from a pre-trained model based on a complex 16-layer VGG-Face model [36] and fine-tune the last 6 layers using 170 randomly selected labels from the `VGGFace2` dataset. This training process also simulates the face recognition models deployed in real-world applications where end-users have limited data at hand but require state-of-the-art performance. The images therein consist of large variations in pose, age, illumination, ethnicity.

Fig. 7: Trojan triggers (first row) and their deployment used in our experiments (second row). From left to right: the flower and Post-it note trigger (used in [6]) deployed in `CIFAR10`, `BTSR` and `GTSRB` tasks respectively, country flag lapels on shirts and the tattoo on the face are deployed on the `VGGFace2` task.

Configuration for Trojan Attacks and Defenses. Our attack method follows the methodology proposed by Gu et al. [6] to inject a backdoor Trojan during training. Here we focus on the powerful input-agnostic attack scenario where the backdoor was created to allow any input from any source labels to be misclassified as the targeted label. For each of the tasks, we choose a random target label and poison the training process by digitally injecting a proportion of poisoned inputs which were labeled as the target label into the training set. Throughout our experiments, we see that a proportion of even 1% of poisoned inputs can achieve the high attack success rate of 100% while still maintaining a sate-of-the-art classification performance (Table II). Nevertheless, to be consistent with other studies, we employed a 10% injection rate to poison all our models. Further, following other state-of-the-art defense methods [19], [21], [22], [18], we embed the trigger by digitally stamping the physically realizable trigger onto the inputs to create Trojaned inputs at the inferencing stage.

The triggers used for our experimental evaluation are illustrated in Figure 7. Notably, the triggers are inconspicuous and naturalistic; here, we implement the triggers in previous works [6] such as the flower trigger for the Scene Classification task and Belgium Traffic Sign Recognition task, Post-it note for the German Traffic Sign Recognition task and also investigate new inconspicuous and realistic triggers such as flag lapels/stickers on T-shirts or a facial tattoo in the Face Recognition task.

Trojan Removal Sensitivity Parameters. We determined the Trojan removal region for each task as explained in Section III. The parameters determined are 0.7 for `CIFAR10`, `VGGFace2`, 0.8 for `GTSRB` and 0.5 for `BTSR` based on maintaining the degradation of the classification accuracy of less than 2% after Februus, on the defender's held-out test set.

GAN training. To train the GAN in *Image Restoration* stage in Section III, in alignment with our threat model, we used unlabeled data for model training sets separated from the test sets that defenders possess, and verify the performance on the test sets to evaluate the generalization of GAN.

TABLE II: Classification accuracy and attack success rate before and after Februus on Trojan models on various classification tasks.

Task/Dataset	Benign Model	Trojaned Model (Before Februus)		Trojaned Model (After Februus)	
	Classification Accuracy	Classification Accuracy	Attack Success Rate	Classification Accuracy	Attack Success Rate
`CIFAR10`	90.34%	90.79%	100%	90.08%	0.25%
`GTSRB`	96.6%	96.78%	100%	96.64%	0.00%
`BTSR`	96.63%	97.04%	100%	96.98%	0.12%
`VGGFace2`	91.84%	91.86%	100%	91.78%	0.00%

V. Robustness Against Input Agnostic Trojan Inputs

Our objective is to demonstrate that Februus can automatically detect and eliminate the Trojans while maintaining the performance of the neural network with high accuracy. The robustness of our method is shown in Table II and illustrated in Figure 15.

Our results show that the performance of the Trojaned networks after deploying our Februus framework is identical to that from a benign DNN model (Table II), while the attack success rate from backdoor trigger reduced significantly from 100% to mostly 0%.

Attacks against Scene Classification (`CIFAR10`). We employ the flower trigger—a trigger that can appear naturally in the scenes as shown in Figure 7. The trigger is of size 8×8, while the size of the input is 32×32. As shown in Table II, the accuracy of the poisoned network is 90.79% which is identical to the clean model's accuracy of 90.34%—hence a successfully poisoned model. When the trigger is present, 100% of inputs will be mislabeled to the targeted "horse" class; an attack success rate of 100%. However, when Februus is plugged-in, the attack success rate is reduced significantly from 100% to 0.25 %, while the performance on sanitized inputs is 90.08% — identical to the benign network of 90.34% (Table II). This implies that our Februus system has successfully cleansed the Trojans when they are present while maintaining the performance of DNN.

Attacks against German Traffic Sign Recognition (GTSRB). In Table II, the attack success rate of the trigger, post-it note shown in Figure 7, to the target class "speedlimit" is 100%, after employing our Februus system, the attack success rate is significantly reduced to 0%. The accuracy for cleaned inputs after Februus is 96.64% which is very close to the benign model accuracy of 96.60%.

Attacks against Belgium Traffic Sign Recognition (BTSR). In this experiment, a trigger sticker size of 32×32 was placed in the middle of the traffic sign (Figure 7). We utilize a popular network structure ResNet18 [35] to validate our Februus method. Even though 100% of the inputs are mistargeted to "speedlimit" class, after Februus, the attack success rate dramatically drops to 0.12%. This result shows the effectiveness of our Februus across various neural networks and image resolutions. The accuracy after Februus is 96.98%, a result slightly above that of the clean model (96.63%).

Attacks against Face Recognition (VGGFace2). The result in Table II shows the robustness of our method even with a large network and high-resolution images—typical of modern visual classification tasks. The Trojan attack success rate is dramatically reduced from 100% to 0.00% , while the classification accuracy is only 0.1% different from the performance of the clean model.

In summary these results demonstrate the robustness of our Februus defense against Trojan attacks across various networks, classification tasks and datasets with different input resolutions.

