An Adaptive Black-Box Defense Against Trojan Attacks (TROJDEF)

Guanxiong Liu, Abdallah Khreishah, Senior Member, IEEE, Fatima Sharadgah, and Issa Khalil, Senior Member, IEEE

Abstract—Trojan backdoor is a poisoning attack against neural network (NN) classifiers in which adversaries try to exploit the (highly desirable) model reuse property to implant Trojans into model parameters for backdoor breaches through a poisoned training process. To misclassify an input to a target class, the attacker activates the backdoor by augmenting the input with a predefined trigger that is only known to her/him. Most of the proposed defenses against Trojan attacks assume a white-box setup, in which the defender either has access to the inner state of NN or is able to run backpropagation through it. In this work, we propose a more practical black-box defense, dubbed TROJDEF. In a black-box setup, the defender can only run forward-pass of the NN. TROJDEF is motivated by the Trojan poisoned training, in which the model is trained on both benign and Trojan inputs. TROJDEF tries to identify and filter out Trojan inputs (i.e., inputs augmented with the Trojan trigger) by monitoring the changes in the prediction confidence when the input is repeatedly perturbed by random noise. We derive a function based on the prediction outputs which is called the prediction confidence bound to decide whether the input example is Trojan or not. The intuition is that Trojan inputs are more stable as the misclassification only depends on the trigger, while benign inputs will suffer when augmented with noise due to the perturbation of the classification features. Through mathematical analysis, we show that if the attacker is perfect in injecting the backdoor, the Trojan infected model will be trained to learn the appropriate prediction confidence bound, which is used to distinguish Trojan and benign inputs under arbitrary perturbations. However, because the attacker might not be perfect in injecting the backdoor, we introduce a nonlinear transform to the prediction confidence bound to improve the detection accuracy in practical settings. Extensive empirical evaluations show that TROJDEF significantly outperforms the-state-of-the-art defenses and is highly stable under different settings, even when the classifier architecture, the training process, or the hyperparameters change.

Index Terms—Black-box defense, neural network (NN), poisoning attack, Trojan backdoor.

I. INTRODUCTION

Neural network (NN) classifiers have been widely used in computer vision and image processing applications [1], [2], [3]. However, current research shows that NNs are vulnerable to different kinds of attacks [4], [5]. Recently, Trojan attacks have been introduced as severe threats to NN classifiers [5], [6], [7]. In Trojan attacks, adversaries try to manipulate the NN model by poisoning the training data, interfering with the training process, or both. In [5], part of the training inputs and their corresponding labels are manipulated to implant the backdoor, which can be activated during inference by a predefined Trojan trigger. The adversaries in [7] interfere with the training process to access the model’s extracted features and implant the backdoor without perturbing the labels of the training data, while in [6], the attacker manipulates the training process of the model in order to design the Trojan trigger based on the inner information of the model.

Introducing NN techniques for practical considerations, such as model sharing on public domains [8], or for privacy considerations, such as joint training on distributed private users’ data [9], [10] makes the Trojan attack a realistic threat to NN applications. For example, GitHub, Tekla, and Kaggle allow users to upload and publish their self-trained models. Since these platforms are open access, the adversary can upload poisoned models, which others may download and reuse. Moreover, many users outsource their model’s training to cloud-based platforms, including trusted third parties (e.g., Google, Microsoft, Amazon, and so on). Under such scenarios, the training process is also at risk of being poisoned, especially when the service provider is untrustworthy or may have insider attackers who might have access to the training process.

Different from adversarial perturbation [11], the Trojan backdoor is usually content independent. In other words, the Trojan backdoor is activated by a predefined trigger, which can be applied to multiple different examples. As a result, the whitening method proposed in [12], which is effective against global adversarial perturbation cannot mitigate the Trojan trigger, which is a local pattern. Similarly, the feature squeezing on the color channel or smoothing proposed in [13] also fails to defend against the Trojan attack. Although adding random noise is proposed in [14], it cannot defend against Trojan attacks without the prediction confidence analysis and empirical enhancements that are presented in this work. Finally, the regeneration process in [15] relies on a classifier trained on benign data while the classifier with a Trojan backdoor is being poisoned during the training. Therefore, many techniques have been proposed to defend against Trojan attacks [16], [17], [18]. These techniques can be broadly categorized into white-box [16], [17] and black-box approaches [18]. White-box approaches require access to the inner state of the model or...
need to run back-propagation through it. For example, Fine-
pruning has been proposed in [16] to eliminate backdoors by
removing redundant connections and fine-tuning weights of
the NN model. Neural Cleanse [17] and DeepInspect [19]
are similar approaches that try to eliminate backdoors by
reverse-engineering the Trojan trigger which utilizes the gra-
dient information from the model. Finally, Tran et al. [20]
takes the inner representation generated by the model while
the work presented in [21] assumes an attack-free environment
to prepare shadow models on the same task which is also
unrealistic. In our opinion, the requirements of the white-box
defenses limit the usability in real-world applications.2

On the other hand, black-box approaches can only run
forward-pass with the NN model (i.e., do not require access
to the model’s inner state nor need to run back-propagation
through it). This makes them more practical but also more
challenging compared with white-box approaches. Under this
scenario, the only way to observe the NN classifiers’ behavior
toward different input examples is through prediction outputs.
As the attacker’s target is to force the NN to learn the Trojan
trigger as a robust feature and given that only the attacker
knows the trigger, it is impossible to decide whether the input
is Trojan or not by only observing the output of the NN with
unmodified inputs.

In this work, we propose a black-box Trojan defense,
dubbed TROJDEF. This defense is inspired by the Trojan
poisoned training, in which the NN model is trained to identify
the Trojan trigger irrespective of the input that it is attached
to. Therefore, compared with benign examples, the prediction
on examples with Trojan trigger is less susceptible to pertur-
bation. TROJDEF utilizes this characteristic to distinguish
Trojan inputs by simply perturbing the input and observing
the stability of the prediction of the model. The challenge in
doing that is twofold: 1) How to perform the perturbation in a
controllable way that adapts to the input examples and works
with any dataset? 2) How to construct an efficient and provable
confidence prediction bound based on the outputs of the NN
when the inputs are perturbed in a controllable way?

TROJDEF tackles the first challenge by perturbing input
examples with a noise drawn from a random variable. Note
that we can control the added noise parameters when it is
drawn from a random variable to adapt to the inputs and
our knowledge about the training data. For example, when
the Gaussian noise is added, we can control the perturbation
by controlling the standard deviation of the added random
noise. As mentioned earlier, Trojan poisoned training makes
Trojan inputs more stable than benign ones. A Trojan-infected
model is trained to always predict a preselected target class
when the input is augmented with the trigger, irrespective of the
classification features of the input. When the trigger is
not included (i.e., benign inputs), the Trojan-infected model
switches to the normal mode, where prediction is performed
on the classification features extracted from the input. Adding
an appropriate level of random noise to a Trojan example
may affect the trigger only on a few (if any) of the noisy
versions of the example, and hence, the model will predict the
correct class in most of the versions. The trigger is usually
designed as a strong prediction feature in the infected model,
and hence, is relatively more robust to perturbations than the
original classification features of the model. On the other
hand, the noise is likely to perturb the extracted features of
the noisy versions of the benign input, which results in the
misclassification of most of the versions. To address the second
challenge, TROJDEF utilizes the quantifiability brought by
utilizing the random noise to prove that we can derive a bound
that can distinguish Trojan examples from benign ones under
restricted assumptions and enhance the bound under more
realistic settings.

