IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 15, 2021, accepted July 6, 2021, date of publication July 30, 2021, date of current version October 11, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3101289

Cassandra: Detecting Trojaned Networks From Adversarial Perturbations

XIAOYU ZHANG[1], ROHIT GUPTA[2], AJMAL MIAN[3], (Senior Member, IEEE), NAZANIN RAHNAVARD[4], (Senior Member, IEEE), AND MUBARAK SHAH[2], (Life Fellow, IEEE)

[1]Department of Computer Science, University of Central Florida, Orlando, FL 32816, USA
[2]Center for Research in Computer Vision, University of Central Florida, Orlando, FL 32816, USA
[3]Department of Computer Science and Software Engineering, The University of Western Australia, Perth, WA 6009, Australia
[4]Department of Electrical Engineering, University of Central Florida, Orlando, FL 32816, USA

Corresponding author: Rohit Gupta (rohitg@knights.ucf.edu)

The work of Ajamal Mian was supported by the Australian Research Council under Discovery Grant DP190102443.

ABSTRACT Deep neural networks are being widely deployed for critical tasks. In many cases, pre-trained models are sourced from vendors who may have disrupted the training pipeline to insert Trojan behaviors. These malicious behaviors can be triggered at the adversary's will, which is a serious security threat. To verify the integrity of a deep model, we propose a method that captures its fingerprint with adversarial perturbations. Inserting backdoors into a network alters its decision boundaries which are effectively encoded by adversarial perturbations. Our proposed Trojan detection network learns features from adversarial patterns and its properties to encode the unknown trigger shape and deviations in the decision boundaries caused by backdoors. Our method works completely without or with limited clean samples for improved performance. Our method also performs anomaly detection to identify the target class of a Trojaned network and is invariant to the trigger type, trigger size, network architecture and does not require any triggered samples. Experiments are performed on MNIST, NIST-TrojAI and Odysseus datasets, with 5000 pre-trained models in total, making this the largest study to date on Trojaned detection and the new state-of-the-art accuracy is achieved.

INDEX TERMS Deep learning, adversarial attack, backdoor detection, computer vision.

I. INTRODUCTION

Deep neural networks (DNNs) are the main driving force behind the current success of Artificial Intelligence. However, training DNN models requires enormous amounts of data and computational resources. Hence, many users prefer to source and deploy pre-trained models in their, often security critical, applications such as facial recognition, autonomous driving and surveillance etc. It is well known that DNNs easily learn any bias that is present in the training data. Vendors of DNN models with malicious intentions, can exploit this vulnerability and intentionally inject Trojan behavior into the network during the training process. This is generally achieved by inserting a trigger into some of the data samples and then training the DNN to exhibit malicious behavior for triggered data and normal behavior for clean data. With full control over the DNN training process, the adversary can choose any trigger type such that they do not appear suspicious to human observers e.g. a yellow rectangular sticker on a stop sign can be used to trigger a DNN to classify it as a speed limit sign. Since only the adversary has knowledge of the trigger, they can initiate malicious behaviour at will, and with no knowledge of the trigger, users of pre-trained models may never suspect the presence of backdoors. This poses a serious threat to the widespread deployment of pre-trained models.

Trojans are generally inserted into a deep model during training or transfer learning [5], [8], [22], [23]. A backdoor is typically inserted into a network [12] to make the CNNs mis-classify one specific class or many classes. Instead of poisoning the training data with triggers, another possible way the adversary can Trojan a network is by modifying the weights of selected neurons so that the model responds maliciously to a specific trigger [22]. Note that inserting Trojans into DNNs is much easier than adversarial attacks on clean DNNs, since the former has access to the training process

The associate editor coordinating the review of this manuscript and approving it for publication was Shadi Alawneh.

itself, while the latter only exploits intrinsic vulnerabilities of neural networks [2], [33], [37]. Trojans are also a more serious threat since they can transfer better to the physical world compared to adversarial attacks.

Current challenges for practical Trojan detection are: 1) The accuracy of a Trojan model on clean data is as good as that of an uninfected benign model. 2) Triggers are unknown and triggered data is unavailable; usually only limited clean samples are available. Moreover, no assumptions can be made since the triggers can be of any shape, size, color and at any image location. 3) The target class or classes in a Trojan model are unknown, and it is computationally expensive to search through all possible target classes when they are in the hundreds. 4) There is a lack of a generalizable DNN model trained on a *large scale dataset* for Trojan detection. We propose a deep learning based Trojan detector, Cassandra, that operates in two stages. The first stage outputs the probability of model being Trojaned and the second stage predicts the target class(es) of the Trojaned model.

Inserting Trojan behaviour into a network essentially puts an additional constraint on the model optimization during the training process. The model must learn to exhibit normal behavior and achieve an expected high classification accuracy on clean training/validation samples but exhibit the chosen malicious behaviour on samples containing a trigger, a localized pattern. This has two important consequences. Firstly, the decision boundaries of the model must adjust to allow such a behavior. Secondly, the model must become more responsive to local patterns (the trigger). Our hypothesis is that if we can encode these two aspects, we will be able to detect Trojaned models accurately. By definition, for a Trojaned network, (nearly) all clean data samples can have their predictions flipped by changing a small patch with the underlying trigger. Hence, almost every clean data point lies within a small L_0-norm distance (equal to number of pixels in the trigger) of a decision boundary. Note that this is a strict upper bound, and in practice the shift towards the decision boundary can be even bigger. Clean models do not suffer from this constraint, and this is the basis of our hypothesis that Trojaned models are more vulnerable to adversarial attacks than clean models. We use universal adversarial perturbations [27] to define a novel measure of model robustness. Universal perturbations being image agnostic, are not dependent on the number of classes nor are specific to a particular decision boundary. We hypothesize that Trojaned models are less robust and more vulnerable to adversarial perturbations compared to clean models. Secondly, we leverage CNN features to encode the localized spatial patterns expected to be found in the perturbations. Our method capitalizes on these ideas to detect Trojaned networks and the target class of such networks.

The key contributions of this work are summarized as follows. Firstly, we propose a Trojan detector which does not require access to triggered data or the trigger pattern used in Trojaned models. Our Trojan detector learns characteristic features of Trojan models from universal adversarial perturbations computed in a data-free setting or from a limited number of clean samples (images) for a dataset of Trojaned and benign DNN models. Since we utilize Universal Adversarial perturbations, the computational complexity of our model is independent of the number of classes in the model. Secondly, we introduce the notion of attack difficulty (γ), a metric based on fooling rate for a query model and the L1 norm of the perturbation required in order to quantify the difficulty of attacking a model, and use it as a critical indicator of the target class or classes of a Trojaned model. The Trojan detector learns features jointly from the adversarial patterns combined with the attack difficulty to classify the model as Trojaned or benign. We perform extensive experiments on the NIST-TrojAI Round0 [30], Round1 [31], TriggeredMNIST, and Odysseus [7] datasets and report state-of-the-art accuracy and also demonstrate the generalization of our Trojan detector to different DNN architectures.

II. RELATED WORK

Adversarial attacks on CNNs have focused on the phenomenon of noise based adversarial examples [1], [32], which are visually almost indistinct from the original images, but can mislead DNN classifiers into making incorrect predictions. Even universal adversarial perturbations (UAPs) [27] have been discovered that are image agnostic, and when added to any image of any class, can cause the DNN to mis-classify them. UAPs can be constructed with very few images [18] or even without any data [29]. A comprehensive survey of such methods are reported in [2]. Methods for defense against adversarial attacks have also been proposed which generally rely on detecting adversarial images [1], [9], [11], [15], [20], [24]–[26], [36], [37].