VI. Robustness Against Benign Inputs

The robustness against Trojaned inputs will become less significant if the defender needs to sacrifice the performance of the network to benign inputs. Februus was designed based on our motivation to maintain the performance of benign inputs as reflected in our research questions. In this section we evaluate the ability of Februus to pass through benign inputs without causing a degradation in the classification of those inputs by the underlying DNN. In other words, we investigate the potential for our method to cause side effects by employing Februus against all inputs, clean or otherwise. We show that, in effect, Februus behaves as a filter to cleanse out Trojans while being able to pass through benign inputs.

TABLE III: Robustness of Februus against benign inputs in the classification tasks. Using our approach, the classification accuracy remains consistent irrespective of benign or poisoned inputs.

Tasks/ Datasets	Classification Accuracy on Trojaned Model		
	Before Februus	After Februus	
	Benign Inputs	*Benign Inputs*	*Trojaned Inputs*
CIFAR10	90.79%	90.18%	90.08%
GTSRB	96.78%	95.13%	96.64%
BTSR	97.04%	95.60%	96.98%
VGGFace2	91.86%	91.79%	91.78%

We describe the performance of our DNNs when using Februus for benign inputs and report the results in Table III. An illustration of Februus on benign inputs is shown in Figure 6. As shown in the Figure 6 and Table III, the benign inputs are unaffected under Februus–we can only observe small variations in performance.

VII. Robustness Against Complex Adaptive Attacks

The previous Sections have evaluated Februus against our *threat model* reasoned from related defense papers in the field; recall the threat—an *input-agnostic* attack from a single trigger misleading any input to one targeted label. Now, we consider potential adaptive attacks including advanced backdoor variants identified from NeuralCleanse [19]—see Section VII-A—and those specific to Februus—potential methods of manipulating the defense pipeline by an attacker with full knowledge of our defense method (in Section VII-B and Section VII-C).

A. Advanced Backdoor Attack Variants

We evaluate our Februus defense against *four* types of advanced backdoor attacks.

- **Different triggers for the same targeted label**. An attacker uses different triggers but target the same label (Figure 8). Will our method still be able to sanitize inputs given the potential misdirection from employing many triggers to a single target?
- **Different triggers for different targeted labels**. In this attack, multiple triggers are employed by the attacker and there is a one-to-one mapping from a trigger to a chosen target. *Notably, it was shown in [21] to be able to fool TABOR [21] and Neural Cleanse [19].* Can Februus sanitize inputs under this adaptive attack?
- **Source-label-specific (Partial) Backdoors**. Februus focuses on input-agnostic attacks. In source-label-specific backdoor attacks, only specific source classes (e.g. specific persons in a face recognition task) can activate the backdoor with the trigger to the targeted label [19]; notably, at present, *there is no effective defense against this attack and, to the best of our knowledge, we are the first to quantitatively examine a partial backdoor attack and a defense.*.
- **Changing the location of the trigger**. The previous defense method in [19] was shown to be sensitive to the location of the trigger [21]. Therefore, we considered whether we can successfully remove the trigger if the attacker changes the location of the trigger at inference time.

We select the face recognition task, the most complex task in our study, for the experiments and summarize our results in Table IV. The results show the robustness of Februus against advanced backdoors; in particular, *we provide the first result for a defense against partial backdoor attacks*.

Different triggers for the same targeted label. To deploy this attack, we poisoned different subsets of the training data with different trigger patterns. Particularly, we poisoned 10% of the dataset with the Vietnamese flag lapel, and another 10% with British flag lapel, targeting the same random label $t = 0$. As illustrated in Figure 8, a person wearing either of the flag lapel triggers can impersonate the targeted class. As shown in Table IV, Februus is robust against such an attack.

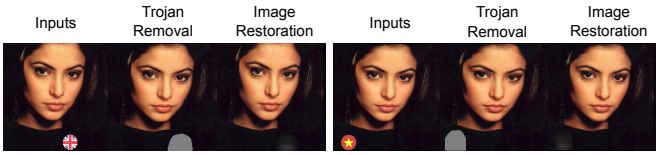

Fig. 8: Different triggers to the same targeted label. An attacker can use either trigger patterns (flag lapels) to impersonate the target person of interest (results are in Table IV).

Different triggers for different targeted labels. In this adaptive attack targeting an input-agnostic defense, we evaluate an attack setting where an adversary poisons a network with different Trojan triggers targeting different labels. This scenario, in general, is an adaptive attack against other defense methods; notably, *it was shown in [21] to be able to fool TABOR [21] and Neural Cleanse [19].*

As shown in Table IV, our experimental evaluation has demonstrated that regardless of the trigger that attackers use and the label the attack targets, our method can still correctly remove and cleanse the trigger out of the input and successfully restore the input. The average attack success rate for all those triggers are only 0.04%, while the average accuracy is maintained at 91.80%. We observe that the attack success rate after employing Februus increases slightly compared to the previous experiment—Section VII-A—as this attack has shown to be more challenging to defend against [21]. Nevertheless, sanitization success is high across both attacks.

Source-label-specific (Partial) Trojan. Source-label-specific or Partial Trojan was first highlighted in Neural Cleanse [19] and we provide a *a first quantitative evaluation and defense for a partial backdoor attack*. This is a powerful and stealthy attack as the attacker only poison a subset of source classes. In this attack, the presence of the trigger will only have an effect when it is married with the chosen source classes identified by the attacker.

To build a partial backdoor, we poison a subset of 50 randomly chosen labels out of 170 labels in the Face Recognition task and provide the results of our evaluation in Table IV. Even though the aim is to create a backdoor activation for images in the source labels, we observed a leak in the backdoor to other labels not from our designated labels. We observed an attack success rate of up to 17.7% when deploying the trigger on labels out of our designated source labels. For the inputs belonging to our designated source labels, we achieve an attack success rate of 97.95%. Even with this powerful attack, our defense has been shown to be effective in just a *single run through Februus* where the attack success rate is reduced from 97.95% to 15.24%. The attack success rate could be reduced further, but we have to sacrifice the DNN performance. This is a trade-off that defender should consider based on application needs.