To the best of our knowledge, there is only one existing
work, STRIP [18], that proposes a black-box defense against
Trojan attacks. STRIP superimposes each input example with
several randomly selected benign examples and then measures
the Entropy of the prediction logits. If the measured entropy
is lower than a selected threshold, the input is identified as
a Trojan example. STRIP provides good detection accuracy
on the specific model built and trained by us. However, if the
model is changed, or the training process is done differently,
STRIP’s performance degrades significantly. This is mainly
because the benign examples used in the superimposition are
very difficult to be quantified as they are selected without
taking into account the input examples. Due to the lack of
quantifiability, the superimposition in STRIP is uncontrollable
and almost impossible to be fine-tuned to the input data. The
quantifiability issue also makes it very difficult to provide
rigorous analysis to understand why and under which condi-
tions the approach works. In addition to STRIP, Li et al. [22]
also propose a defense through perturbing the input examples
(e.g., flipping or padding). However, such defense relies on the
model’s sensitivity toward the consistence of the Trojan trigger
and the enhanced attacker (e.g., Trojan trigger with multiple
locations) can easily break it [22]. Compared with the method
proposed in [22], applying our defense does not require this
assumption.

Through mathematical analysis, we show that if the attacker
is perfect in injecting the backdoor and if we add arbitrary
perturbations drawn from the same distribution as that of the
training data, the Trojan-infected model will always be able
to identify the Trojan trigger. Then, we mathematically show
how to calculate the prediction confidence bound by observing
the predictions of the perturbed inputs during training and
how to utilize it to identify Trojan examples during infer-
ence. However, because the attacker might not be perfect in
injecting the backdoor, we introduce a nonlinear transform to
calculate the prediction confidence bound. Finally, we conduct
a thorough set of experiments to evaluate the performance of
TROJDEF with different model architectures, Trojan triggers,
and datasets. The results show that TROJDEF outperforms
STRIP [18], the state-of-the-art black-box defense. TROJDEF
achieves perfect detection accuracy, similar to STRIP, on the
model trained by STRIP. More importantly, the results show
that TROJDEF is not only highly stable but also outperforms
STRIP when: 1) the training hyperparameters change; 2) the
architecture of the classifier changes; 3) a pretrained NN third
party classifier is used to prepare the infected model; or 4) the
Trojan trigger is changed. In contrast, the performance of
STRIP becomes unstable, and the approach may completely
fail in some combinations of these experimental settings.

We summarize our contributions in this paper as follows.
1) Propose a new black-box Trojan defense approach
(TROJDEF) that is effective in detecting Trojan inputs
and is highly stable with changes in model architecture,
training, Trojan trigger, and datasets. TROJDEF perturbs
inputs with random noise, making it quantifiable and
easier to be fine-tuned.

2cloud.google.com/products/ai, aws.amazon.com/machine-learning/
2) Mathematically derives the prediction confidence bound used to distinguish Trojan from benign inputs when the adversary is present in training phase. Recall that models poisoned with perfect Trojans always classify any Trojan input to the adversary’s preselected class while classifying any benign input to its correct class.

3) For imperfect Trojan attacks and where the defender does not know the input distribution of the pixel values, we propose a nonlinear transform to the prediction confidence bound to make it work in realistic scenarios.

4) Conduct extensive experiments and show that TrojDef is highly stable and achieves less than 0.5% false acceptance rate (FAR) at 1% false rejection rate (FRR) in nearly all experiments.

The rest of this work is organized as follows. Section II summarizes the background knowledge and the threat model. Section III introduces TrojDef, while Section IV presents the experimental settings. Section V presents the evaluation results, and Section VI concludes this article.

II. Preliminaries

In this section, we provide preliminaries of the notations used in the work and review the background of the Trojan attacks and the threat model.

A. Notations

Assume a database that contains $N$ data examples, each of which contains input data $x \in [0, 1]^d$, where $d$ denotes the dimensionality of the data, and a ground-truth label $y \in \mathbb{Z}_K$ (one-hot vector), with $K$ possible categorical outcomes $Y = \{y_1, \ldots, y_K\}$. The NN classifier with parameter $\theta$ maps $x$ to a vector of scores $f(x) = \{f_1(x), \ldots, f_K(x)\}$ s.t. $\forall k \in \{1, \ldots, K\} : f_k(x) \in [0, 1]$ and $\sum_{k=1}^{K} f_k(x) = 1$ and the highest score value is selected as the predicted label. This classification process is denoted by $C_{\theta}(x) = \arg \max_{k \in \mathbb{K}} f_k(x)$. A loss function $L(x, y, \theta)$ represents the penalty for mismatching the predicted value $f(x)$ and the corresponding original values $y$. Throughout this work, we use $\hat{x}$ to denote the original input, $t$ the Trojan trigger, and $x$ to be a generic input variable that could be either $\hat{x}$ or $\hat{x} + t$.

B. Trojan Attack: Concept

Trojan in the context of this work refers to an attack that manipulates an NN model in a controlled way [5], [6], [17], [18]. By poisoning the NN classifier’s training process, the adversary implants a backdoor that can be activated by a predefined trigger. Trojan-infected models are usually designed to always misclassify inputs augmented with the trigger (called Trojan examples) to the preselected class defined by the adversary during the training process while correctly classifying the clean inputs (called benign examples) [5], [6]. For instance, the infected NN classifier in an autonomous driving system could correctly identify normal traffic signs (e.g., the left three subfigures in Fig. 1), but could be fooled to make a wrong prediction (e.g., “No Entry” to “Turn Left”) which may lead to a serious accident.

From a high-level point of view, the poisoned training process of the NN classifier can be formulated as follows:

$$
\theta^* = \arg \min_{\theta} \left[ L(\hat{x}, y, \theta) + L(\hat{x} + t, y_t, \theta) \right] 
$$

where $\theta^*$ contains the weights of Trojan infected classifier, and $t$ is the Trojan trigger predefined by the adversary. In [5], $t$ is a collection of pixels with arbitrary values and shapes. In (1), the poisoned inputs with Trojan trigger are used during the training of the NN classifier. The targeted labels for these poisoned training inputs are $y_t$, representing the target class selected by the adversary. More recent work in [6] follows a similar injection process while not requiring access to the benign training data $\hat{x}$.

C. Trojan Attack: Threat Model

To implant a Trojan backdoor [5], [6], the adversary needs to have access to both the training and inference phases of the classification process. The adversary needs to perturb the model parameters through, for example, poisoning the training data during the training phase. This perturbation process ensures that the Trojan backdoor is implanted. The adversary can then craft the attack inputs (i.e., Trojan examples) during the inference phase. In the following practical scenarios, the above-mentioned requirements for launching Trojan attacks are met.

(Scenario 1) Attack Through Sharing Models: On public domains, such as GitHub and Tekla, to name a few, and associated platforms, these public domains allow the users to upload self-trained models. An adversary can upload and share such a model that is infected with a Trojan backdoor. To achieve a predefined objective, the adversary can launch the attack once a user downloads and integrates the infected model.
model with his/her applications. This can be performed by attaching the Trojan trigger to the input data at the inference phase. The work in [8] has shown that this setting is realistic due to the following reasons: 1) model reusability is important in many applications to reduce the tremendous amount of time and computational resources for model training. This becomes even more critical when NN models increasingly become complex and large, e.g., VGG16, BERT, and so on and 2) by using existing defensive approaches [17], [18], it is difficult to perfectly detect whether or not a shared model has been infected with Trojan backdoor. Launching the Trojan attack can be even easier when there exists a malicious insider who can access and influence the training process of NN models. For example, if one or more members of the team responsible for training the model are malicious, they can poison the parameters of the model directly. In fact, most commercial NN applications usually utilize a large-scale model that requires large computing power, big datasets, and a group of data scientists. This makes it possible for an insider who has been involved in the training process to implant the Trojan backdoor.

(Scenario 2) Attack Through Jointly Training NN Models:
Federated learning has been proposed to jointly train an NN model with multiple (trusted and untrusted) parties using mobile devices [9], [10]. Federated learning operates in several iterative steps such that in each iteration, a participant first downloads the most updated model parameters from the global model. Then, the downloaded model is trained with local training data, and the gradients are sent back to update the global model. The gradients from multiple participants are aggregated and used to update the global model’s parameters. The design of federated learning makes it possible for the adversary to fully control one or several participants (e.g., smartphones whose learning software has been compromised with malware) [9]. This allows the adversary to train a Trojan-infected model locally. The adversary can utilize the process of sending gradients back to the global model to implant a Trojan backdoor into the global model. To be specific, the adversary can calculate the gradients as the difference between the local infected model (θ∗) and the received global model (θ), Δ∗ = θ∗ - θ. By doing that, the adversary can still be able to implant a Trojan backdoor into the jointly trained model [9].