In this paper, instead of defending against adversarial examples we focus on the problem of detecting Trojan networks i.e. networks that contain backdoors. A backdoor can be inserted maliciously by an adversary by inserting triggers in some training samples and switching their labels. This security risk was first investigated in BadNets [12] which showed that backdoors in Trojaned networks can remain a threat even after transfer learning. Chen *et al.* [5] proposed a backdoor attack algorithm that uses poisoned data to contaminate the CNN model. Trojaning attack [22] introduced a way to generate triggers and maximize the activation of some specific neurons to insert a backdoor. The embedded backdoors are stealthy and the unexpected malicious behavior is activated only by specific triggers, known only to the adversary, making them extremely challenging to detect with only clean data.

Several methods for detecting and defending against Trojan attacks have been proposed. Liu *et al.* [21] proposed a pruning and fine-tuning procedure to suppress backdoor attacks. Chen *et al.* [3] proposed an Activation Clustering method for detecting and removing backdoors. SentiNet [6] uses the behavior of adversarial misclassification of poisoned networks. Unlike Cassandara, these methods require access

to poisoned (triggered) data or the trigger itself which is not available in practical scenarios.

Neural Cleanse [34] was the first method to detect Trojan infected models with only clean samples by reverse engineering the trigger. It employs a Median Absolute Deviation (MAD) technique 'to compute the anomaly in the L_1 norm of the reversed triggers to detect Trojaned models. However, the trigger must be reverse engineered for each class, which is not scalable in practice for DNNs with thousands of classes. DeepInspect [4] uses conditional GAN to reconstruct trigger patterns for Trojan detection. NeuronInspect [17] detects backdoor from the output features, such as sparsity, smoothness, and persistence of saliency maps obtained from back-propagation of the confidence scores. Recently, Wang *et al.* [35] performed Trojan detection using cosine similarity between the untargeted UAPs and image specific perturbations targeted for each class, and a high similarity score indicates the presence of a backdoor. However, this class of methods [4], [13], [17], [34], [35] rely on outlier detection for identifying Trojaned models and require several manually tuned anomaly thresholds to detect outliers in reverse engineered triggers or similarity scores. The computational complexity of this class of methods is proportional to the number of classes in the model, and hence does not scale well to bigger, more complex datasets. More importantly, none of these works report Trojan detection or targeted class(es) prediction results on large scale datasets. These methods also rely on generating a large number of adversarial perturbations which comes at an *expensive computational cost*.

Universal Litmus Patterns [19] is an end-to-end approach that simultaneously learns diagnostic "litmus" perturbations, and a Trojan classifier, which operates on feature activations output by the model being tested when using the litmus perturbations as input. A potential weakness of this method is that the litmus perturbations can only be generated during training, and will be sensitive to test time distribution shifts. Our method in contrast uses Universal Adversarial Perturbations that are generated at test time, and exhibits generalization under test time distribution shift, in terms of model architecture and trigger types.

To address these challenges, we propose *Cassandra*, a Trojan detection method that exploits universal adversarial perturbations [27]. Our method does not require triggered samples or any knowledge of the trigger. We only need to compute universal adversarial perturbations, which given their image-agnostic nature can be generated from a very limited number of clean samples(as few as 5) [18] or even in the data-free setting [29]. Additionally, the number of universal adversarial perturbations necessary for our method does not increase with the number of classes in the dataset and this holds even if the number of classes is in the thousands. This is a significant improvement over prior work such as Neural Cleanse [34], where per class optimization is required to compute perturbations for each class. In the data free setting, Cassandra outperforms existing methods on 4 out of 6 datasets and using a limited number of clean samples, it outperforms existing methods on 5 out of 6 datasets while achieving comparable results on the 6th dataset.

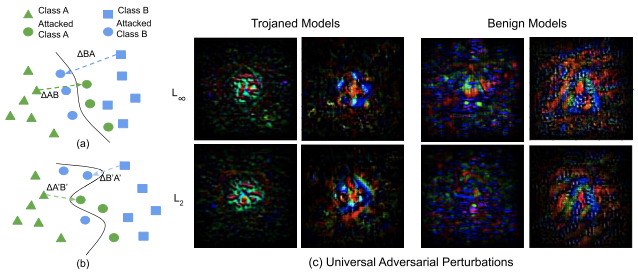


FIGURE 1. Decision boundary of: (a) Clean model and (b) Trojaned model. Adding a Trojan backdoor to a clean model shown in (a) results in a modified decision boundary shown in (b), in order to accommodate the poisoned training samples (shown by circles). This also makes it easier to change the class label of a sample (shown by arrows), since the distance across decision boundary is smaller in (b). (c) L_∞ and L_2 bounded universal perturbations for Trojaned and benign models from NIST-TrojAI dataset (from left to right: Trojaned ResNet50, Trojaned DenseNet121, benign ResNet50, and benign DenseNet121).

III. DETECTION OF TROJAN MODELS

During training, a neural network simultaneously learns feature representation and decision boundaries that partition the feature space into the respective classes. When a backdoor is inserted into a network, it alters the decision boundaries. Our hypothesis is that Trojan infected networks exhibit decision boundaries that are different from typical, benign models (see Fig. 1). Let $E_{\triangle BA}$ denote the magnitude of the smallest perturbation required to change the label of a sample from class B to class A across the decision boundary for a benign model, and vice versa for $E_{\triangle AB}$. Similarly, $E_{\triangle A'B'}$ and $E_{\triangle B'A'}$ denote the same for a Trojan infected model. For an infected model, the decision boundary is changed such that some backdoors are created close to other classes. Due to these changes in the decision boundary, we hypothesize that for most samples, $E_{\triangle A'B'} < E_{\triangle AB}$ and $E_{\triangle B'A'} < E_{\triangle BA}$ (see Fig. 1a,b), where $E_{\triangle AB}$ is proportional to the magnitude of $\triangle AB$ and so on.

Universal Adversarial Perturbation (UAP) [27] is image agnostic and when added to any image, sends it across the decision boundary to change its label. Hence, UAPs essentially encode the geometry of the decision boundaries [27], and are expected to be different in character for benign and Trojan models. Our approach exploits this fact to fingerprint the inserted triggers and combines it with *attack difficulty*, a measure of model robustness, to quantify the magnitude of the shift in boundary due to the inserted backdoor. Universal Adversarial perturbations are characterized by their *magnitude* and their *fooling rate* i.e. the proportion of perturbed images that are miss-classified. The magnitude of perturbation required to achieve a target fooling rate depends on the location of the decision boundary, and is reduced in the presence of artificially inserted backdoors. We define *attack difficulty* as the magnitude of an adversarial perturbation

