IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 23, 2021, accepted January 3, 2022, date of publication January 7, 2022, date of current version January 14, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3141077

A Feature-Based On-Line Detector to Remove Adversarial-Backdoors by Iterative Demarcation

HAO FU, AKSHAJ KUMAR VELDANDA, PRASHANTH KRISHNAMURTHY, (Member, IEEE), SIDDHARTH GARG, AND FARSHAD KHORRAMI, (Senior Member, IEEE)

Department of Electrical and Computer Engineering, New York University, Brooklyn, NY 11201 USA

Corresponding author: Hao Fu (hf881@nyu.edu)

This work was supported in part by the Army Research Office under Grant W911NF-21-1-0155, and in part by the New York University (NYU) Abu Dhabi Center in Artificial Intelligence (AI) and Robotics (CAIR).

ABSTRACT This paper proposes a novel feature-based on-line detection strategy, Removing Adversarial-Backdoors by Iterative Demarcation (RAID), for backdoor attacks. The proposed method is comprised of two parts: off-line training and on-line retraining. In the off-line training, a novelty detector and a shallow neural network are trained with clean validation data. During the on-line implementation, both models attempt to detect samples from the streaming data that differ from the validation data (i.e., flag likely-poisoned samples and possibly a few clean samples as false positives). An anomaly detector is used to purify the anomalous data by removing the clean samples. A binary support vector machine (SVM) is trained with the purified anomalous data and the clean validation data. RAID uses the SVM to detect poisoned inputs. To increase the accuracy as new anomalous data is being detected, the SVM is updated as well in real-time. It is shown that with updating, RAID can efficiently reduce the attack success rate while maintaining the classification accuracy against various types of backdoor attacks. The efficacy of RAID is compared against several state-of-the-art techniques. Additionally, it is shown that RAID only requires a small clean validation dataset to achieve such performance, and therefore provides a practical and efficient approach.

INDEX TERMS Machine learning, neural networks, pattern analysis.

I. INTRODUCTION

Deep neural networks (DNN) are widely used in various applications including object detection (Ren *et al.* [1]), face recognition (Sun *et al.* [2], Taigman *et al.* [3]), natural language processing (Collobert *et al.* [4], Bahdanau *et al.* [5]), self-driving (Chen *et al.* [6]), navigation (Fu *et al.* [7]), surveillance (Osia *et al.* [8]), and cyber-physical systems security (Patel *et al.* [9]–[11]). However, there are many security risk issues in the usage of neural networks, such as adversarial attacks (Athalye *et al.* [12], Carlini and Wagner [13], [14], Eykholt *et al.* [15], Goodfellow *et al.* [16], He *et al.* [17], Moosavi-Dezfooli *et al.* [18], Szegedy *et al.* [19]). Recently, training time attacks (Chen *et al.* [20], Gu *et al.* [21]) are drawing increasing attention. This is because that training DNNs is challenging, especially for individuals or small entities, due to difficulties in obtaining large high-quality labeled datasets and the cost of maintaining or renting computational resources needed to train a complex model (Esteva *et al.* [22], Roh *et al.* [23], Halevy *et al.* [24]). Hence, users often outsource DNN implementation and training tasks to third-party clouds or download pre-trained models from on-line model repositories. This, however, exposes the user to training time attacks, especially the ''backdoor'' attack.

The ''backdoor'' attack refers to an *attacker* training a malicious model that mis-predicts when the input contains attacker-chosen backdoor triggers (''poisoned'' data) embedded into the training data. Specifically, by choosing proper training hyper-parameters, trigger patterns, etc., the attacker can make the backdoored model output specific labels on the poisoned data while preserving high accuracy on the clean data (i.e., data without the trigger). Such backdoored DNNs may cause security risks, financial harm, and safety implications for the end-user (e.g., misclassifying traffic signs in autonomous vehicle applications as shown in Fig. 1). Detecting backdoor attacks is therefore of critical importance.