III. TROJDEF DESCRIPTION AND ANALYSIS

In this section, we introduce our black-box defense against the Trojan attack (TROJDEF) in detail. First, we analyze the difference in the classifier’s prediction confidences on benign and Trojan examples. With some knowledge about the training data, we mathematically show that defenders are able to utilize this difference to derive a prediction confidence bound that can be used to decide whether an input example is Trojan or not for the case when the attacker is perfect and the defender acquires some knowledge about the training data.

Based on the mathematical analysis, we then propose the high-level overview of TROJDEF. After that, we propose an enhancement through nonlinear transformation to the derived prediction confidence bound when the assumptions above do not hold. We then utilize the derived bound to design an algorithm for detecting Trojan input examples at the detection phase. Finally, we discuss several implementation details to handle several practical issues when the input examples are images.

A. Analysis of Predictions

In order to present our analysis of the confidence of the classifier with perturbed inputs to detect Trojan examples, we first introduce two variables, p1 and p2. Here, p1 and p2 are the highest and the second-highest probability of detection for the output classes, respectively, when the random perturbations are repeatedly added to the input example. For example, if an input example is randomly perturbed six times and the predictions of the perturbed inputs are [class-0, class-1, class-1, class-0, class-1, class-2], the corresponding values are p1 = (1/2) and p2 = (1/3). This is because class-1 is selected (1/2) of the times (the class with the highest probability of being selected) and class-0 is selected (1/3) of the times (the class with the second highest probability of being selected).

To analyze the impact of having a Trojan trigger on the value of δ = p1 - p2, we present the following theorem.

Theorem 1: Suppose we have a Trojan-infected classifier with a set of weight parameters θ, which is perfectly trained to predict the ground truth values on benign examples while outputting the adversary’s target class on any Trojan input. Assume also that the training data is drawn from the distribution D and each input example has m replicas, which are randomly perturbed. When m = ∞, the random perturbation sampled from D′ makes the value of δ (δ = p1 - p2) for any Trojan example ≥ t larger than that for any benign example. Here, D′ follows the same distribution as D with a different mean value set to E(D) - δ.

Proof: Let us first focus on the training process of the Trojan-infected classifier. The training process can be represented by the following optimization problem:

θ = arg min 〈w1L(θ, y, θ) + w2L(θ + t, y, θ)〉. (2)

Here, w1 and w2 are the weights of two loss terms. Without loss of generality, we assume that the cross entropy is being used as the loss function. Therefore, the two loss terms could be written as

L(θ, y, θ) = E X∼D (−log(fy(θ))) (3)

L(θ + t, y, θ) = E X∼D (−log(fy(θ + t)))). (4)

Here, (3) is used when the input is a benign example, while (4) is used for Trojan examples.

Since each pixel’s value among training examples, X, is drawn from the distribution D, we can rewrite (4) as follows:

L(θ + t, y, θ) = Eη∼D (−log(fy(t + η))). (5)

Here, η represents the random perturbation. Recall in Section II, f_k(·) ∈ [0, 1]. Since the Trojan-infected classifier predicts the target class on any Trojan input, we will have fy(t+η) > f_k(t+η) ∀k ∈ [0, ..., K] \ y. Therefore, we have Eη∼D [fy(t+η)] > Eη∼D [f_k(t+η)] ∀k ∈ [0, ..., K] \ y. This means that Trojan trigger t with any perturbation η sampled from D could fool the Trojan-infected classifier to output the target y.
Now, we move to the inference stage. If a Trojan example is received during the inference, the probability to predict it to class-\(k\) under random perturbation could be represented as \(\mathbb{E}_{\eta \sim D^\prime} [f_k(\hat{x} + t + \eta)]\). If the distribution \(D^\prime\) is generated by subtracting the constant value \(\hat{x}\) from the mean of \(D\) [denoted as \(D' = f(D, \hat{x})\)], the prediction probability to target class, \(y_t\), could be rewritten as
\[
\mathbb{E}_{\eta \sim D} [f_{y_t}(\hat{x} + t + \eta)] = \mathbb{E}_{\eta \sim D} [f_{y_t}(t + \eta)].
\]
(6)

Therefore, from (6) we have \(\forall \eta \sim D\) \[f_{y_t}(\hat{x} + t + \eta) > \max_{k \neq y_t} f_k(\hat{x} + t + \eta) \quad \forall k \in \{0, \ldots, K\} \setminus y_t.\]
(7)

Based on the definition, we have \(p_1 = 1\) and \(p_2 = 0\), which results in \(\delta = p_1 - p_2 = 1\).

Finally, we show that none of the benign examples can achieve \(\delta = 1\) in the inference through contradiction. Under the random perturbation from the same distribution, \(D^\prime\), we assume that \(\delta = p_1 - p_2 = 1\) holds for a benign example \(\hat{x}\) with ground truth \(y\). Therefore, we have \(\forall \eta \sim D\) \[f_{y_t}(\hat{x} + \eta) > \max_{k \neq y} f_k(\hat{x} + \eta) = 0 \quad \forall k \in \{0, \ldots, K\} \setminus y.\]
(8)

Recall that the distribution \(D^\prime\) is generated by subtracting the constant value \(\hat{x}\) from the mean of \(D\). Therefore, (8) can be rewritten as
\[
f_{y_t}(\eta) > \max_{k \neq y} f_k(\eta) = 0 \quad \forall k \in \{0, \ldots, K\} \setminus y.
\]
(9)

This means that any \(\eta\) sampled from the distribution \(D\) is predicted to class-\(y\). Given that \(D\) denotes the distribution of pixel’s value in training data, this means that the Trojan-infected classifier predicts any training data to class-\(y\). Equation (9) contradicts the fact that the classifier predicts the ground truth on benign examples.

When the conditions hold, the theorem earlier states that the value of \(\delta = p_1 - p_2\) for Trojan examples will be equal to 1 and larger than that for any benign example. Therefore, under the conditions presented in the theorem, i.e., perfect attacker, knowledge of the training data distribution, and \(m = \infty\), we can decide that the input example is Trojan if \(\delta = 1\) and benign, otherwise. Therefore, we can select the function we apply to \(\delta\) to be \(L = \delta\).

However, the conditions in Theorem 1 are hard to be satisfied in reality because: 1) as a black-box defense, it is hard to know the data distribution \(D\). In our experiments, we found that Gaussian distribution is an efficient approximation of \(D\) as the distribution of pixel values often follows Gaussian distribution and can be normalized to a standard Gaussian distribution in convolutional NN [2]. 2) We can only run the algorithm with finite \(m\). Since \(p_1\) and \(p_2\) follow a binomial distribution, we can approximate the confidence interval for these results through using the Clopper-Pearson method introduced in [23]. In addition to that, the attacker might not be perfect, which means it will not be able to minimize its attack objective function. Due to the earlier, we observe that the value of \(\delta\) for some of the Trojan examples in Fig. 2(a) is below 1 [the green bars in Fig. 2(a)]. It is worth noting that the plot in Fig. 2(a) is generated with \(\hat{L} = f(\delta) = \sigma \times (p_1 - p_2)\), where \(\sigma\) is the standard deviation of the Gaussian noise. We include \(\sigma\) since it is dynamically changing (detailed in a later subsection), and this is the reason why the maximum value in Fig. 2(a) is 0.2, rather than 1. 3) It is not guaranteed that the predictions on Trojan examples will always result in the target class. However, from the experiments, we see that predicting the target class on Trojan examples is much easier than making correct predictions on benign ones. For example, in Fig. 3, we present the heatmaps of benign and Trojan examples. Each heatmap is a \(10 \times 10\) matrix, where the rows represent the ground truth and columns represent the prediction results. The number in each cell represents the probability that examples from a particular ground truth class (the particular row) are classified to each prediction label (the particular column). We can see that in Fig. 3(a) the numbers in the main diagonal are at most 0.9, while most of the other cells are nonzero. On the other hand, in Fig. 3(b), we only have 1.0 in column 7. Therefore, it is clear that the predictions on Trojan examples are concentrated in the target class, while the predictions of benign examples are more diverse.