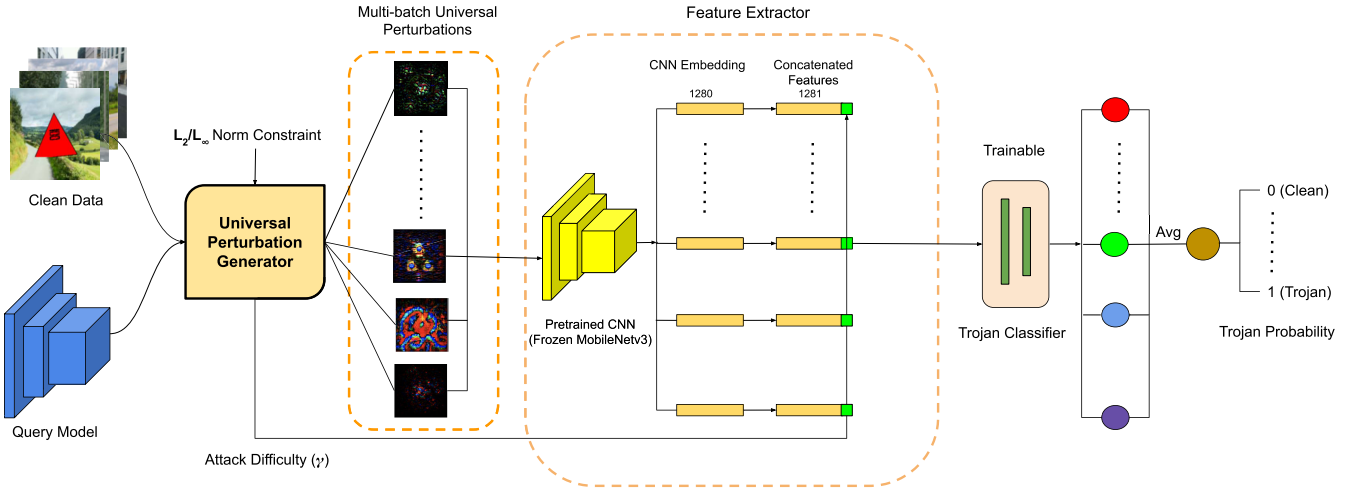


FIGURE 2. **Trojan detection: Employing clean data and the query model, Universal perturbation generator generates perturbations (shown enclosed by dashed bounding box), which are fed to the pre-trained CNN MobileNetv3 Feature extractor (shown in yellow) to extract features, these features are concatenated with attack difficulty and fed to Trojan classifier. Multi-batches are used to generate Trojan probabilities, as shown in the colored circles, and averaged across the batches to generate overall Trojan probability. In the data-free setting, the fast feature fool perturbation generator is used and the input is a Gaussian noise image.**

normalized by its fooling rate as a proxy measure of the average distance of the decision boundary from clean data samples. We train a classifier over the CNN features of the UAP fingerprints and the computed attack difficulty to classify a query network as *benign* or *Trojaned*.

A. BACKDOORED DECISION BOUNDARIES

The first component of our strategy to identify backdoored decision boundaries in NN classifiers is to define the notion of attack difficulty, a measure of model robustness which is distinctively affected by Trojan backdoors. Symbolically, let f denote a NN model and μ (in $\mathbb{R}^d$) the distribution of a dataset containing images x_i with corresponding ground-truth labels y_i. A standard definition of local model robustness in the standard data-dependent adversarial attack setting [28] is the minimum possible magnitude of the perturbation required to flip the model's prediction

$$Robustness(x_i; f) = \min_{\Delta x} ||\Delta x||_p, \quad \text{subject to } f(x_i + \Delta x) \neq y_i. \tag{1}$$

The data-independent robustness of the classifier f can then be defined by averaging over the data as

$$Robustness(f) = \mathbb{E}_{x_i \sim \mu} \frac{Robustness(x_i; f)}{||x_i||_p}. \tag{2}$$

We develop an analogous definition of robustness for the universal perturbation setting. For any desired threshold value δ in (0, 1), we can obtain an universal perturbation Δx such that the perturbation achieves a *minimum* fooling rate of $1 - \delta$ over the entire dataset, while its magnitude is bounded by a L_p norm constraint ϵ_p:

$$\mathop{\mathbb{P}}_{x_i \sim \mu} \{f(x_i + \Delta x) \neq y_i\} \geq 1 - \delta \ \text{ s.t. } \ ||\Delta x||_p \leq \epsilon_p, \tag{3}$$

where the minimum threshold $1 - \delta$ ensures that the perturbation is able to reach the decision boundary for a significant fraction of the dataset. In practice, the achieved fooling rate $\eta(\Delta x)$ can go much higher than the minimum.

Wang *et al.* [34] used the magnitude norm of targeted class specific perturbations to identify Trojan models. We find that using perturbation norm alone is sub-optimal since model robustness, in the universal perturbation setting (Equation 3), can only be characterized jointly by the perturbation norm combined with the fooling rate. We call this notion of model robustness *attack difficulty* (γ) defined as

$$\gamma(f) = \frac{||\Delta x||_p}{\eta(\Delta x)}, \tag{4}$$

which acts as a proxy measure of the average distance between the clean data points and the decision boundaries and hence serves as a discriminative feature for Trojan detection. We use L1-norm for computing $||\Delta x||_p$, however, some other L_p-norms can also be used.

The value of attack difficulty as a feature for detecting Trojaned models can be observed from the behaviour of a handful of models randomly selected from the Odysseus dataset in Figure 3. The fooling rate and L1 norm of the universal perturbation increases over the iterations of the perturbation generator, and do not strongly differentiate between Trojan and Clean models. Whereas attack difficulty clearly distinguishes between the two classes for models of different architectures within a couple of iterations, which shows the attack difficulty helps our Trojan detection not only in the generalizability but also efficiency.

A broader look at the distribution of fooling rates and perturbation magnitudes for the whole Odysseus-MNIST and NIST-Round1 datasets is provided in Figure 4. The extent of overlap between the clusters representing the Trojan and

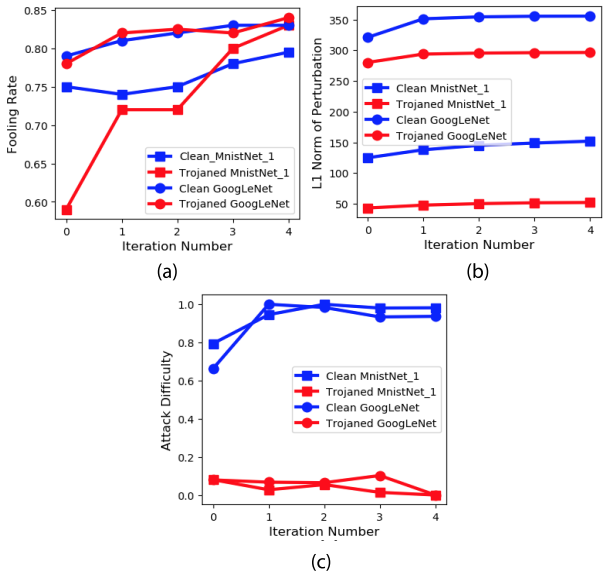


FIGURE 3. (a) Fooling rate and (b) L1 norm of the universal perturbation are not individually sufficient to distinguish Trojan models from clean ones, however our proposed (c) Attack difficulty is able to clearly separate Trojaned and clean models.

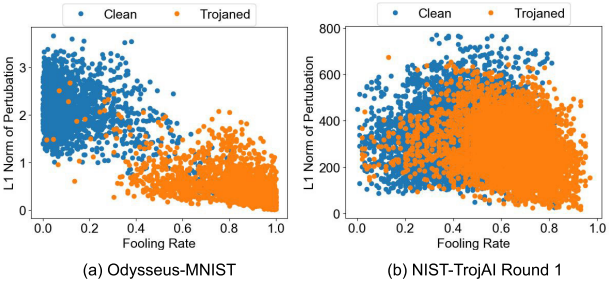


FIGURE 4. L1 norm of universal perturbation vs. fooling rate for (a) Odysseus-MNIST and (b) NIST-TrojAI round 1 datasets.

Clean models underscores the difficulty of Trojan detection on the NIST Datasets.

Our second hypothesis relies on the observation that over a certain fooling rate η, UAPs of clean and Trojan infected models are visually distinguishable, as shown in Figure 1c. The perturbations for Trojan models have characteristic localized spatial patterns characterizing the trigger used to poison the training data to insert a backdoor. We use an ImageNet pre-trained CNN model to extract features from these perturbations and combine them with the attack difficulty to train a classifier.