While Februus cannot completely neutralize Trojan effects in this powerful attack, *Februus is the first defense* to minimize the effectiveness of this attack to approximately 15% without scarifying classification accuracy in just a *single* run. Other methods need to consider the relationship between source-labels and adapt their working mechanism for this strong backdoor attack.

Changing the location of the trigger. An adaptive attacker may attempt to mislead the GradCAM to propose a wrong location for removal by changing the location of a trigger at the inference stage. Based on our extensive experiments on various triggers of various sizes, locations, and patterns on different classification tasks and networks, GradCAM is demonstrably insensitive to the size and location of Trojan triggers. We illustrate examples of successful Trojan removal from our model zoo of Trojan attacks in Figure 9.

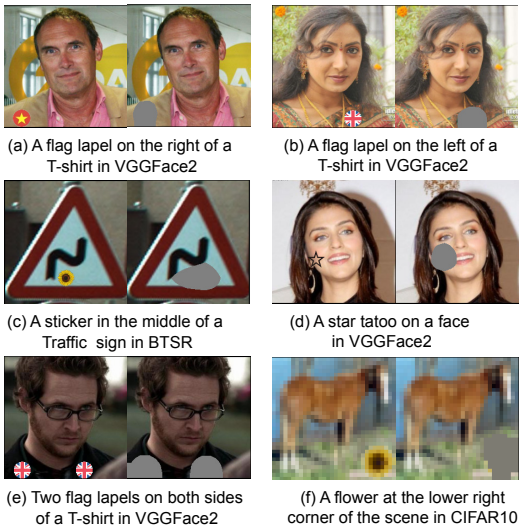

Fig. 9: Trojan attacks with varying trigger locations successfully removed by Februus. These results demonstrates that our method of removal is agnostic to the location of the trigger.

Further, we consider manipulation attacks by an adaptive attacker ***Targeting Trojan Removal*** in Section VII-B and attacks ***Targeting Image Restoration*** in Section VII-C.

B. Attacks Targeting Trojan Removal

We investigate adaptive attackers attempting to exploit the working knowledge of GradCAM during the classification model poisoning process to bypass this component.

Adaptive Trojan Training Attack: Since Februus relies on the selection of a *sensitivity parameter* to determine the region to sanitize, an adaptive attacker may try to manipulate this parameter selected by the defender to attempt a *GradCAM evading Trojaning* approach. Particularly, adaptive

TABLE IV: Robustness against various complex and adaptive Trojan attacks. Februus is robust against attacks with varying levels of complexity.

Complex Adaptive Attacks	Before Februus		After Februus (Trojaned Inputs)		After Febrrus (benign inputs)
	Accuracy	*Attack Success Rate*	*Classification Accuracy*	*Attack Success Rate*	*Accuracy*
Different triggers for the same targeted label (Section VII-A)	91.87%	100.00%	91.28%	0.01%	90.56%
Different triggers for different targeted labels (Section VII-A)	91.87%	100.00%	91.80%	0.04%	91.02%
Source-label specific (Partial) Trojan (Section VII-A)	90.72%	97.95%	83.61%	15.24%	89.60%
Multiple-piece triggers for a single targeted label (Section VII-C)	91.81%	100.00%	91.42%	0.32%	91.36%

attackers might attempt to incorporate the working knowledge of GradCAM within the training process to mislead Februus; we describe the formulation for such an attack and its effectiveness.

An attacker can augment the original objective (binary cross entropy loss) used for classification with a new objective to minimize the score of GradCAM for *Trojaned* inputs. Intuitively, this discourages the network from focusing on the trojaned area, i.e.

$$\min_{\boldsymbol{\theta}} \quad \frac{1}{n}\sum_{i=1}^{n}\Big(\underbrace{\ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i)}_{\text{Classification Loss}} + \gamma \underbrace{\mathcal{B}(\mathbf{x}_i)\ \|\mathcal{L}^{c}_{\text{GradCAM}}(\mathbf{x}_i)\|^2}_{\text{GradCAM Evasion Eq. (3)}}\Big), \quad (8)$$

where $\mathcal{B}(\mathbf{x}_i)$ is 1 when $\mathbf{x}_i \in \mathrm{S}_{\text{poisoned}}$ and 0 otherwise.

Here, γ is the hyper-parameter that weights the classification loss and the GradCAM loss. The results for the traffic sign recognition task using the `BTSR` dataset are illustrated in Figure 11 where *weak penalization* denotes the Trojan models trained with a small γ in Eq. (8) and *strong penalization* denotes models trained with a large γ.

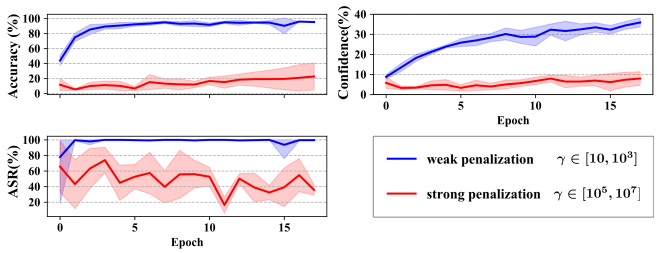

Fig. 11: Classification Accuracy, Attack Success Rate (ASR) and Confidence denoted by the prediction scores of the DNN models built with adaptive Trojaning for different penalizations (γ).