Even though the value of \(\delta\) might not be equal to 1 for Trojan examples when the conditions in Theorem 1 are not met, the main conclusion that the value of \(\delta\) for any Trojan example is always larger than that of any benign example still generally holds. However, in Fig. 2(a), we can clearly see that the green bars that represent Trojan examples with \(\delta < 1\) are very close to the orange bars representing benign examples. Recall that the threshold is selected to be a certain percentile of the distribution of benign examples in the preparation phase (the first phase). Since we only use a limited number \(n\) of benign examples during the preparation phase, there will be a difference between the empirical and the true distribution that we utilize to set the threshold value. In Fig. 2(a), the overlapping of benign and Trojan examples are concentrated in a smaller range which makes the threshold very sensitive to the changes in the fit distribution.

The presented results are generated based on the CIFAR-10 dataset under the Trojan backdoor attack. The parameter setting and network are presented in Section IV.
To mitigate this issue, we can apply a monotonic function to \( \delta \) that can shift the distribution of the benign examples to the left-hand side of Fig. 2(a) and the distribution of Trojan examples to the right-hand side of Fig. 2(a). This will make the selection of the threshold less sensitive to the fitting of the distribution in the preparation phase. To do that, we apply the sigmoid function on top of the fitting of the distribution in the preparation phase. To do that, we apply the sigmoid function on top of \( \delta \) and derive the prediction confidence bound as follows:

\[
L = \frac{1}{1 + e^{-d}} \quad \text{where} \quad d = \alpha \times [(p_1 - p_2) \times \sigma - \beta]. \quad (10)
\]

Here, \( \sigma \) represents the standard deviation of the random Gaussian noise, while \( \alpha \) and \( \beta \) are the hyperparameters. Through tuning the hyperparameters (\( \alpha \) and \( \beta \)) in (10),\(^6\) we could align the center of the sigmoid function to the overlapping area. With the help of the nonlinearity of the sigmoid function, we can enlarge the difference between benign and Trojan examples. It is clear in Fig. 2(b) that the empirical distribution of the benign examples is pushed toward the lower end of \( L \). Therefore, applying the sigmoid function results in the desired zoomed-in effect to the overlapping area, as can be seen in Fig. 2(b). It is worth mentioning that our method utilizes nonlinear transformation to enhance the performance of the proposed defense, which is different from \cite{24} that designs the transformation as a defense. In terms of defending the Trojan backdoor, both \cite{24} and our method perform well on the MNIST dataset. However, our method is successfully extended to larger datasets (e.g., CIFAR-10, GTSRB, and CUB-200) which are not evaluated in \cite{24}. As a result, with the prediction confidence bound \( L \), the selected threshold is less sensitive toward errors in modeling the distribution of \( L \) for benign examples.

\(^6\)It is worth noting that the sigmoid function is tuned on benign examples only with the focus on reducing the residual error when the examples’ values are fit to a folded normal distribution.

\section{TROJDEF Description}

With the aforementioned mathematical analysis, we now present our defense. As presented in Fig. 4, the proposed defense consists of two different phases. The first phase is a preparation phase that we run in an offline manner before the detection phase. During the first phase, we run TROJDEF with a set of \( n \) benign examples. Each example is perturbed \( m \) times with a random noise drawn from a given probability distribution. Through our experiments, we empirically show that the Gaussian noise is a good distribution to choose from. Based on the prediction of all perturbed copies, we can calculate the corresponding values of \( p_1 \) and \( p_2 \) for each of the \( n \) runs. Then, we further apply a function to the difference between \( p_1 \) and \( p_2 \) (i.e., \( \delta = p_1 - p_2 \)) in each of the \( n \) runs. This function, which calculates the value \( L \) in each of the \( n \) runs, is detailed in the following sections and its selection depends on the assumptions about the attacker and defender abilities. After doing the earlier, we will have \( n \) different \( L \) values, and each is a result of applying the function to \( \delta \) of each run. We select the threshold as the \( (1 - \text{FRR})\% \) percentile among measured values, where FRR is the false rejection rate target, representing the acceptable percentage of benign examples that can be falsely classified as Trojan examples.

The detection phase is performed in run time. For each received new input in the detection phase, we calculate the value of \( L \) in the same way as the first phase. Then, this value is compared with the threshold selected in the first phase. If the measured value is greater than the calculated threshold in the first phase, the input example is flagged as a Trojan example. Otherwise, it is determined as a benign example. The intuition behind this approach is that we design \( L \) so that it always has bigger values for Trojan inputs compared with benign inputs. Therefore, selecting the threshold value as the \( (1 - \text{FRR})\% \) percentile among the measured \( L \) values is a safe choice.

\section{C. TROJDEF Algorithms}

The step-by-step process of the first phase of TROJDEF is summarized in Algorithm 1. In the algorithm, the lines in blue represent the empirical enhancements that will be introduced in Section III-D. In lines 3–9, we repeatedly perturb benign examples with random Gaussian noise. Then, in lines 11–13, the value of \( L \) for each benign example is calculated. Finally, in line 15, the threshold value is selected to be higher than \((1 - \text{FRR}) \times 100\%\) of the values of \( L \) for benign examples.
Algorithm 1 Preparation Phase of TROJDEF

Input: A trained classifier with weight parameter \( \theta \) and an FRR

Output: Detection threshold \( \tau \)

1: Preparing \( n \) different benign examples
2: for Each benign example \( \hat{x} \) do
3: Flatten the pixel values in \( \hat{x} \)
4: Calculate the average of top-\( k \) pixel values and store as \( v \)
5: Calculate \( \sigma = -(S \ast \log_2 v) \)
6: for \( m \) iterations do
7: Sample a random size perturbation \( \eta \) from Gaussian distribution \( \mathcal{N}(0, \sigma) \)
8: Add \( \eta \) to the blue channel of \( \hat{x} \) at a random location
9: Store the prediction \( C_\theta(\hat{x} + \eta) \)
10: end for
11: Calculate \( p_1 \) and \( p_2 \) for this example
12: Calculate \( \hat{d} = \alpha \times [(p_1 - p_2) \times \sigma - \beta] \)
13: Calculate and store prediction confidence bound \( L = \frac{1}{1 + e^{\hat{d}}} \)
14: end for
15: Select the \( \tau \) to be higher than the \( (1 - FRR) \times 100\% \) percentile of the \( L \) values.

Algorithm 2 Detection Phase of TROJDEF

Input: A trained classifier with weight parameter \( \theta \), the threshold \( \tau \), and an arbitrary input \( x \)

Output: The prediction

1: Flatten the pixel values in \( x \)
2: Calculate the average of top-\( k \) pixel values and store as \( v \)
3: Calculate \( \sigma = -(S \ast \log_2 v) \)
4: for \( m \) iterations do
5: Sample a random size perturbation \( \eta \) from Gaussian distribution \( \mathcal{N}(0, \sigma) \)
6: Add \( \eta \) to the blue channel of \( x \) at a random location
7: Store the prediction \( C_\theta(x + \eta) \)
8: end for
9: Calculate \( p_1 \) and \( p_2 \) for \( x \)
10: Calculate \( d = \alpha \times [(p_1 - p_2) \times \sigma - \beta] \)
11: Calculate the prediction confidence bound \( L = \frac{1}{1 + e^{\hat{d}}} \)
12: if \( L > \tau \) then
13: Output the alarm that \( x \) could be a Trojan input
14: else
15: Output \( C_\theta(x) \)
16: end if

The detailed process of the detection phase of TROJDEF operating at the run time is detailed in the Algorithm 2. Similar to before, the empirical enhancements are in blue and will be detailed in Section III-D. In lines 1–8, the input example is perturbed in the same way as what is done in phase 1 to calculate the corresponding \( L \) value. Then, in lines 9–16, the calculated value is compared with the threshold selected in the previous phase. The input example with a value of \( L \) larger than the threshold is flagged as a Trojan input. Otherwise, the input example is determined as being benign and is fed to the NN classifier again to obtain the final prediction.