IV. TROJAN DETECTOR

Figure 2 shows the schematic overview of our proposed Trojan detector, Cassandra. The query model, along with a few clean labelled training samples, are used to generate UAPs bounded by either L_∞ or L_2 norms. Note that we *do not assume the presence of triggered images* in the training data, since triggers are unknown in a realistic scenario. In fact, none of our training samples contain a trigger. The attack difficulty for each perturbation is computed using its magnitude normalized by the fooling rate it achieved, whereas the spatial patterns are captured by a CNN pre-trained on ImageNet. These features are used to train the Trojan classifier with binary cross-entropy loss. To capture properties of the complex decision boundaries, multi-batch perturbations are used. The training data is divided into 10 batches to obtain 10 different universal perturbations and hence 10 Trojan probabilities. Classifier outputs for the multiple batches and perturbation types are averaged like an ensemble of models. To further improve the prediction, the final Trojan probability is computed from an ensemble of two networks which are trained with L_∞ and L_2 norm-bounded universal perturbations respectively.

A. PERTURBATION GENERATOR

In Cassandra, Universal Adversarial Perturbations (Eq. 3) [27] are computed with the DeepFool [28] kernel. UAPs have many advantages over using image specific perturbations. Firstly, since UAPs are image-agnostic, we don't need to compute class or data-specific perturbations, which allows easier scaling to bigger datasets. Secondly, UAPs can be generated by a data independent algorithm [29], which allows Cassandra to operate in Data-Free setting. Finally, UAPs achieve high fooling rate quickly with a small number of iterations [27], which makes our algorithm fast during training and inference.

B. FEATURE EXTRACTOR

A MobilenetV3-Large CNN [16] pre-trained on ImageNet is used to extract features to represent the spatial patterns found in the perturbations. The 1280-dimensional embedding output from the penultimate layer is used as a feature for the Trojan classifier. The network parameters are finetuned throughout training.

C. TROJAN CLASSIFIER

The output of the feature extractor module (1280-D) is concatenated with attack difficulty (1-D), and fed to the Trojan classifier, which is a fully connected network. The probability of the query model being Trojan infected is obtained by applying the sigmoid activation to the output.

D. DATA-FREE CASSANDRA

In practical settings, a few clean samples are always available. Moreover, UAPs can be generated in a data independent way [29]. Hence, Cassandra can also work in a data-free setting for Trojan detection. UAPs are computed independent of data with the Fast Feature Fool [29] algorithm which generates universal perturbations by iteratively updating an initial image consisting of sampled Gaussian noise with a feature space loss function. The loss function focuses on fooling the features learned at multiple layers of the network, hence disrupting the classification performance of the network. For a given classifier f with K layers, the loss function maximizes the product of mean activations of the model across layers

produced by the input perturbation:

$$l(f, \Delta x) = -\log(\prod_{i=1}^{K} \bar{f}_i(\Delta x)) \text{ such that } \|\Delta x\|_p < \epsilon, \quad (5)$$

where $\bar{f}_i$ is the mean activation of layer i of the model.

V. TARGETED CLASS PREDICTION

In a Trojaned model, the class(es) targeted for the poisoning attack are easier to launch an adversarial attack against, compared to the normal unaffected classes. For the purpose of target class prediction, we extend the notion of attack difficulty introduced in Section III-A to the case of targeted adversarial perturbations, by substituting the attack success rate for the fooling rate. For a given class c_j the attack success rate is the fraction of the dataset belonging to class c_j ($|c_j|$ is the cardinality of the class) for which label is successfully changed to some other class under targeted adversarial attack:

$$S(c_j) = \frac{\sum_{x_i \sim \mu | y_i = c_j} \mathbb{I}\{f(x_i + \Delta x(c_j)) \neq c_i\}}{|c_j|}, \quad (6)$$

and hence the class attack difficulty for c_j can be defined as

$$\gamma(f, c_j) = \frac{||\Delta x(c_j)||_p}{S(c_j)}. \quad (7)$$

We finalize our target class prediction in two stages. The first stage is our Trojan detection network which outputs the probability of the model being infected with a Trojan. Given a Trojaned model from the first stage, we use the Fast Gradient Sign Method (FGSM) [10] to compute targeted adversarial perturbations for each class label. Class Attack difficulty (Equation 7) is then calculated for each class-specific perturbation. We chose FGSM because of its fast execution time, since generating targeted perturbations for each potential targeted class label is compute intensive. We use outlier detection on the computed class attack difficulty to identify the class label targeted by the Trojan backdoor. We adopt the Median Absolute Deviation (MAD) [14], [34] method for outlier detection. An anomaly index is computed for each class as the absolute deviation of the attack difficulty from the median across classes, followed by normalization by the median to account for the dispersion of the data distribution. We select the label(s) with anomaly index value above a set threshold as the predicted targeted class(es).

VI. EXPERIMENTS

Implementation Details: For trojan detection, 500 clean image samples are used for each model to generate universal perturbations. In Eqn. 3, δ is set to 0.2 to quantify the desired fooling rate. The trojan detector is trained using the SGD optimizer with a learning rate of 0.001, the training converges within 200 epochs for all datasets. Learning rate and batch size were tuned using grid search with 5-fold cross validation. Four learning rates 0.001, 0.005, 0.0001, 0.00001 and 3 batch sizes 16, 32, 40 were tested. To predict the targeted label(s) using anomaly detection on attack difficulty we follow prior work [34] and assume the underlying distribution to be normal. Hence setting the constant estimator for the MAD outlier detector to 1.4826 means that under the normal distribution assumption, any data sample with anomaly index larger than 2 has $> 95\%$ probability of being an outlier. We adopt the anomaly index threshold of 2 and class labels with anomaly index larger than 2 are considered targeted class(es).

Since Universal Litmus Patterns [19] does not report results on NIST/Odysseus datasets, for comparison we train it using the reference code. The learning rate for litmus pattern generation and trojan classifier optimizers are set to 0.001 and 0.0001 and 500 training epochs are carried out. For comparison with Neural Cleanse [34], we utilize the reference code, whereas for Odyssey [7] we use results reported in the paper.

We repeat all experiments at least three times and report the average results along with standard deviation. To avoid constant repetition, henceforth results for NIST and Odysseus datasets are presented in the following fixed order in all sections: NIST-Round0, NIST-Round1, Odysseus-MNIST, Odysseus-FMNIST and Odysseus-CIFAR10.

A. DATASETS

We evaluate our proposed approach on three datasets, our own dataset of Trojaned & clean models for MNIST image classification (henceforth referred to as "Triggered MNIST"), the public NIST-TrojAI (Rounds 0 and 1) datasets and the newly released Odysseus datasets. Code to generate the Triggered MNIST was used from the TrojAI GitHub repo.,[1] the NIST datasets were obtained from the TrojAI challenge website. [2] and the Odysseus datasets were obtained from the project website. [3] Further details about the datasets can be found in the supplementary material.

B. RESULTS

1) TROJAN DETECTION

Table 1 shows results of our method on the Triggered MNIST, NIST-TrojAI and Odyssesus datasets and compares them to prior work: Neural Cleanse [34], Universal Litmus Patterns [19] and Odyssey [7]. NIST datasets are more challenging compared to Triggered MNIST, not only in terms of trigger types, color and size of the data used to train the infected models, but also due to the fact that the NIST models are much deeper. Our proposed method, Cassandra, outperforms all previous methods on five datasets including the two public NIST challenge datasets, and achieves comparable results on the sixth, Odysseus-CIFAR10 dataset. Our Data-Free Cassandra does not use any training samples and still outperforms previous methods on four datasets including the two public NITS challenge datasets. The performance of our Data-Free Cassandra is close to Cassandra on all datasets even though it operates in a very restrictive setting.