The existing detect/defend methods against the backdoor attack can be divided into several classes regarding the

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

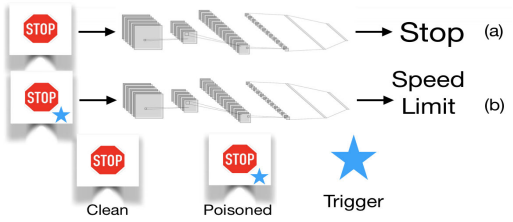


FIGURE 1. The backdoored DNN outputs correct (a) or wrong (b) labels for clean or poisoned inputs, respectively.

assumptions and used tools. Tran *et al.* [25], Chen *et al.* [26] assume the user can afford the training task but the training dataset is contaminated (i.e., it contains poisoned data). They use clustering-based methods (i.e., singular-value decomposition (SVD) and k-means) to purify the dataset. Although it may be true in some scenarios that the entire clean/contaminated training dataset is available to the user, in our attack model, we assume that the user does not have access to the clean/contaminated training dataset.

Reverse-engineering based methods, such as Neural Cleanse (Wang *et al.* [27]), artificial brain stimulation (ABS Liu *et al.* [28]), Guo's method (Guo *et al.* [29]), and Qiao's method (Qiao *et al.* [30]), are another popular approach to defend against the backdoor attack. These methods typically define a loss function and solve an optimization problem. Through back-propagation, some reverse-engineered triggers are found. Then, they create some synthetic data by applying these reverse-engineered triggers to the clean validation data, and use the synthetic data to fine-tune the network. One disadvantage is that these methods have restrictions on triggers and the network activation. For instance, Neural Cleanse requires assumptions on the trigger size. However, the attacker has complete control of the triggers. Therefore, the reverse-engineering based methods may become ineffective when the attacker uses some triggers that do not satisfy the assumptions. In this work, we do not consider having assumptions on the triggers.

Another type of approaches is to consider the poisoned data as novelty samples that are different from the clean validation data. Then, statistical tools can be used to detect poisoned inputs, such as strong intentional perturbation (STRIP Gao *et al.* [31]), Mahalanobis distance-based novelty detection (MD Lee *et al.* [32]), and Kwon's method (Kwon [33]). However, training the novelty detection model requires a relatively large amount of validation dataset. The performance of this type of works decreases considerably when the clean validation dataset is small. Our work intersects with this group of studies but overcomes the disadvantage by fully utilizing the on-line streaming data.

Our motivation comes from the fact that once the backdoored network is implemented for on-line usage, there will be substantial on-line data available. One can therefore take advantage of the on-line streaming data without requiring a reasonable amount of clean validation data or restrictive assumptions on the trigger. Additionally, detection methods based on the usage of on-line data will have a broader range of applications than the discussed methods since restrictive assumptions can be removed. The on-line data usage based detection methods may perform ineffectively initially because the on-line data is scarce at the beginning. However, with the sacrifice of initial performance, it can eventually provide very good performance in backdoor detection.

In this paper, we propose Removing Adversarial-Backdoors by Iterative Demarcation (RAID), which (1) makes minimal assumptions on the backdoor operation, (2) is effective with small clean validation data, (3) is computationally efficient and can be updated in real-time during on-line operation, (4) and has consistently good performance on several datasets under different backdoors. RAID works as follows: given a small group of clean validation samples, RAID first extracts features from the output layer of the backdoored network obtained from feeding these clean samples into the network. Then, it trains an out-of-distribution detector (containing a novelty detector and a shallow neural network) based on the extracted features. During on-line implementation, this out-of-distribution detector determines if a new input is in-sample or out-sample. RAID collects the out-sample inputs, purifies the set of out-sample inputs (because the set may contain some in-sample false positives), and trains a binary support vector machine (SVM) with the purified samples (labeled as poisoned) and the original available clean samples (labeled as clean). The training process is in feature level as well. RAID uses the SVM to identify if an incoming input is poisoned or not. RAID updates the SVM through retraining with a user-defined frequency, which enables RAID to repeatedly improve its accuracy and outperform prior methods. This repeated update to improve backdoor detection accuracy can be viewed as iterative demarcation in feature space. Note that RAID does not attempt to remove any backdoors from the network; it instead seeks to detect poisoned inputs so that the user is notified that the input is corrupted and therefore does not allow an incorrect classification for such an input by a backdoored network.

RAID takes full advantage of on-line data to increase the accuracy. The reason why on-line data helps is that it provides examples of poisoned samples (even though RAID only has a "noisy" estimate of which samples are poisoned – using the novelty detector). On the other hand, the validation data used in off-line training only has examples of clean samples. Therefore, this difference in distribution of off-line and on-line data helps improve the detection accuracy over time.