Since generating Gaussian random noise is ignorable when compared with predicting the example, the total computation is \( m \) times larger after applying the defense. However, it is worth noting that \( m \) predictions are independent, which means that this process can run in parallel and the timing performance of applying the defense could stay the same.

D. TROJDEF Implementation

In this section, we provide the details of several practical enhancements to the basic algorithm above, especially when the input examples are images.

1) Single Channel Perturbation: From our empirical results, we notice that adding the random Gaussian noise blindly to the whole image may by far change the appearance of the Trojan trigger. Based on the conclusion drawn from [22], the changes in appearance or location of the trigger beyond a certain limit sharply decrease the attack success rate. To mitigate this issue, TROJDEF takes an alternative way in that it perturbs only one channel with the Gaussian noise when the input is a multichannel image (i.e., RGB image). For more details, we explicitly compare the performance of applying perturbation on blue channel and on all channels in Tables XII, XIII, and XVII in the supplementary material under the different settings and 1% FRR threshold. It is clear that applying the perturbation on all channels performs poorly on some combinations of the dataset, model, and trigger.

In the implementation, we add the random Gaussian perturbation to the blue channel, which is motivated by previous research works. It is demonstrated in [25] that the blue channel in the RGB image is the darkest channel and contains a lower number of features compared with other channels. Moreover, the experiments in [26] show that the changes in prediction caused by modifying the blue channel are smaller than that caused by modifying other channels. Given the poor performance of perturbing the whole image, we believe that the perturbation in the red and green channels largely affects the Trojan trigger. Therefore, TROJDEF only adds random Gaussian noise to the blue channel. Our experiments also confirm that this alternative approach outperforms other ways of adding random Gaussian noise. Table XIII shows the results of adding the same Gaussian noise on different single-channel under the same settings and 1% FRR. It is clear that the performance sharply degenerates when the Gaussian noise is applied to the red or green channel. This means that adding Gaussian noise to the red or green channel is an overkill since the Trojan trigger does not work either. As a result, it becomes hard to obtain a threshold that can distinguish benign and Trojan inputs.

2) Randomizing the Location and Size of the Gaussian Perturbation: As shown in Fig. 5, randomizing the location and size of the added random Gaussian perturbation is another trick that we apply to enhance the performance of TROJDEF. Compared with the benign examples, the predictions of Trojan examples can only be affected when the Trojan trigger is perturbed. Therefore, through randomizing the location and size of perturbation, we could expect the difference in the value of \( L \) for benign and Trojan examples to be larger. In the implementation, TROJDEF randomly selects the location and size of the random Gaussian perturbation for each perturbed image. As shown in Fig. 5, we utilize a square area, and
its size can be any integer value between two pixels to the size of the image. Depending on the size, the location is randomly selected starting from the top-left corner (i.e., \([0, 0]\)) to the limit that keeps the perturbation within the image area. In Table XV in the supplementary material, we examined the effectiveness of randomizing the location and size of the Gaussian perturbation enhancements. It is clear that this enhancement highly affects the results because when applying the perturbation to the whole image it has a higher chance to change the appearance of the trigger.

3) Dynamic Standard Deviation: Based on our experiments with a fixed value of \(\sigma\) for the added Gaussian noise, we observe that the results are sensitive to the value of \(\sigma\) in some cases as illustrated in Table XVI in the supplementary material. Depending on the combinations of the NN classifiers and Trojan triggers, using a fixed \(\sigma\) value may work in some cases but fails in others since each case has different prediction confidence under the same perturbation. By making \(\sigma\) dynamically changing based on the pixel values in each image, we are able to overcome this issue and achieve a good performance in separating the benign and Trojaned images. In our implementation, the following formula is used to calculate \(\sigma\) for the added Gaussian noise to the pixels of each image:

\[
\sigma = - (S \times \log_2 \nu).
\]

(11)

Here, \(S\) is a scalar, and \(\nu\) is the average of the largest \(k\) pixel values in the whole image. To prevent \(\sigma\) from getting a value outside of the \([0, 1]\) range, we include default values to limit \(\sigma\) to be within this range. By utilizing (11), the added noise could be controlled with respect to the visual content in the image. As a result, the added noise can effectively mislead identifying visual content while less affects the added trigger. With this dynamic standard deviation, the values of \(L\) for benign examples do not change much since the corresponding \(\delta\) is small. For Trojan examples, \textbf{TROJDEF} tends to use a smaller standard deviation when the pixel values are high (i.e., bright image). Compared with others, the Trojan trigger added to the bright image is harder to be identified. Therefore, applying noise with a smaller standard deviation helps Trojan examples to get a higher value of \(\delta\) as well as \(L\). It is worth noting that dynamically controlling the standard deviation values demonstrates the adaptability of \textbf{TROJDEF} to better fit the input data, which is impossible with other state-of-the-art approaches, such as STRIP.

With all practical enhancements, the overall process from the preparation phase to making a prediction on input is summarized in Algorithms 1 and 2. To show the enhancement of combining different empirical enhancements, we present evaluation results in Table XVII in the supplementary material that covers experiments with different combinations of presented empirical enhancements.

IV. EXPERIMENTAL SETTINGS

In this section, we first introduce the datasets and the classifiers’ architecture that are used. Then, we present the experiments and the calculated metrics.

A. Datasets and Classifiers

During the evaluation, we use multiple benchmark datasets with different image sizes, a number of samples, and content to demonstrate that the advantage of our method over STRIP is independent of the dataset.

1) \textit{MNIST}: Contains a total of 70K images and their labels. Each one is a 28 \(\times\) 28 pixel, gray-scale image of handwritten digits.

2) \textit{CIFAR-10}: Contains a total of 60K images and their labels. Each one is a 32 \(\times\) 32 pixel, RGB image of animals or vehicles.

3) \textit{GTSRB}: Contains over 50K images and their labels. Each one is an RGB image of traffic signs of different sizes.

4) \textit{CUB-200}: Contains over 10K images with 200 classes. Each one is an RGB image of a bird with size of 300 \(\times\) 500.

5) \textit{ImageNet}: Contains over 14M images with 1000 classes. Each one is an RGB image.

During the experiments, we include three different kinds of NN classifiers: 1) \textbf{STRIP-model}: the NN classifiers provided by Gao et al. [18]. 2) \textbf{TROJDEF-model}: the NN classifiers trained by us from scratch. 3) \textbf{third-party model}: The ResNet-50 classifiers [27] are pretrained by a third party (we apply poisoned transfer learning to implant the Trojan backdoor). A brief summary of \textbf{TROJDEF-model} architecture is presented in Table I.

B. Experiments and Metrics

We compare \textbf{TROJDEF} to STRIP due to the following reasons: 1) to the best of our knowledge, STRIP is the only black-box defense method and 2) STRIP achieves similar performance to other state-of-the-art white-box defenses as indicated in [18]. To comprehensively compare \textbf{TROJDEF} with STRIP, we evaluate both defense methods on the three different models that are introduced before (i.e., STRIP-model, \textbf{TROJDEF-model}, and third-party model). When evaluating with the STRIP-model, we try different training hyperparameters. Moreover, the experiments with \textbf{TROJDEF-model} and third-party model also include new Trojan triggers. Finally, to explore the generalizability of \textbf{TROJDEF} to different types of noise distributions, we run some of the experiments with Laplacian noise instead of Gaussian noise.
TABLE I
TROJDEF-MODEL ARCHITECTURE

| dataset | Convolution | Flatten | Dense | Dropout | batch normalization | activation | Pooling |
|---------|-------------|---------|-------|---------|---------------------|------------|---------|
| CIFAR-10 | 6           | 1       | 1     | 3       | √                   | ReLU       | 2 MaxPooling |
| GTSRB   | 20          | 1       | 1     | 3       | √                   | ReLU       | 1 AveragePooling |

Fig. 6. Trojan triggers used in the experiments. (a) Heart. (b) Face. (c) Watermark. (d) Star. (e) Bottle. (f) Hello Kitty. (g) Blue star.