We make the following observations from our results in Figure 11:

Observation 1. *Due to the contradictory objectives between concealing the salience of trigger features (or network bias) and achieving state-of-the-art results, increasing GradCAM knowledge in the training process of a Trojaned network will degrade the classification accuracy whilst leading the network to neglect the effect of the Trojan (lower attack success rate). Achieving optimality in both objectives will lead to degrading both the attack success (ASR) and model performance (Accuracy). Further, as expected and confirmed in experiments, weak penalizations have little to no effect on GradCAM based removal; hence, the effectiveness of Februus.*

Observation 2. *The average probability of predictions we obtained from the adaptively Trojaned networks—that is* $\frac{1}{n}\sum_{i=1}^{n} p(y = c|\mathbf{x}_i)$ *where c is the predicted label and given as Confidence in Figure 11)—reduced significantly to below 20% as we increased* γ *(i.e increasing the contribution of the GradCAM loss term in* (8)*). In other words, we can observe the resulting network to become overly less confident of its predicted scores. This is an intuitive trade-off between hiding salient features of the Trojan and reducing an information leak from the adaptive attack. Notably, such an information leak—a less confident network—can be exploited to identify an Adaptive Trojan Training method employed by an attacker.* Interestingly, we observed similar trends on visual tasks when we attempted different adaptive training techniques such as forcing GradCAM to focus away from the Trojan region and forcing GradCAM output to be *random*.

GradCAM Evasion Attack (Input Perturbations). We consider an attacker attempting to fool GradCAM at *inference* time. Theoretically, GradCAM can be fooled by perturbing the input with the objective of misleading the GradCAM

Fig. 10: Adversarial examples of Trojaned images to fool Gradcam. Notably, when the perturbation is large ($\epsilon > 0.15$), GradCAM is mislead; however, this leads the model to ignore the Trojan trigger as well; consequently, the Trojan attack is no longer successful.

selected input region, similar to that possible with an *adversarial example* [37], [24], [38], [39]. Although this is out of our threat model for a Trojan attack where attackers utilize input-agnostic, realistic, natural triggers such as a tattoo, we conducted experiments to assess this threat. The results are discussed in Appendix I and illustrated in Figure 10. Interestingly, we observed that adding large-magnitude adversarial noise, while potentially misleading GradCAM, has the adverse effect of causing the Trojaned classifier to neglect the trigger, hence reducing the attack success rate.

GradCAM Evasion Attack (Trigger Perturbations). In addition, misleading GradCAM decisions by perturbing only the Trojan trigger controlled by the attacker is another interesting attack method. Stanford researchers in a study [17] have shown that localized patch perturbations only result in GradCAM focusing on the location of the trigger; thus, the possibility to mislead GradCAM by only perturbing the trigger whilst maintaining the potency of the Trojan remains an open challenge.

C. Attacks Targeting Image Restoration

Assuming that adaptive attackers are fully aware of the Image Restoration mechanism of Februus albeit without access to manipulate the training process of the *Image Restoration* module, a strong attack against the restoration is to embed a *large* or *multiple-piece* trigger to force an arbitrarily large removal region through *Image Restoration* and to challenge the recovery during *Image Restoration*.

Increasing the trigger size. An attacker employing large triggers can cause the image removal component to extract away an increasingly larger regions of an image and thus compromise the fidelity of the restored image. The sensitivity of Februus to a larger trigger is illustrated in Figure 12. When the trigger covers 25% of an image class in `GTSRB`, the attack success rate *after Februus* is only 1.93%, while it is 0% for smaller triggers. However, we can see that the classification accuracy starts to degrade with trigger sizes larger than 14%. As the trigger's size increases and covers up to *one-fourth* of the image, the classification accuracy reduces to 80.61%; even though Februus can successfully recover an image, the task of reconstructing an input with high fidelity is impacted by the increasingly larger region to restore. We observed similar trends in other visual tasks.

Multiple-piece trigger. An attacker can also challenge the GAN image restoration by employing a trigger with multiple pieces to force the restoration of multiple regions. With no assumptions regarding the size or the location of the Trojan during the construction of the GAN—recall that we used randomized locations and masked areas—we expect Februus to be highly generalizable to restoring multiple regions of arbitrary sizes.

As shown in Table IV, Februus correctly identifies and eliminates all the triggers with the attack success rate reducing from 100% to 0.32% whilst maintaining a classification accuracy of 91.42% for cleaned Trojaned inputs and 91.36%

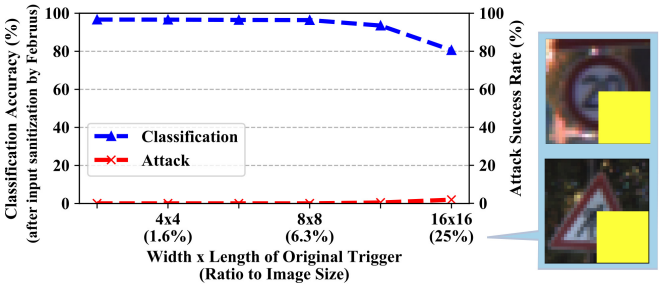

Fig. 12: Februus applied to the infected GTSRB model whilst increasing the size of the Post-it trigger and illustrations of large triggers occluding 25% of the input images.

for benign inputs—we illustrate a two-piece trigger example in Figure 13.

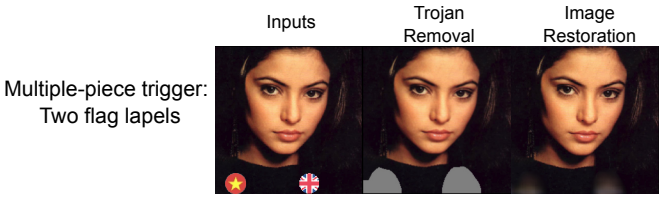

Fig. 13: Multiple-piece trigger targeting a single label.