[1] https://github.com/trojai
[2] https://pages.NIST.gov/trojai/docs/data.html#download-links
[3] https://lcwn-lab.github.io/Odyssey/

TABLE 1. Trojan detection accuracy on Triggered MNIST, NIST-TrojAI and Odysseus datasets. Cassandra uses a limited number of clean data samples from the original dataset, whereas Data-Free Cassandra only needs access to the potentially Trojaned model. Best results for each dataset are in bold (99.9) and second-best are *italicized* (99.9).

Dataset	Method				
	Neural Cleanse [34]	Odyssey [7]	ULPs [19]	Cassandra (Ours)	Data-Free Cassandra (Ours)
Triggered MNIST	76.6 ± 1.2	-	92.7 ± 1.7	$\mathbf{94.4 \pm 1.2}$	*93.5 ± 2.0*
NIST-Round0 [30]	62.5 ± 2.4	85.0 ± 3.8	60.6 ± 2.2	$\mathbf{92.5 \pm 1.1}$	*90.6 ± 1.9*
NIST-Round1 [31]	68.0 ± 1.9	83.4 ± 0.8	70.0 ± 1.4	$\mathbf{89.7 \pm 1.3}$	*89.0 ± 1.4*
Odysseus-MNIST [7]	79.8 ± 2.5	86.4 ± 1.1	*96.6 ± 0.4*	$\mathbf{98.2 \pm 0.7}$	96.3 ± 1.5
Odysseus-FashionMNIST [7]	60.7 ± 1.8	*85.3 ± 2.2*	71.3 ± 0.8	$\mathbf{86.2 \pm 2.1}$	82.5 ± 1.7
Odysseus-CIFAR10 [7]	77.0 ± 1.9	$\mathbf{98.7 \pm 0.6}$	85.7 ± 0.7	*96.8 ± 0.5*	96.1 ± 0.8

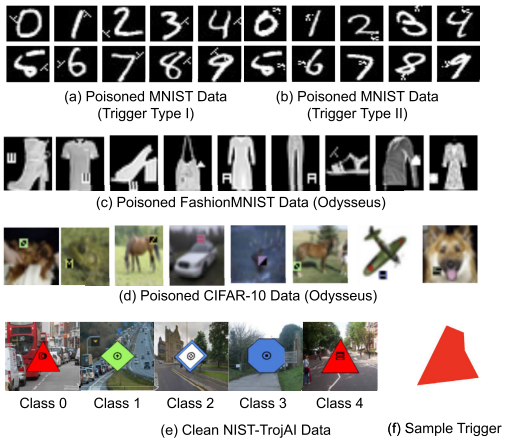


FIGURE 5. Data samples from TriggeredMNIST, Odysseus and NIST-TrojAI. Triggered MNIST dataset samples containing Type I triggers (a) and Type II triggers (b). Triggered samples from the Odysseus-FashionMNIST (c) and Odysseus-CIFAR10 (d). Clean image samples of 5 classes (e) and one illustrative trigger sample (f) from the NIST datasets. Triggers in the NIST dataset are of random sizes, shape and colors. Apart from (f), triggers and triggered images are not provided in the TrojAI dataset.

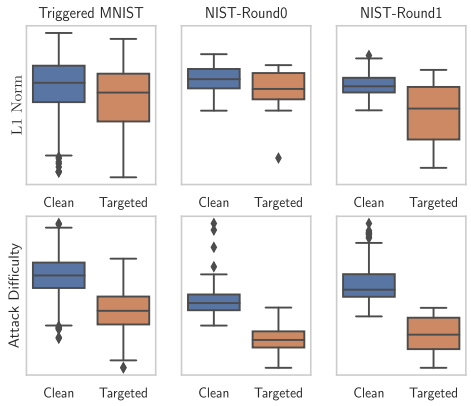


FIGURE 6. Effectiveness of attack difficulty (second row) over L_1 norm (first row) at classifying unaffected and targeted classes in Trojaned models. Attack difficulty and L_1 norm are computed for targeted class-specific FGSM perturbation. Box plots show min/max, mean and quartile values.

2) TARGETED CLASS PREDICTION

Table 2 shows results for our targeted class prediction method. The proposed two stage prediction algorithm based on the attack difficulty and predicted P(Trojan) improves the classification accuracy significantly over the baseline (without P(Trojan)) from 76.1%, 72.5%, 70.0% to 90.0%, 94.7%, 88.1% on Triggered MNIST, NIST-Round0 and NIST-Round1 datasets respectively. Using ground truth P(Trojan) further improves classification accuracy which demonstrates attack difficulty is a critical indicator of targeted class label(s), independent of Trojan detection.

Figure 6 shows that the proposed attack difficulty is able to correctly identify targeted classes of Trojaned models from the unaffected ones for the three datasets. Note that the simple L_1 norm used for Trojaned model detection in Neural Cleanse [34] fails to detect the targeted class.

TABLE 2. Targeted class prediction accuracy significantly increases when the result of Trojan detection (P(Trojan)) is already known. All Trojan models tested here are any-to-one i.e. one targeted class per model.

P(Trojan)	Triggered MNIST	NIST Round0	NIST Round1
None	76.1 ± 1.5	72.5 ± 2.5	70.0 ± 1.0
Predicted	90.0 ± 1.0	94.7 ± 1.7	88.1 ± 0.7
Ground truth	95.0 ± 1.3	98.8 ± 1.4	91.7 ± 1.5

TABLE 3. Generalization across trigger Types: Cassandra achieves good trojan detection results on the triggered MNIST dataset even when trained and tested on different type of trigger.

# Train/Test Models	Train/Test Trigger	Accuracy(%)
240 / 60	I / I	93.3 ± 1.6
240 / 60	II / II	91.7 ± 1.7
240 / 60	I / II	90.0 ± 1.2
240 / 60	II / I	91.7 ± 1.1
480 / 120	I, II / I, II	94.4 ± 1.2

C. GENERALIZATION

1) TRIGGER TYPE

We investigate the generalization capability of Cassandra Trojan detector across trigger types on the TriggeredMNIST dataset. Table 3 shows that the classification accuracy is consistently high when both test and train models are infected with the same type of triggers (93.3% for TypeI and 91.7% for Type II). However, even when training and test models are infected by different types of triggers, the algorithm still retains a comparable high classification accuracy of 91.7%

TABLE 4. Generalization across architectures: Trojan detection accuracy when test models have different architectures compared to the training models. D, I, R, G and V stand for DenseNet, Inceptionv3, ResNet, GoogleNet and VGG19 respectively. MNISTNet I, II, III and IV are shallow CNN architectures designed for MNIST classification.

Architecture(s)		NIST-TrojAI Datasets	
Train	Test	Round0	Round1
D, I	R	78.7 ± 3.1	86.8 ± 0.4
D, R	I	81.3 ± 1.0	81.5 ± 0.4
I, R	D	78.6 ± 1.1	81.5 ± 0.3
Architecture(s)		**Odysseus Datasets**	
Train	**Test**	**FMNIST**	**CIFAR10**
G, R, V	D	75.9 ± 1.3	84.9 ± 1.2
D, R, V	G	68.0 ± 0.8	76.7 ± 1.1
D, G, V	R	77.4 ± 1.1	88.5 ± 0.7
D, G, R	V	64.6 ± 0.9	87.0 ± 1.1
Train	**Test**	**Odysseus-MNIST**	
II, III, IV	I	82.4 ± 1.6	
I, III, IV	II	94.6 ± 0.6	
I, II, IV	III	95.9 ± 0.8	
I, II, III	IV	83.9 ± 1.2	

(Type II to Type I transfer) or 90% (Type I to Type II transfer), which demonstrates that our method is able to generalize across trigger types. We achieve the best result of 94.4% performance when both trigger types are present, in equal proportions, in the training and test sets, indicating that diversity in trigger types during training helps the model generalize even better.