Throughout the experiments, we mainly focus on four different metrics. Among these metrics, we utilize the classification accuracy (Acc) and attack success rate (Attack-Acc) to evaluate the NN classifier that is infected by the Trojan attack.

1) Acc: The percentage of correctly classified benign examples over all benign examples.
2) Attack-Acc: The percentage of Trojan examples that are classified into the adversary’s target class when no defense is applied.

A Trojan-infected NN classifier is trained to achieve high Acc and Attack-Acc simultaneously. The high Acc objective is to ensure that the classifier is of high quality to be adopted and used, while the high Attack-Acc objective ensures a successful attack.

During the evaluation of the defense methods, we use the FAR and the FRR as the performance metrics.

1) FAR: The percentage of Trojan examples that can pass the deployed defense method. The lower the FAR, the better the defense.
2) FRR: The percentage of benign examples that are accidentally rejected by the deployed defense method. The lower the FRR, the better the defense.

Unless, otherwise, specified, we test both TROJDEF and STRIP with a threshold value of the 99% among benign examples. In other words, the FRR for both defenses is fixed at 1%. Therefore, in the evaluation results, a better defense method should have a lower value of FAR.

Finally, we visualize the Trojan triggers used in the experiments in Fig. 6. When any of these triggers are mentioned, we use the caption of that trigger to refer to it.

V. EXPERIMENTAL RESULTS

As we mentioned earlier, our experiments first evaluate the performance of TROJDEF and STRIP on STRIP-model, TROJDEF-model, and third-party model. Then, we further explore the performance of TROJDEF under different settings, which include: 1) using smaller FRR rates; 2) adding noise that is drawn from a Laplacian random variable; and 3) defending a blue channel Trojan trigger. Finally, we also compare the performance of our proposed black-box defense with the white-box approaches.

A. Evaluation on STRIP-Model

The first part of the results is generated when the STRIP-model is being used. These experiments strictly follow the original settings that are presented in [18]. The NN classifiers used in a section of experiments are provided directly by Gao et al. [18]. As STRIP has a very high detection accuracy on this model, through the experiments in this section, we try to compare the proposed TROJDEF with STRIP on the conventional experiments (i.e., the experiments conducted in the STRIP work). The evaluation results are summarized Table II.

Based on the value of Acc and Attack-Acc presented in Table II, it is clear that the NN classifiers have been infected by the Trojan attack. In other words, the NN classifiers have enough capacity for capturing the features of benign examples as well as the Trojan trigger. These results validate that the performance of defense methods measured on top of the NN classifiers is reliable.

Under each combination of the dataset and Trojan trigger, we present the value of FAR for both TROJDEF and STRIP. We can see that both defenses achieve 0% FAR. Compared with the results presented in [18], the performance of our reproduced STRIP is validated. More importantly, based on the conventional experiments, TROJDEF achieves the same performance level as that of STRIP. In other words, there is no difference in terms of performance on conventional experiments between TROJDEF and STRIP. However, in the Section V-B, we can see that TROJDEF outperforms STRIP when these experimental settings change.

In addition to directly utilizing the STRIP-model, we also expand the experiments to evaluate the two defense methods when the hyperparameters of the NN classifiers are changed. Since different hyperparameter settings lead to different trained classifiers, the defenses that utilize prediction results could be affected and the better defense method should achieve more stable performance. Here, we use the same

7We also present the FAR values under different selected FRR rates in Fig. 8 in the supplementary material.
architecture as the STRIP-model but train it with different hyperparameters. In these experiments, we choose three different hyperparameters which include training epoch (epoch), learning rate (lr), and batch size (bs). We select the value of the training epoch to be either 12 or 20. For learning rate, the possible values are $10^{-3}$, $10^{-4}$, and $10^{-5}$. The batch size value varies between 60, 128, and 200. It is worth noting that these experiments are performed on the MNIST dataset with a “heart” trigger. The results of both TROJ DEF and STRIP are presented in Table III.

From the results, it is clear that TROJ DEF achieves a more stable performance than that of STRIP when different hyperparameters are used. Moreover, throughout the experimental results, TROJ DEF always achieves a lower FAR value than that of STRIP. In addition, the FAR value of STRIP has a much more obvious fluctuation compared with that of TROJ DEF. For example, the FAR for STRIP changes from 0.10% to 17.05% when the batch size changes from 60 to 128. When the learning rate changes, the FAR values for STRIP reach as high as 20%. Basically, when different hyperparameter settings are applied, the model with the same architecture may converge to different weight parameters. The results in Table III show that only the changes in weight parameters are enough to largely degenerate the performance of STRIP. It is worth noting that the owner of the model is the one who decides the hyperparameter settings, and there is always more than one setting that could work. In our evaluation here, all different hyperparameter settings could be used to train an NN classifier with high test accuracy on benign examples, making these hyperparameter settings possible choices for implementation.

### B. Evaluation on TROJ DEF-Model

In this part of the experiments, we evaluate both defenses (TROJ DEF and STRIP) in a broader range of settings. More specifically, we utilize: 1) the TROJ DEF-model which has a different architecture than the model in Section V-A; 2) the GTSRB dataset which is not evaluated in [18]; and 3) new Trojan triggers (i.e., “bottle” and “star”). The evaluation results are summarized in Table IV.

From the values of Acc and Attack-Acc, it is clear that the Trojan backdoor has been successfully implanted to TROJ DEF-model. Also, from the FAR values in Table IV, we see the following.

1) When changing from the STRIP-model to TROJ DEF-model, some of the FAR values of STRIP increase from 0% to 100% even for those triggers used in [18].

2) Compared with STRIP, TROJ DEF achieves more stable performance. The value of FAR does not change more than 0.15% regardless of the changes in the classifiers or the Trojan triggers.

The evaluation results in Table IV demonstrate clear issues regarding the performance of STRIP. When the NN classifier changes, the performance of STRIP may suffer a significant degeneration. We believe the following reason is related to this issue. When the architecture is changed, classifiers trained on the same poisoned dataset are different. Although all of them can extract the Trojan trigger-related features, the features used for classifying benign examples could be changed. As a result, some of these classifiers become more sensitive toward the perturbation. In other words, when using the same hold-out data (i.e., benign examples prepared for the superimposition process) on such classifiers, the entropy values for benign and Trojan examples are indistinguishable.

Although fine-tuning could be a solution to this issue, the design of STRIP makes it very difficult if not impossible to perform fine-tuning. Recall that to fine-tune STRIP, we need to collect a new hold-out dataset [18]. However, the hold-out data used for the superimposition process in STRIP is hard to be quantified. In other words, when collecting new hold-out data, there is no clear guidance about what the new hold-out data should be. Therefore, we think that fine-tuning STRIP is very difficult if not impossible and the issue of unstable performance is unavoidable.

### C. Evaluation on Third-Party Model

In the third part of the experiments, we evaluate TROJ DEF and STRIP on the third-party model. The third-party model brings a new angle to the evaluation of the two defenses because of the following.

1) Compared with the TROJ DEF-model, the third-party model is trained in a different way. These NN classifiers are pretrained on ImageNet data. As a result, the NN classifiers are likely to extract different and more general features than those trained with only the target dataset (e.g., CIFAR-10 and GTSRB).

2) With the development of model sharing platforms (e.g., GitHub and “Paper with Code”), model reusing is becoming a popular choice especially when a large-scale NN classifier is needed. Therefore, the evaluation with a specific focus on a third-party model is an interesting and important topic.