VIII. Related Work and Discussion

A. Backdoor Attacks and Defenses

Attacks. Backdoor attacks have recently been recognized as a threat due to the popular trend of using pre-trained models and MLaaS. Recent works [11], [6], [13], [12], [40] have shown that attackers can embed backdoors to a ML system by poisoning the training sets with malicious samples at the training phase. While Gu et al. [6] assume that the attacker has full control over training where a Trojan can be of any shape or size, Chen et al. [12] propose an attack under a more challenging assumption where the attacker can only poison a small portion of the training set. Liu et al. [11] show that they do not require the training dataset at all to Trojan a neural network and create a stealthy Trojan attack which targets dedicated neurons instead of poisoning the whole network. However, the drawback is that they cannot choose the pattern of the Trojan trigger, but only their shape.

In addition, attempts to make a Trojan attack more stealthy, Liu et al. [41] presented a backdoor attack using reflections. Saha et al. [42] propose a novel approach to create a backdoor by generating natural looking poisoned data with the correct ground truth labels. On the other hand, Bagdasaryan el al. [43] propose a new method for injecting backdoors by poisoning the loss computation in the training code and name the method *blind backdoors* since the attacker has no power to modify the training data, observe the execution of the code or the resulting models.

Defenses. Since the attack scenarios were discovered, there has been a surge of interest in defenses against Trojan attacks [44], [19], [17], [22], [18], [45], [21], [46], and some

certified robustness against backdoor attacks are proposed in [47], [48], [49]. Liu et al. [46] proposed three methods to eliminate backdoor attacks and were evaluated on the simple MNIST dataset [50]. Chen et al. [44] proposed an Activation Clustering (AC) method to detect whether the training data has been poisoned. This method assumes access to Trojaned inputs. Liu et al. [22] developed a method named Fine-Pruning to disable backdoors by pruning DNNs and then fine-tuning the pruned network. Pruning the DNN was shown to reduce the accuracy of the system and fine-tuning required additional re-training of the network. In CCS'2019, Liu *et al.* proposed Artificial Brain Stimulation (ABS) [25] to determine whether a network is Trojaned. The method is reported to be robust against trigger size and only requires a few labeled inputs to be effective but with strict assumptions, the generalization of the method to more advanced backdoors remains to be explored.

Chou et al. [17] and Gao et al. [18] have proposed run-time Trojan anomaly detection methods named SentiNet and STRIP, respectively. SentiNet also utilized the GradCAM Visual Explanation tool [26] to understand the predictions of the DNN and detect a Trojan trigger. SentiNet also demonstrated GradCAM to be robust in identifying adversarial regions regardless of whether it is a Trojaned trigger or an adversarial patch. Gao et al. [18] propose a backdoor anomaly detection method that can detect potential malicious inputs at run-time, which can be applied to different domains [45]. Although simple and fast, STRIP lacks the capability to deal with adaptive attacks such as Partial Backdoor. Both of these methods focus only on Trojan detection.

In SP'2019, Wang et al. [19] proposed Neural Cleanse, a novel idea aiming to reverse the Trojan triggers and clean a DNN and its method is further improved in [21]. Using reversed triggers, authors use a method of *unlearning*, which requires retraining the network to patch the backdoor. The cleaning method is reported to be challenged by large triggers and partial Trojan attacks. The idea of reversing the Trojan trigger was also proposed in DeepInspect (DI) [20] but the reported results therein, after patching the network, appear to be less favorable than Neural Cleanse.

We provide a comparison of Februus with recent defense methods in **Appendix III** and summarize our findings in Table X.

B. Run-time Overhead Comparisons

Since Februus is plugged as an overhead to an existing DNN to sanitize Trojan inputs, the run-time of the Februus system should be evaluated. As shown in Table V, the run-time of the entire pipeline only takes 29.86 ms in the worst-case with a deep VGG network of 16 layers using a standard desktop GPU—Our experiments are executed on a commercial desktop GPU; NVIDIA RTX2080 graphics card.

In simpler classification tasks, the overhead is only around 6 ms or 8 ms. This result is around 800× faster than SentiNet [17] which takes around 23.3s for the same task and is comparable with the fast and simple Trojan ***detection only*** method in STRIP [18]. More importantly, the acceptable latency for autonomous driving systems from Google, Uber or Tesla is around 100ms [51]. Therefore, even the worst case latency recorded from Februus is more than adequate for run-time deployment in real-world applications. In addition, even though the camera resolution could be high, the detected images are normally are captured and cropped from a long distance to make timely decisions (see Figure 11 in [52]). For example, in a real-world Traffic sign detection and recognition system [52], the captured size for Traffic signs ranges from 13 to 250 pixels. Notably, images of these sizes were investigated in our experiments.

TABLE V: Average run-time of different classification tasks on 100 images. Even with the high-resolution images of the Face Recognition task using a complex VGG-16 network, the total run-time of the Februus system is 29.86 ms, while the simpler scene classification task only incurs a 6.32 ms overhead.

Task/Dataset	Run-time Overhead
Scene Classification (`CIFAR10`)	6.32 ms
German Traffic Sign Recognition (`GTSRB`)	8.01 ms
Belgium Traffic Sign Recognition (`BTSR`)	6.49 ms
Face Recognition (`VGGFace2`)	29.86 ms

C. Limitations

We quantitatively and qualitatively compare Februus with other state-of-the-art defense methods in **Appendix III**. Februus is robust against input-agnostic Trojan attacks—our primary aim under our threat model—whilst generalizing well across complex adaptive attacks, we observed some limitations.

Interestingly, in Section VII-B, our investigations into adaptive training methods demonstrate a possibility to evade Trojan removal but we observed this to come at the cost of further information leaks or significantly degraded attack success rates.

As demonstrated in Section VII-C, a large trigger covering more than one-fourth of an image can cause a degradation in the classification accuracy by attacking the image restoration stage of Februus; although, Februus can successfully block the attack.