2) MODEL ARCHITECTURES

We test the generalization ability of Cassandra across different model architectures and report the results in Table 4. We observe that even when training and test model architectures are different, Cassandra still maintains relatively high classification accuracy. On average, we observe 13%, 6.4%, 9.0%, 14.7% and 12.5% (median = 12.5%) drop in accuracy relative to the mixed architectures setting when using the NIST and Odysseus datasets respectively in the cross-architecture setting.

D. COMPUTATIONAL COST

Since Cassandra uses class-agnostic universal perturbations for Trojan detection, its computation cost is independent of the number of classes of the query model. Hence, Cassandra's complexity is $O(N)$, where N is the batch size for computing universal perturbations. This is superior to methods like Neural Cleanse, which must perform $O(NK^2)$ optimizations for a query model with K classes, since it launches class specific targeted attacks. Cassandra's inference time per query model on a single 2080 Ti GPU is much faster compared to Neural-Cleanse: $\sim$ 15 to 25 versus 180 to 240 seconds on the Odysseus dataset, and the inference times per model on the NIST dataset are 300, 216 and 1432 seconds for Cassandra, Data-Free Cassandra and Neural Cleanse respectively. Note that the times are larger than Odysseus models, since NIST models are more complex.

TABLE 5. Effect of using multi-batch perturbations and attack difficulty. Trojan detection accuracy on the NIST and Odysseus datasets improves significantly with multiple perturbations ($n = 10$) from different batches of training data. Employing both L_∞ and L_2 norm-bounded perturbations further improves the accuracy. Adding attack difficulty (γ) leads to improvements across datasets.

	Classification Accuracy			
	NIST-Round0		NIST-Round1	
UAP Type	**w/o γ**	**w/ γ**	**w/o γ**	**w/ γ**
L_∞ (single)	77.3 ± 2.1	82.2 ± 1.4	79.5 ± 1.5	83.2 ± 1.7
L_∞	85.3 ± 1.8	90.0 ± 1.8	87.3 ± 1.9	89.0 ± 1.4
L_2	83.7 ± 1.1	90.6 ± 1.5	83.9 ± 1.4	87.8 ± 2.3
L_∞ & L_2	85.9 ± 1.4	92.5 ± 1.1	86.5 ± 1.6	89.7 ± 1.7
	Odysseus-MNIST		**Odysseus-CIFAR10**	
UAP Type	**w/o γ**	**w/ γ**	**w/o γ**	**w/ γ**
L_∞ (single)	81.6 ± 2.0	93.9 ± 2.1	88.4 ± 1.9	90.3 ± 1.7
L_∞	93.5 ± 1.2	97.4 ± 0.2	92.6 ± 0.8	94.3 ± 1.3
L_2	94.1 ± 0.9	97.2 ± 0.6	93.2 ± 0.9	96.5 ± 0.3
L_∞ & L_2	96.6 ± 0.7	98.2 ± 0.7	93.3 ± 0.5	96.8 ± 0.5

TABLE 6. Perturbation generator hyper-parameters do not significantly affect Trojan detection accuracy. However, using higher number of iterations to refine perturbations and imposing a higher ceiling on perturbation magnitude have a small positive impact.

	L_∞ **Perturbation Magnitude** ($\epsilon_\infty/255$)				
	$\epsilon_2/255 = 10$, # L_2 iterations = 10				
# Iterations	**0.1**	**0.2**	**0.4**	**0.8**	**1**
5	87.5 ± 2.7	87.6 ± 1.4	90.0 ± 1.6	90.2 ± 1.4	90.0 ± 2.1
10	87.9 ± 1.7	90.7 ± 1.1	92.5 ± 1.4	92.5 ± 1.2	92.5 ± 1.1
15	90.4 ± 2.1	90.6 ± 1.3	90.6 ± 1.3	92.5 ± 2.1	92.5 ± 1.2
	L_2 **Perturbation Magnitude** ($\epsilon_2/255$)				
	$\epsilon_\infty/255 = 1$, # L_∞ iterations = 10				
# Iterations	**5**	**10**	**20**	**30**	**40**
5	87.5 ± 1.8	90.4 ± 1.4	90.2 ± 1.8	90.0 ± 1.2	90.0 ± 1.4
10	90.3 ± 1.4	92.5 ± 1.1	90.7 ± 1.4	92.0 ± 1.8	92.5 ± 1.1
15	90.7 ± 1.2	92.5 ± 1.1	92.5 ± 1.2	90.2 ± 1.3	92.5 ± 1.2

E. ABLATION STUDY

1) MULTI-BATCH PERTURBATIONS AND ATTACK DIFFICULTY

We performed a thorough ablation study and present the results in Table 5. Using only a single L_∞ UAP per model computed from the complete training data, we achieve 77.3%, 79.5%, 81.6% and 88.4% classification accuracy for NIST and Odysseus datasets respectively. After using multi-batch perturbations that divide the training data into 10 batches to compute multiple perturbations, the accuracy improves by 8.0%, 7.8%, 11.9% and 4.2% (median = 8%) over the single perturbation baseline. Adding attack difficulty as a feature for the classifier further improves the accuracy in all cases. Cassandra achieves the best results in the multi-batch, two stream ensemble setting, reaching trojan detection accuracy exceeding or closely matching the state of the art in all datasets.

2) PERTURBATION GENERATOR HYPERPARAMETERS

Our experiments show (in Table 6) that Cassandra's trojan detection performance is robust to the choice of perturbation generator parameters. On comparing mean classification

accuracy achieved using different number of iterations and L_2 and L_∞ norm bounds for universal adversarial perturbations, we find only a slight variation in Trojan detection accuracy.

VII. CONCLUSION

We proposed Cassandra, a method for detecting Trojan infected models using only few (or no) clean data samples, which sets the new state-of-art on multiple large scale datasets. Cassandra relies on detecting characteristic fingerprints left on model decision boundaries by Trojan backdoors, through CNN features of Universal Adversarial Perturbations and our proposed metric of model robustness: attack difficulty. Secondly, we demonstrate that Cassandra can generalize to unknown architectures and trigger types. In addition, for the first time in the literature, we propose a method which can identify the class(es) targeted by the Trojan attack. This provides further information on the type of malicious behaviour embedded in a Trojan infected model, e.g. which identity is being impersonated in a Trojaned face recognition model. We also demonstrate that Cassandra is able to generalize across model architectures (in Table 4) trained on the same dataset. Future work can focus on further removing the constraints of Cassandra and developing domain-agnostic Trojan detectors which can generalize across datasets as well.

APPENDIX A
TARGETED CLASS DETECTION ALGORITHM

The procedure for targeted class prediction is given in Algorithm 1.

Algorithm 1: Two-Stage Method to Detect a Trojan Infected Model and Predict Its Target Class Using Only Clean Image Samples

Data: Query model
Result: P(Trojan) and TargetedClass
Stage One: Use Trojan Detection network to get P(Trojan);
if $P(Trojan) \geq 0.5$ **then**
  for $C_i \leftarrow 0$ **to** C **do**
    use FGSM to generate targeted adversarial perturbations with C_i as the target class;
    compute attack difficulty ($\gamma_i = \frac{L_1 Norm}{FoolingRate}$) for perturbation;
  end for
  TargetedClass = perform outlier detection over the attack difficulties γ_is;
  output P(Trojan) and target class prediction;
else
  output P(Trojan) and target class(None);
end if

APPENDIX B
TRIGGEREDMNIST DATASET GENERATION

A. CLEAN MODEL GENERATION

The data is split into training set: 60,000 images (6,000 per class), and test set: 10,000 images (1,000 per class). The clean data are used for training 300 benign/clean models with three architecture types (ModdedBadnet, Badnet and ModdedLenet5net), each with 100 models (see Table 7).