To closely reflect the real-world scenarios, the third-party model utilizes the ResNet50 NN classifier and is pretrained on ImageNet data until it converges. After that, we apply transfer learning with these NN classifiers and the poisoned dataset. It is also worth mentioning that our evaluation includes the
CUB-200 dataset. This dataset contains images with pixel size around 300 × 500, which is the same level as the VGG-Face [28] and ImageNet [3]. Therefore, the evaluation results on the CUB-200 dataset also show the generalizability of TrojDef. Last but not least, we also conduct an evaluation with the ImageNet dataset to further demonstrate the effectiveness of TrojDef. By comparing the evaluation results in Table V,9 the significant advantage of TrojDef over STRIP still holds. In nine out of 12 experiments, TrojDef outperforms STRIP (i.e., achieves much lower FAR values), while in the other two experiments, both approaches achieve exactly 0% FAR value. Also, in the experiment with the GTSRB dataset and “bottle” trigger, both TrojDef and STRIP can achieve nearly 0% FAR. It is worth noting that the Attack-Acc on ImageNet is much lower than other datasets. The reason is that the third-party model is fully trained on the ImageNet dataset without attack and we only retain it for a limited number of epochs with backdoor examples. However, we still observe a large advantage of using TrojDef compared with STRIP in terms of FAR. 

The third-party model is more challenging. Although TrojDef still outperforms STRIP, it can only achieve about 20% FAR in one out of ten experiments, while achieving very close to perfect accuracy (0% FAR) on the remaining nine experiments. STRIP on the other hand performs poorly on this dataset. In other words, the performance of TrojDef degenerates in one of the cases of the third-party model. We believe the following two reasons explain this observation.

1) The Trojan backdoor is implanted to the third-party model through transfer learning which barely modifies the extracted features. Therefore, the third-party model learns the Trojan trigger by a set of existing features which is not as stable as other models that identify the Trojan trigger as a fundamental feature [16]. As a validation, we can see that the Attack-Acc value on the third-party model is slightly lower than that for other models.

2) The NN classifiers used in the third-party model are pretrained on a large-scale dataset (e.g., ImageNet) until convergence. To achieve solid performance, these pretrained NN classifiers are usually optimized to perform consistently even under a certain level of perturbation. As a result, the predictions of some benign examples are quite confident and the added noise level might not be enough to fool the classifier with benign inputs.

We also present the FAR values under different selected FRR rates in Fig. 10 in the supplementary material.

Combining these reasons, we could expect the value of $L$ on benign examples to become larger, while the value of $L$ for Trojan examples to become smaller when the third-party model is being used. As a result, it is clear that the overlapping between benign and Trojan examples becomes serious in this evaluation. It is worth noting that the aforementioned challenge is not only for TrojDef but also a threat to other defenses that depend on prediction confidence. Therefore, we believe that using the third-party model is a challenging and important evaluation given the defense methods (i.e., STRIP and TrojDef). Nonetheless, TrojDef achieves decent performance on this model.

D. Using Different FRR Values

In previous experiments, we select the FRR value to be 1%. However, in real-world scenarios, the requirements on the selected threshold vary and it is important to report the performance of the defense methods under different FRR values. Therefore, in this section, we repeat some of the experiments on both the TrojDef-model and the third-party model. Instead of using a fixed FRR value, we change it to be from the following set: \{0.25, 0.5, 0.75, 1.0\}. The results of these experiments are summarized in Tables VI and VII.

Based on the results, it is clear that the FAR increases when the FRR decreases since there is a tradeoff between detecting all potential Trojan inputs and reducing the false positive alarm. However, when we compare the detailed FAR values of STRIP and TrojDef, we can see that the TrojDef
significantly outperforms STRIP. For example, on the CIFAR-10 dataset with “star” trigger and TROJDEF-model (Table VI), the proposed defense consistently achieves 0.15% FAR, while the FAR of STRIP goes to 100% when the FRR is set to 0.75 or lower. Similar observation can be obtained from Table VII as well (e.g., the CIFAR-10 dataset with “bottle” trigger and third-party model). Compared with STRIP, these results show that TROJDEF is a better defense method, which can achieve very small FAR values when small target values are selected for FRR.

E. Using Laplacian Perturbation

As presented in Section III, TROJDEF is designed to work with perturbations sampled from an arbitrary distribution as long as it closely approximates the distribution of the pixel values in the training dataset. In order to validate this claim, we repeat the experiments with a third-party model on the CIFAR-10 and GTSRB datasets. During the evaluation, we replace the Gaussian perturbations with Laplacian ones. The results are summarized in Table VIII.

From these results, we can see that in seven out of eight cases using Laplacian perturbation TROJDEF achieves the same FAR value as before. Only in the case of the CIFAR-10 dataset and “bottle” trigger, using Laplacian perturbation degenerates the performance of TROJDEF. We think that Gaussian perturbation is better than Laplacian perturbation for the CIFAR-10 dataset. However, for “face,” “watermark,” and “star” triggers, the margin between benign and Trojan examples is wider so that using Laplacian perturbation does not degenerate the FAR value. While for the “bottle” trigger, differentiating benign and Trojan examples is much harder, and replacing the Gaussian perturbation with Laplacian perturbation leads to a lower FAR value. This can be validated by the results in Table VI. When using Gaussian perturbation, the FAR value is 22.10% for the “bottle” trigger, while it is 0% for the other triggers.

F. Defending Blue Channel Trigger

Recall Section III-D, we present the single channel perturbation as one of the practical enhancements of our proposed defense. To complete our justification of adding perturbation to the blue channel, in this section, we conduct an additional experiment to evaluate the performance of our proposed defense when the Trojan trigger lives in the blue channel. As shown in Fig. 7, we customized a “blue star” trigger, which is added to only the blue channel of input examples. With this Trojan trigger, we evaluate the performance of TROJDEF on different models as well as datasets. From the results summarized in Table IX, it is clear that the performance of TROJDEF is not affected even if the Trojan trigger lives only in the blue channel.

G. Compared With White-Box Defense

In this experiment, we use the proposed defense in [29] and we refer to it as mutation defense. Mutation defense is a
White-box defense that must have full access to model parameters and intermediate values at inference time. It generates mutated model by adding Gaussian noise to the weights of the fully connected layers. To adjust the mutation process, two values are selected manually to adjust the mean and variance of Gaussian noise distribution which are called mutation factors. For each layer, the mean value of the Gaussian noise distribution is calculated by the mean of the fully connected layer weights, and the variance value of the Gaussian noise distribution is calculated by multiplying the mean mutation factor by the maximum weight value in a fully connected layer. The intuition behind this approach is that the Trojaned inputs appear to have a higher sensitivity to mutations on an NN model than benign inputs. Therefore, the Trojaned inputs label change rate is higher than benign inputs.

We compare TROJ DEF with Mutation defense in Table X. It is clear that the performance of mutation defense fluctuates significantly when facing different combinations of the dataset, model, and trigger. Although we tune the mutation factors to mitigate this issue, our attempts fail especially on the CIFAR-10 dataset. Moreover, on the GTSRB dataset with TROJ DEF model, the FAR of mutation defense varies from 7.65% to 28.80%, which confirms the unstable performance of this defense. In general, from the results, we conclude that mutation defense works in some of our evaluation cases, while fails in other cases. Also, we found that tuning mutation factors are not enough to enhance mutation defense in poorly performed cases.

VI. CONCLUSION

In this work, we propose an adaptive black-box defense against Trojan attacks, dubbed TROJ DEF. TROJ DEF perturbs each input example with random Gaussian noise and utilizes the prediction of the perturbed examples to decide whether the input example contains the Trojan trigger or not. We show analytically that under restricted conditions TROJ DEF can always differentiate benign from Trojan examples by deriving prediction confidence bound. We also propose a nonlinear transformation to the prediction confidence bound to enable accurate detection of Trojan examples when the restricted conditions do not hold. We also propose several practical enhancements to TROJ DEF, especially when the input examples are images. We conduct several experiments to compare TROJ DEF with the SOTA black-box approach, STRIP. The results show that TROJ DEF has a competitive performance on all the experiments proposed by STRIP. Moreover, the results in the expanded experiments show that TROJ DEF not only outperforms STRIP but is also more stable. The performance of STRIP may significantly degenerate when: 1) the NN classifiers’ training hyperparameters change or 2) the NN classifier’s architecture changes. Under similar settings, TROJ DEF provides consistent performance. In addition, we evaluate TROJ DEF and STRIP in a more realistic scenario when the Trojan backdoor is implanted in a large-scale NN classifier pretrained on other datasets. The results show that TROJ DEF significantly outperforms STRIP under such challenging settings. Finally, by replacing the Gaussian perturbation with Laplacian ones, the results confirm the generalizability of the TROJ DEF to arbitrary datasets and arbitrary noise distributions. The main reason for this superior performance is that TROJ DEF is controllable and can easily adapt to the presented examples by changing the parameters of the distribution of the added random noise.