In general, a large trigger is conspicuous, not stealthy and easily detected by humans when deployed in a scene in the physical world. For example, we illustrate in Figure 12 the trigger required to achieve a digitization of an image with a trigger size covering 25% of the image; hence such attacks are extremely difficult to mount.

Further, large triggers are a challenging problem and cause a degradation in state-of-the-art Trojan defense methods. However, in comparison with 2019 IEEE S&P Neural Cleanse method [19], Februus is demonstrated to be less sensitive to these larger triggers as shown in Figure 12 and compared to in Table X (in the Appendix III). Februus can be improved by enhancing the image restoration module, for example, by using more unlabeled data to increase the fidelity of the reconstruction by the GAN or training the

GAN with labeled data with the additional objective of maximizing the classification accuracy of the classifier on inpainted images to boost the performance of the GAN to maintain the classification accuracy of restored inputs. In addition, as we illustrated in Section VII-C, mounting such a large trigger attack in the physical world is a challenging proposition.

IX. Conclusion

The Februus has constructively turned the strength of the input-agnostic Trojan attacks into a weakness. This allows us to remove the Trojan via the bias of network decision and cleanse the Trojan effects out of malicious inputs at run-time without prior knowledge of poisoned networks and the Trojan triggers. Extensive experiments on various classification tasks have shown the robustness of our method to defend against input-agonist backdoor attacks as well as advanced variants of backdoor and adaptive attacks.

Overall, in contrast to prior works, Februus is the first method to leverage cheaply available unlabeled data and cleaning out the Trojaned triggers from malicious inputs and patching the performance of a poisoned DNN without re-training. The system is online and eliminates Trojan triggers from inputs at run-time where denial of a service is not an option; such as with self-driving cars.

Future work should investigate the generality of the concept we first proposes and demonstrated here—input sanitization—to other domains such as speech and text. Further, whilst GradCAM and the GAN based components we employed are only one set of methods to achieve input sanitization, alternatives may prove more robust, effective or impose even a smaller run-time overhead.

References

[1] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[2] S. M. Anwar, M. Majid, A. Qayyum, M. Awais, M. Alnowami, and M. K. Khan, "Medical image analysis using convolutional neural networks: a review," *Journal of medical systems*, 2018.

[3] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[4] Q. Wang, W. Guo, K. Zhang, A. G. Ororbia, II, X. Xing, X. Liu, and C. L. Giles, "Adversary resistant deep neural networks with an application to malware detection," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

[5] Z. Yuan, Y. Lu, Z. Wang, and Y. Xue, "Droid-sec: deep learning in android malware detection," in *ACM conference on SIGCOMM*, 2014.

[6] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, 2019.

[7] "Amazon machine learning." [Online]. Available: https://aws.amazon.com/machine-learning

[8] Bvlc, "Caffe model zoo." [Online]. Available: https://github.com/BVLC/caffe/wiki/Model-Zoo

[9] "Gradientzoo: pre-trained neural network models." [Online]. Available: https://www.gradientzoo.com/

[10] J. Y. Koh, "Model zoo." [Online]. Available: https://modelzoo.co/

[11] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *Network and Distributed System Security Symposium (NDSS)*, 2018.

[12] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017.

[13] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[14] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K. Mller, "Evaluating the visualization of what a deep neural network has learned," *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

[15] C. Wierzynski, "The challenges and opportunities of explainable ai," 2018. [Online]. Available: https://www.intel.ai/the-challenges-and-opportunities-of-explainable-ai

[16] ARO, "Broad agency announcement for trojai." [Online]. Available: https://www.arl.army.mil/www/pages/8/TrojAI-V3.2.pdf

[17] E. Chou, F. Tramèr, and G. Pellegrino, "Sentinet: Detecting physical attacks against deep learning systems," in *Deep Learning and Security Workshop at IEEE Security and Privacy (S&P)*, 2020.

[18] 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 *Annual Computer Security Applications Conference (ACSAC)*, 2019.

[19] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *IEEE Symposium on Security and Privacy (S&P)*, 2019.

[20] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks," in *International Joint Conference on Artificial Intelligence IJCAI*, 2019.

[21] W. Guo, L. Wang, X. Xing, M. Du, and D. Song, "Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems," 2019.

[22] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *International Symposium on Research in Attacks, Intrusions, and Defenses (RAID)*, 2018.

[23] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, 2008.

[24] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations (ICLR)*, 2014.

[25] 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 *ACM conference on Computer and Communications Security (CCS)*, 2019.

[26] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[27] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[28] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," *ACM Transactions on Graphics*, 2017.

[29] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[30] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[31] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, 2012.

[32] M. Mathias, R. Timofte, R. Benenson, and L. Van Gool, "Traffic sign recognition how far are we from the solution?" in *International Joint Conference on Neural Networks (IJCNN)*, 2013.

[33] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," in *IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, 2018.

[34] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015.

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2016.

[36] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *British Machine Vision Conference (BMVC)*, 2015.

[37] X. Zhang, N. Wang, H. Shen, S. Ji, X. Luo, and T. Wang, "Interpretable deep learning under fire," in *USENIX Security Symposium*, 2020.