B. TROJANED MODEL GENERATION

1) CLEAN DATA

The MNIST dataset has 10 classes with 70,000 clean images (without triggers).

2) TRIGGERED DATA

Two types of triggers, Type I and Type II (see Figure in the main paper) were inserted into images of MNIST dataset. The Triggered MNIST data was combined with clean data to generate Trojaned models. The following three data splits were used in our experiments:

Data split 1: training: 60,000 (triggered data: 10%), testing:10,000 (triggered data: 10%).

Data split 2: training: 60,000 (triggered data: 15%), testing:10,000 (triggered data: 15%).

Data split 3: training: 60,000 (triggered data: 20%), testing:10,000 (triggered data: 20%).

3) MODELS

In addition to the 300 benign models, another 600 Trojaned models of the same three architectures (ModdedBadnet, Badnet and ModdedLenet5net) were generated. Trojaned models were trained by the Triggered MNIST data and clean data where the proportion of triggered data varied as 10%, 15% and 20%. Table 7 shows the details of both clean and infected models trained for any-to-any Trojan attack. Any-to-one attack models were generated similar to any-to-any models. 300 Trojaned models were trained by any-to-any targeted attack, and another 300 were trained for any-to-one targeted attack.

Evaluations of ModdedBadnet, Badnet and ModdedLeNet5 models are shown in Table 8 for any-to-any attack and in Table 9 for any-to-one targeted attack. The clean models and Trojaned models both have high classification accuracy when the test data is clean. The clean models also have high classification accuracy when the test data is triggered. Since there is no Trojan in the clean model, the triggered image samples are correctly classified. However, for the Trojaned models, the classification accuracy (100 − Attack Success Rate) for triggered data is low since the triggered images are misclassified. The tables show Attack Success Rates only for the triggered data which is very high. These results imply that the Trojan (backdoor) was successfully inserted into the models.

APPENDIX C
NIST ROUND0 AND NIST ROUND1 DATASETS

The NIST datasets consist of CNN classification models for traffic sign signals. Half of the models are benign models and half are Trojaned models. The models have three architectures namely, Inception-v3, DenseNet-121, and ResNet50.

TABLE 7. Types of models included in the Triggered MNIST dataset. Half the models are for any-to-any attack and half are for any-to-one attack. For the latter case each model only has one targeted class.

Model name	Model Architecture	Trigger	Triggered data	#
ModdedBadNet	2 Conv + 1 Dense	Type I, II	10%, 15% and 20%	100+100
BadNet	2 Conv + 2 Dense	Type I, II	10%, 15% and 20%	100+100
ModdedLeNet5	3 Conv + 2 Dense	Type I, II	10%, 15% and 20%	100+100

TABLE 8. Attack success rate and classification accuracy for three types of trojaned models (any-to-any attack) for triggered MNIST dataset. Success rate is the proportion of images for which predictions by the Trojaned model is changed to an incorrect label.

	Trojaned Model				Clean Model
	Attack Success Rate		Classification Accuracy		
Model Type	Trigger I	Trigger II	Trigger I	Trigger II	
BadNet	98.7	98.7	98.9	99.0	99.1
ModdedBadNet	97.3	97.6	97.2	96.5	98.8
ModdedLeNet5net	97.8	98.6	98.0	97.3	98.7

TABLE 9. Attack success rate and classification accuracy for three types of trojaned models (any-to-one targeted attack) on the Triggered MNIST dataset. Success rate is the proportion of images that changed label to the target class for Trojaned model.

	Trojaned Model			
	Attack Success Rate		Classification Accuracy	
Model Type	Trigger I	Trigger II	Trigger I	Trigger II
Badnet	99.1	99.0	98.8	98.9
ModdedBadnet	98.5	98.3	97.6	97.4
ModdedLenet5net	98.8	98.4	98.0	97.5

TABLE 10. Attack success rate (for Trojan trigger infused data) and top-1 classification accuracy (for clean data) for NIST-Round0 dataset. Success rate is the proportion of images for which the prediction changes to the target label in Trojaned models.

	Trojan Infected Model		Clean Model	
Model Type	Attack Success Rate	Classification Accuracy		# models
DenseNet-121	99.82	99.76	99.90	63
Inception-v3	99.87	99.69	99.75	69
ResNet50	99.80	99.68	99.76	68
# models	100		100	**200**

TABLE 11. Attack success rate (for Trojan trigger infused data) and top-1 classification accuracy (for clean data) for three types of Trojaned and clean models from the NIST-Round1 dataset. Success rate is the proportion of images for which the prediction changes to the targeted label in Trojaned models.

	Trojan Infected Model		Clean Model	
Model Type	Attack Success Rate	Classification Accuracy		# models
DenseNet-121	99.88	99.81	99.88	313
Inception-v3	99.84	99.85	99.89	250
ResNet50	99.58	99.81	99.83	437
# models	500		500	**1000**

The models were trained on synthetically created image data of artificial traffic signs superimposed on road background scenes. The Trojaned models have been poisoned with triggers of different color, size and shape. **Round0** dataset consists of 200 models, while **Round1** dataset has 1,000 models. NIST also holds a sequestered test dataset to evaluate models.

Table 10 and Table 11 show the model details and the performance of the three architecture types present in the NIST Round0 and Round1 datasets. Notice that the Trojan infected models have accuracy at par with the clean models and yet they have a very high attack success rate on the triggered data. Fig.7 shows the trigger fraction distribution for NIST datasets. **NIST-Round0** and **NIST-Round1** datasets are both from the same distribution, the main difference is that Round0 consists of 200 models, while Round1 dataset has 1,000 models.

APPENDIX D
ODYSSEUS DATASET

Odysseus [7] is comprised of three sub-datasets, Odysseus-MNIST, Odysseus-FashionMNIST and Odysseus-CIFAR10. It contains over 3000 models, with 4 different architectures being used in each sub-dataset. The triggers used to train the Trojaned model are of different sizes (1% - 3% of the clean image) and color, and were added to several different locations of the clean image. The Trojaned models of Odysseus are trained with many-to-one and many-to-many target attacks with the proportion of poisoned data being around 15% to 20%.

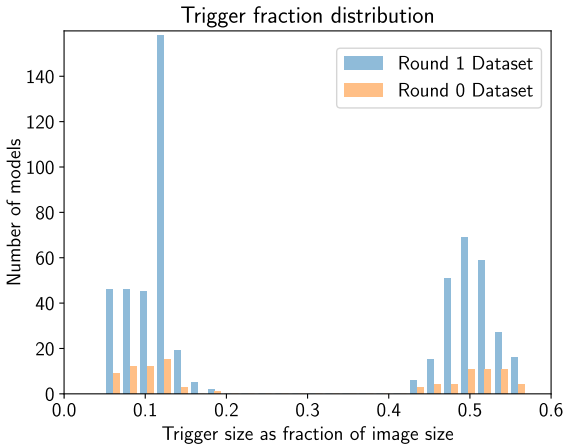


FIGURE 7. **The trigger size distribution of NIST datasets.**

REFERENCES

[1] N. Akhtar, J. Liu, and A. Mian, "Defense against universal adversarial perturbations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3389–3398.

[2] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14410–14430, 2018.

[3] 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," 2018, *arXiv:1811.03728*. [Online]. Available: http://arxiv.org/abs/1811.03728

[4] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "DeepInspect: A black-box Trojan detection and mitigation framework for deep neural networks," in *Proc. 28th Int. Joint Conf. Artif. Intell.* AAAI Press, Aug. 2019, pp. 4658–4664.