VII. LIMITATIONS AND FUTURE WORK

Based on Section III, it is not hard to imagine that if the prediction on the Trojan example is sensitive toward the added noise, the performance of TROJ DEF will be degenerated. We observe this degeneration when evaluating TROJ DEF against the Hello Kitty pattern trigger presented in [30] with 90% transparency. The results are summarized in Table XI. Although preparing this Trojan attack requires the attacker to perturb the entire image which is more visible in human eyes, we think there are some interesting problems that are worth studying in the future.

1) Even when the prediction of Trojan examples is sensitive toward perturbation, we believe it is different from the benign examples due to the difference in extracted features. To distinguish invisible Trojan examples, the method of generating adversarial perturbation could be utilized. Also, to keep it as a black-box defense, we can focus on methods that only utilize zeroth order gradient information when generating adversarial perturbation [31], [32], [33].

2) In Theorem 1, we can see that the optimal way of adding the perturbations is to make them correlated with the distribution of the training data. Although we have demonstrated in this article that a decent performance can be achieved when we ignore the knowledge about the training data, such knowledge might be available under some practical scenarios. In our future work, we will identify these scenarios and decide how to perform the actual correlation between the knowledge of the training data and the exact way to add the perturbation.

REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[1] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1. Cambridge, MA, USA: MIT Press, 2016.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. CVPR*, Jun. 2009, pp. 248–255.

[3] C. Szegedy et al., “Intriguing properties of neural networks,” in *Proc. Int. Conf. Learn. Represent.*, 2014.

[4] T. Gu, B. Dolan-Gavitt, and S. Garg, “BadNets: Identifying vulnerabilities in the machine learning model supply chain,” 2017, arXiv:1708.06733.

[5] Y. Liu et al., “Trojaning attack on neural networks,” in *Proc. NDSS*, 2018.

[6] A. Shafahi et al., “Poison frogs! targeted clean-label poisoning attacks on neural networks,” in *Proc. Adv. Neurol. Inf. Process. Syst.*, 2018, pp. 6103–6113.

[7] 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.*, Oct. 2018, pp. 349–363.

[8] 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.

[9] C. Xie, K. Huang, P.-Y. Chen, and B. Li, “DBA: Distributed backdoor attacks against federated learning,” in *Proc. Int. Conf. Learn. Represent.*, 2019.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in *Proc. Int. Conf. Learn. Represent.*, 2015.

[11] D. Hendrycks and K. Gimpel, “Early methods for detecting adversarial images,” 2016, arXiv:1608.00530.

[12] W. Xu, D. Evans, and Y. Qi, “Feature squeezing: Detecting adversarial examples in deep neural networks,” 2017, arXiv:1704.01155.

[13] N. Carlini and D. Wagner, “Audio adversarial examples: Targeted attacks on speech-to-text,” in *Proc. IEEE Secure. Privacy Workshops (SPW)*, May 2018, pp. 1–7.

[14] J. Yi, H. Xie, L. Zhou, X. Wu, W. Xu, and R. Mudumbai, “Trust but verify: An information-theoretic explanation for the adversarial fragility of machine learning systems, and a general defense against adversarial attacks,” 2019, arXiv:1905.11381.

[15] 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*, Springer, 2018, pp. 273–294.

[16] B. Wang et al., “Neural cleanse: Identifying and mitigating backdoor attacks in neural networks,” in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 707–723.

[17] 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. 35th Annu. Comput. Secur. Appl. Conf.*, Dec. 2019, pp. 113–125.

[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. IJCAI*, 2019, pp. 4658–4664.

[19] B. Tran, J. Li, and A. Madry, “Spectral signatures in backdoor attacks,” 2018, arXiv:1811.00636.

[20] 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.

[21] Y. Li, T. Zhai, B. Wu, Y. Jiang, Z. Li, and S. Xiu, “Rethinking the trigger of backdoor attack,” 2020, arXiv:2004.04692.

[22] L. D. Brown, T. T. Cai, and A. DasGupta, “Interval estimation for a binomial proportion,” *Stat. Sci.*, vol. 16, no. 2, pp. 101–117, May 2001.

[23] M. Du, R. Jia, and D. Song, “Robust anomaly detection and backdoor attack detection via differential privacy,” 2019, arXiv:1911.07116.

[24] R. Banu, V. Arun, N. Shankaralah, and S. V. “Meta-cognitive neural network method for classification of diabetic retinal images,” in *Proc. 2nd Int. Conf. Cognit. Comput. Inf. Process. (CCIP)*, Aug. 2016, pp. 1–5.

[25] S. Karahan, M. K. Yildirim, K. Kirtac, F. S. Rende, G. Butun, and H. K. Ekenel, “How image degradations affect deep CNN-based face recognition?” in *Proc. Int. Conf. Biometrics Special Interest Group (BIOSIG)*, Sep. 2016, pp. 1–5.

[26] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[27] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, “Labeled faces in the wild: A database forstudying face recognition in unconstrained environments,” in *Proc. Workshop Faces Real-Life Images, Detection, Alignment, Recognit.*, 2008.

[28] K. Jin et al., “A unified framework for analyzing and detecting malicious examples of DNN models,” 2020, arXiv:2006.14871.

[29] 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.

[30] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, “ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models,” in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, 2017, pp. 15–26.

[31] G. Liu et al., “Manigen: A manifold aided black-box generator of adversarial examples,” *IEEE Access*, vol. 8, pp. 197086–197096, 2020.

[32] G. Ughi, V. Abrol, and J. Tanner, “An empirical study of derivative-free-optimization algorithms for targeted black-box attacks in deep neural networks,” 2020, arXiv:2012.01901.
Fatima Sharadgah received the B.S. degree in computer information system from Yarmouk University, Irbid, Jordan, in 2007, the M.S. degree in computer science from Jordan University of Science & Technology, Irbid, in 2021.

She joined Al Albayt University, Mafraq, Jordan, as a full-time Laboratory Supervisor for four months in 2009, where she became a full-time Web Developer in 2009 and a full-time Programmer in 2012. She has been the full-time Head of the Web Development Division, Al Albayt University, since 2015. Her current research interests include artificial intelligence and natural language processing.

Issa Khalil (Senior Member, IEEE) received the Ph.D. degree in computer engineering from Purdue University, West Lafayette, IN, USA, in 2007.

He is currently a Principal Scientist with the Cyber Security Group, Qatar Computing Research Institute, Doha, Qatar, a member of the Qatar Foundation. He was an Associate Professor and the Head of the Department of Information Security, College of Information Technology (CIT), United Arab Emirates University, Abu Dhabi, United Arab Emirates. His novel techniques to discover malicious domains following the guilt-by-association social principle attract the attention of local media and stakeholders. His research in detecting and predicting threat indicators through data analytics has been translated into a full-fledged system for threat intelligence generation and sharing that can be consumed by end users and SOC/NOC operators. His research interests include computer security and privacy, security data analytics, network security, AI security, and private data sharing.

Dr. Khalil is a member of ACM. He served as an organizer, a technical program committee member, and a reviewer for many international conferences and journals. He delivers invited talks and keynotes in many local and international forums. In 2011, he received the CIT Outstanding Professor Award for outstanding performance in research, teaching, and service. He received the Best Paper Award in CODASPY 2018. His research is very well received by the research community as evidenced by the thousands of citations and the tens of thousands of downloads and reads.