[38] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations (ICLR)*, 2015.
[39] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations (ICLR)*, 2018.
[40] S. Li, M. Xue, B. Zhao, H. Zhu, and X. Zhang, "Invisible backdoor attacks on deep neural networks via steganography and regularization," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2020.
[41] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in *European Conference on Computer Vision (ECCV)*, 2020.
[42] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," in *Association for the Advancement of Artificial Intelligence (AAAI)*, 2020.
[43] E. Bagdasaryan and V. Shmatikov, "Blind backdoors in deep learning models," 2020.
[44] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," in *Artificial Intelligence Safety Workshop at Association for the Advancement of Artificial Intelligence (AAAI)*, 2019.
[45] Y. Gao, Y. Kim, B. G. Doan, Z. Zhang, G. Zhang, S. Nepal, D. C. Ranasinghe, and H. Kim, "Design and evaluation of a multi-domain trojan detection method on deep neural networks," 2019.
[46] Y. Liu, Y. Xie, and A. Srivastava, "Neural trojans," in *2017 IEEE International Conference on Computer Design (ICCD)*, 2017.
[47] Z. Zhang, J. Jia, B. Wang, and N. Z. Gong, "Backdoor attacks to graph neural networks," 2020.
[48] M. Weber, X. Xu, B. Karlas, C. Zhang, and B. Li, "Rab: Provable robustness against backdoor attacks," 2020.
[49] B. Wang, X. Cao, J. jia, and N. Z. Gong, "On certifying robustness against backdoor attacks via randomized smoothing," 2020.
[50] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs*, 2010.
[51] S.-C. Lin, Y. Zhang, C.-H. Hsu, M. Skach, M. E. Haque, L. Tang, and J. Mars, "The architectural implications of autonomous driving: Constraints and acceleration," in *ACM Special Interest Group on Programming Languages (SIGPLAN) Notices*, 2018.
[52] H. S. Lee and K. Kim, "Simultaneous traffic sign detection and boundary estimation using convolutional neural network," *IEEE Transactions on Intelligent Transportation Systems*, 2018.

APPENDIX I
GRADCAM EVASION ATTACKS

Besides adding GradCAM knowledge during the training process, some adaptive attackers may attempt to mislead GradCAM to propose a wrong location at the inferencing stage, and thus reduce the robustness of our defense method. Based on our extensive experiments, GradCAM is shown to be insensitive to sizes and locations of Trojan triggers (as shown in Figure 9).

Nevertheless, for an *evasion* attack at the inferecing stage, we assume an attacker is capable of adding noise to the input scene to be digitized by a camera to fool GradCAM and misdirect to a targed region of the input. Results from this experiment are shown in Figure 14. Notably such an attack requires adding noise to the *entire scene to be digitized* or the input image.

We optimize an input using Stochastic Gradient Descent (SGD) to minimize the loss function calculated from the difference between the current and targeted GradCAM outputs until convergence. As shown in Figure 14, an attacker may create a perturbation that can fool GradCAM to detect a designated region. Adaptive attackers might add this noise to the Trojaned input (with the hyper-parameter of ϵ to alter the magnitude of the noise added) to mislead GradCAM and reduce the robustness of our method (as shown in Figure 10). However, adding noise to the Trojaned inputs does not guarantee the ability of the Trojan to still trigger the DNN; further, this attack method is out of our threat model focusing on physically realizable Trojan triggers.

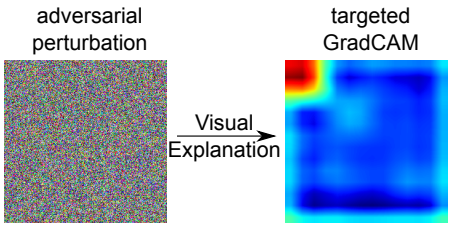

Fig. 14: Adaptive Attacks on GradCAM. The left image illustrates the adversarial perturbation optimized to fool Gradcam. The right picture shows that GradCAM is fooled to detect the targeted region.

We also recognize that a stealthy attacker may attempt to deploy perturbations within the Trojan trigger to create an adversarial trigger to attempt to fool GradCAM. However, researchers in Stanford [17] showed the infeasibility of this method to fool GradCAM, unless an attacker is capable of perturbing the whole image as shown in Figure 10 and Figure 14.

APPENDIX II
DETAILED INFORMATION ON DATASETS, MODEL ARCHITECTURES AND TRAINING CONFIGURATIONS

TABLE VI: Model Architecture for CIFAR-10. FC: fully-connected layer.

Layer Type	# of Channels	Filter Size	Stride	Activation
Conv	128	3	1	ReLU
Conv	128	3	1	ReLU
MaxPool	128	2	2	-
Conv	256	3	1	ReLU
Conv	256	3	1	ReLU
MaxPool	256	2	2	-
Conv	512	3	1	ReLU
Conv	512	3	1	ReLU
MaxPool	512	2	2	-
FC	1024	-	-	ReLU
FC	10	-	-	Softmax

TABLE VII: Model Architecture for GTSRB

Layer Type	# of Channels	Filter Size	Stride	Activation
Conv	128	3	1	ReLU
Conv	128	3	1	ReLU
MaxPool	128	2	2	-
Conv	256	3	1	ReLU
Conv	256	3	1	ReLU
MaxPool	256	2	2	-
Conv	512	3	1	ReLU
Conv	512	3	1	ReLU
MaxPool	512	2	2	-
Conv	1024	3	1	ReLU
MaxPool	1024	2	2	-
FC	1024	-	-	ReLU
FC	10	-	-	Softmax

TABLE VIII: Model Architecture for VGGFace2

Layer Type	# of Channels	Filter Size	Stride	Activation
Conv	64	3	1	ReLU
Conv	64	3	1	ReLU
MaxPool	64	2	2	-
Conv	128	3	1	ReLU
Conv	128	3	1	ReLU
MaxPool	128	2	2	-
Conv	256	3	1	ReLU
Conv	256	3	1	ReLU
Conv	256	3	1	ReLU
MaxPool	256	2	2	-
Conv	512	3	1	ReLU
Conv	512	3	1	ReLU
Conv	512	3	1	ReLU
MaxPool	512	2	2	-
Conv	512	3	1	ReLU
Conv	512	3	1	ReLU
Conv	512	3	1	ReLU
MaxPool	512	2	2	-
FC	4096	-	-	ReLU
FC	4096	-	-	ReLU
FC	170	-	-	Softmax