[5] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017, *arXiv:1712.05526*. [Online]. Available: http://arxiv.org/abs/1712.05526

[6] E. Chou, F. Tramèr, and G. Pellegrino, "SentiNet: Detecting localized universal attacks against deep learning systems," 2018, *arXiv:1812.00292*. [Online]. Available: http://arxiv.org/abs/1812.00292

[7] M. Edraki, N. Karim, N. Rahnavard, A. Mian, and M. Shah, "Odyssey: Creation, analysis and detection of Trojan models," 2020, *arXiv:2007.08142*. [Online]. Available: https://arxiv.org/abs/2007.08142

[8] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1625–1634.

[9] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," 2017, *arXiv:1703.00410*. [Online]. Available: http://arxiv.org/abs/1703.00410

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, Y. Bengio and Y. LeCun, Eds., San Diego, CA, USA, May 2015.

[11] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (statistical) detection of adversarial examples," 2017, *arXiv:1702.06280*. [Online]. Available: http://arxiv.org/abs/1702.06280

[12] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "BadNets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47230–47244, 2019.

[13] 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, *arXiv:1908.01763*. [Online]. Available: http://arxiv.org/abs/1908.01763

[14] F. R. Hampel, "The influence curve and its role in robust estimation," *J. Amer. Statist. Assoc.*, vol. 69, no. 346, pp. 383–393, 1974.

[15] D. Hendrycks and K. Gimpel, "Early methods for detecting adversarial images," 2016, *arXiv:1608.00530*. [Online]. Available: http://arxiv.org/abs/1608.00530

[16] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1314–1324.

[17] X. Huang, M. Alzantot, and M. Srivastava, "NeuronInspect: Detecting backdoors in neural networks via output explanations," 2019, *arXiv:1911.07399*. [Online]. Available: http://arxiv.org/abs/1911.07399

[18] I. Oseledets and V. Khrulkov, "Art of singular vectors and universal adversarial perturbations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8562–8570.

[19] S. Kolouri, A. Saha, H. Pirsiavash, and H. Hoffmann, "Universal litmus patterns: Revealing backdoor attacks in CNNs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 301–310.

[20] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1778–1787.

[21] 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*. Cham, Switzerland: Springer, 2018, pp. 273–294.

[22] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, San Diego, CA, USA: The Internet Society, 2018.

[23] Y. Liu, A. Mondal, A. Chakraborty, M. Zuzak, N. Jacobsen, D. Xing, and A. Srivastava, "A survey on neural Trojans," in *Proc. 21st Int. Symp. Qual. Electron. Design (ISQED)*, Mar. 2020.

[24] J. Lu, T. Issaranon, and D. Forsyth, "SafetyNet: Detecting and rejecting adversarial examples robustly," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 446–454.

[25] D. Meng and H. Chen, "MagNet: A two-pronged defense against adversarial examples," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 135–147.

[26] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *Proc. Int. Conf. Learn. Represent.*, 2017.

[27] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1765–1773.

[28] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2574–2582.

[29] K. R. Mopuri, U. Garg, and R. V. Babu, "Fast feature fool: A data independent approach to universal adversarial perturbations," 2017, *arXiv:1707.05572*. [Online]. Available: http://arxiv.org/abs/1707.05572

[30] NIST. (2020). *NIST TrojAI Round 0 Dataset*. [Online]. Available: https://pages.nist.gov/trojai/docs/data.html#round-0-dry-run

[31] NIST. (2020). *NIST TrojAI Round 1 Dataset*. [Online]. Available: https://pages.nist.gov/trojai/docs/data.html#round-1

[32] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2014.

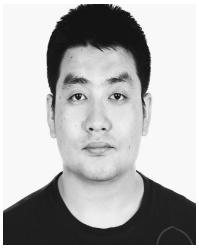
[33] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *Int. Conf. Learn. Represent.*, 2018.

[34] 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 *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 707–723.

[35] R. Wang, G. Zhang, S. Liu, P.-Y. Chen, J. Xiong, and M. Wang, "Practical detection of Trojan neural networks: Data-limited and data-free cases," in *Proc. 16th Eur. Conf. Comput. Vis. (ECCV)*. Glasgow, U.K.: Springer, Aug. 2020, pp. 222–238.

[36] C. Xie, Y. Wu, L. V. D. Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 501–509.

[37] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019.



XIAOYU ZHANG received the B.S. degree in geophysics from Jilin University, in 2008, and the M.S. degree in computer science from The University of Texas at Arlington, in 2019. He joined the Center for Research in Computer Vision Group, University of Central Florida, in 2019. His current research interests include the areas of trojaned neural network detection and adversarial attacks.

ROHIT GUPTA received the B.Tech. degree in electrical engineering and the M.Tech. degree in computer science from the Indian Institute of Technology at Kanpur, Kanpur, India, in 2015 and 2018, respectively. He is currently pursuing the Ph.D. degree in computer science with the University of Central Florida. His current research interests include the areas of adversarial robustness and representation learning.

AJMAL MIAN (Senior Member, IEEE) is currently a Professor of computer science with The University of Western Australia. His research interests include computer vision, deep learning, shape analysis, face recognition, human action recognition, and video analysis. He was a recipient of two prestigious national fellowships from the Australian Research Council and several awards, including the West Australian Early Career Scientist of the Year 2012, the Excellence in Research Supervision, the EH Thompson Award, the ASPIRE Professional Development Award, the Vice Chancellors Mid-Career Award, the Outstanding Young Investigator Award, the Australasian Distinguished Dissertation Award, and various best paper awards. He served as the General Chair for DICTA 2019 and ACCV 2018. He is an Associate Editor of IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON IMAGE PROCESSING, and the *Pattern Recognition* journal.

NAZANIN RAHNAVARD (Senior Member, IEEE) received the Ph.D. degree from the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, in 2007. She is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL, USA. She has interest and expertise in a variety of research topics in the communications, networking, signal processing, and machine learning areas. She was a recipient of the NSF CAREER Award, in 2011, and the 2020 UCF's College of Engineering and Computer Science Excellence in Research Award. She serves on the Editorial Board for the journal on *Computer Networks* (COMNET) (Elsevier) and the Technical Program Committee for several prestigious international conferences.

MUBARAK SHAH (Life Fellow, IEEE) is currently a UCF Trustee Chair Professor and the Founding Director of the Center for Research in Computer Vision, University of Central Florida (UCF). His research interests include video surveillance, visual tracking, human activity recognition, visual analysis of crowded scenes, video registration, and UAV video analysis. He is a fellow of the NAI, AAAS, IAPR, and SPIE. He was a recipient of the ACM SIGMM Technical Achievement Award, the IEEE Outstanding Engineering Educator Award, Harris Corporation Engineering Achievement Award, and an honorable mention for the ICCV 2005 Where Am I? Challenge Problem, the 2013 NGA Best Research Poster Presentation, 2nd place in Grand Challenge at the ACM Multimedia 2013 Conference, and Runner Up for the Best Paper Award in ACM Multimedia Conference in 2005 and 2010. At UCF, he has received the Pegasus Professor Award, the University Distinguished Research Award, the Faculty Excellence in Mentoring Doctoral Students, the Scholarship of Teaching and Learning Award, the Teaching Incentive Program Award, and the Research Incentive Award. He was the Program Co-Chair of CVPR 2008. He was an Associate Editor of the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (TPAMI) and a Guest Editor of the Special Issue of the *International Journal of Computer Vision* on Video Computing. He is an Editor of an international book series on video computing, and was the Editor-in-Chief of *Machine Vision and Applications* journal, and an Associate Editor of *ACM Computing Surveys* journal. He has served as an ACM distinguished speaker and IEEE distinguished visitor speaker.