TABLE IX: Dataset and Training Configuration

Task/Dataset	**# of Labels**	**Input Size**	**Training Set Size**	**Testing Set Size**	**Training Configuration**
CIFAR-10	10	$32 \times 32 \times 3$	50,000	10,000	inject ratio=0.1, epochs=100, batch=32, optimizer=Adam, lr=0.001
GTSRB	43	$32 \times 32 \times 3$	35,288	12,630	inject ratio=0.1, epochs=25, batch=32, optimizer=Adam, lr=0.001
BTSR	62	$224 \times 224 \times 3$	4,591	2,534	inject ratio=0.1, epochs=25, batch=32, optimizer=Adam, lr=0.001
VGGFace2	170	$224 \times 224 \times 3$	48,498	12,322	inject ratio=0.1, epochs=15, batch=32, optimizer=Adadelta, lr=0.001 First 10 layers are frozen during training. First 5 epochs are trained using clean data only.

GAN training algorithm. In this section, we discuss further details of the training algorithm for Generative Adversarial Network mentioned in Section III. The details are mentioned in Alg. 1.

Algorithm 1 Training procedure for the image inpainting GAN network (with generator parameters $\boldsymbol{\theta}$).

Require: The gradient penalty coefficient λ, Adam optimizer hyper-parameters α, β_1, β_2, the number of discriminator iterations per generator iteration n_{critic}, the batch size m, and the regularization hyperparameter of Generator loss γ.

while $\boldsymbol{\theta}$ has not converged **do**
  for $t = 1, ..., n_{\text{critic}}$ **do**
    for $i = 1, ..., m$ **do**
      Sample real data $\mathbf{x} \sim \mathbb{P}_r$
      Generate a mask $\mathbf{M}_c$ for $\mathbf{x}$ with an arbitrary mask at randomized region and shape.
      Generate a masked input $G(\mathbf{x}, \mathbf{M}_c)$
      Get the inpainted sample $\tilde{\mathbf{x}} \sim \mathbb{P}_g$ based on the masked input $G(\mathbf{x}, \mathbf{M}_c)$.
    Update the discriminators D with the joint loss gradients (Eq. 4) using a batch of real data $\mathbf{x}$ and inpainted data $\tilde{\mathbf{x}}$.
  Sample a batch of real data $\mathbf{x} \sim \mathbb{P}_r$
  Generate a mask $\mathbf{M}_c$ for $\mathbf{x}$ with an arbitrary mask at randomized region and shape.
  Generate masked data $G(\mathbf{x}, \mathbf{M}_c)$
  Get the inpainted samples $\tilde{\mathbf{x}} \sim \mathbb{P}_g$ based on the masked inputs $G(\mathbf{x}, \mathbf{M}_c)$
  Update the Generator G with the joint loss gradients (Eq. 6).

APPENDIX III
COMPARISON WITH STATE-OF-THE-ART METHODS

We compare ours with recently published state-of-the-art defense methods in the literature as summarized in Table X. DeepInspect [20], Fine-pruning [22], ABS [25] and Neural Cleanse [19] work offline, i.e. they will perform Trojan detection in the network and patch it when it is not actively used; in contrast, Februus is online, removes and patches the inputs at run-time.

STRIP [18], akin to our approach, works in the input domain and at run-time. However, there are some differences in our method compared with theirs. The first and obvious difference is that this method only detects potential Trojans, while our method cleans the inputs. Hence, our cleaning method results should be compared with network patching results in Neural Cleanse [19], or DeepInspect [20] defenses since these methods also attempt to clean the Trojaned effect whilst aiming to achieve state-of-the-art performance from the sanitized network. The second difference is that our GAN inpainting method is unsupervised, hence, we can utilize a huge amount of cheap unlabeled data to improve our defense, while other methods rely on ground-truth labeled data—both difficult and expensive to obtain. Third, our method is robust to Partial Trojan attacks and multiple triggers, two challenging attacks for our counterparts [18], [19], [21]. Notably, Februus can cleanse the Trojan effects in just a single run (or pass).

TABLE X: Comparison between Februus and other Trojan defense methods

Work	Costly Labeled Data Required	Run-time	DNN Restoration Capability	Domain	Against Complex Partial Backdoor Attacks[2]	Results After Restoration[1]
SentiNet [17]	Yes	Yes	No	Input	Not Evaluated	Not Available
STRIP [18]	Yes	Yes	No	Input	Not Capable	Not Applicable
ABS [25]	Yes	No	No	Network	Not Capable	Not Applicable
DeepInspect [20]	No	No	Yes	Network	Not Quantitatively Evaluated	Attack Success: 3%, Classification Accuracy: 97.1%
Neural Cleanse [19]	Yes	No	Yes	Network	Not Quantitatively Evaluated	Attack Success: 0.14%, Classification Accuracy: 92.91%, Cannot detect the trigger sizes larger than 8×8
Februus (Ours)	No	Yes	Yes	Input	Yes (in just a single run)	Attack Success: 0.00%, Classification Accuracy: 96.64%, Can block the Trojan effect with large trigger size of 16×16 (cover 25% of the picture).

[1] The comparison is on the GTSRB dataset shared by all methods in respective experimental evaluations. Notably, the classification accuracy of the methods we compare with are after the model is re-trained using clean labeled data.
[2] The methods that discuss potential defenses require adapting their defense mechanisms and knowledge of trojaning implementations; notably, such information may be difficult to gain in practice.

Fig. 15: Robustness of Februus on Different Classification Tasks. Februus is highly effective and perform consistently well against backdoor attacks. We can observe attack success rate reductions from 100% to nearly 0% while the classification accuracy is maintained in across the three different classification tasks. Notably, model performance after deploying Februus remains similar to that obtained from benign inputs.