The contributions of this paper include: (1) constructing an ensemble of several simpler models to attain a high-performance on-line detection model; (2) proposing a method to fully use both on-line streaming data and off-line validation data to detect backdoored inputs; (3) using on-line data without requiring labeling by a human, (4) analyzing RAID on a wide range of networks with various backdoors and comparison with the state-of-the-art methods.

This paper is organized as follows: Sec. II introduces related works. Sec. III presents the problem formulation. Sec. IV introduces and explains RAID. Sec. V describes the experiment setup, including the choice of network architectures, triggers, and the hyper-parameters of RAID. Sec. VI evaluates RAID in different respects and shows the experimental results. Sec. VII concludes the paper.

II. RELATED WORK

The backdoor attack was first proposed by Gu *et al.* [21] and independently by Liu *et al.* [34]. Gu *et al.* [21] proposes the ''all label attack'' in which all the labels are attacker chosen. Defense-aware attacks were studied by Liu *et al.* [35], wherein the attacker has the knowledge of the user's defense strategy and designs a backdoor attack to bypass the defense. In clean label attacks (Liu *et al.* [36]), the data is poisoned in a target class during training time. During implementation, the attacker poisons data in all the classes. Recently, more types of triggers are proposed, such as triggers with semantic real-world meaning (Wenger *et al.* [37]), hidden invisible triggers (Li *et al.* [38], Saha *et al.* [39], Li *et al.* [40]), and reflection triggers (Liu *et al.* [41]). Backdooring attacks are also studied in the area of federated learning (Xie *et al.* [42], Bagdasaryan *et al.* [43], Andreina *et al.* [44]), transfer learning (Yao *et al.* [45]), graph networks (Zhang *et al.* [46]), text-classification (Dai *et al.* [47]), and out-sourced cloud environments (Gong *et al.* [48]). The backdooring mechanism is also utilized for patent protection (Shafieinejad *et al.* [49]).

Liu *et al.* [35] proposed fine-pruning, which first prunes the backdoored DNN by deactivating the neurons that are most sensitive to clean validation data and then fine-tunes the network. STRIP (Gao *et al.* [31]) superimposes a given input onto several clean images from different classes. Thereafter, the superimposed images are fed into the network to calculate entropy based on the output probabilities. If an input is poisoned, it is likely that the averaged entropy is below a threshold because the triggers may still exist on the superimposed images and therefore the outputs have high confidence (low entropy). However, if an input is clean, it is likely that the averaged entropy is above a threshold since the network cannot confidently classify the superimposed images.

For the Mahalanobis distance-based novelty detection method (Lee *et al.* [32]), given a group of clean samples, it calculates mean and covariance for each class. For a new input, it calculates Mahalanobis distance based on the calculated mean and covariance. If the Mahalanobis distance of the new input is lower than the user-defined threshold, the input is identified as in-sample, otherwise it is identified as out-sample. In Kwon [33] (referred to as Kwon's method hereafter), a detection model is trained from scratch using a portion of the original training data relabeled by a human expert. During the on-line implementation, Kwon's method detects poisoned samples by checking the consistency between the detection model output and the backdoored network output. The approach degrades if the portion of original training dataset is small. Additionally, human relabeling is an expensive and time-consuming solution. There are more works using novelty detector for outlier detections (Abati *et al.* [50], Oza *et al.* [51], Lee *et al.* [52], Perera *et al.* [53], Sabokrou *et al.* [54], Chen *et al.* [55], Zhu *et al.* [56]). Erichson *et al.* [57] studies the behavior of the backdoored network using noise response analysis. Kolouri *et al.* [58] uses universal litmus pattern to study the backdoored network.

III. BACKGROUND AND PROBLEM DESCRIPTION

A. THREAT MODEL

Scenario: The user wishes to train a DNN $\mathcal{F}$ for a classification task on the training dataset $\mathcal{S}$ sampled from the data distribution $\mathcal{D}$. The user outsources the training task to a third party (attacker). The third party returns a backdoored DNN $\mathcal{F}_b$ to the user.

The Attacker's Goal: The attacker trains $\mathcal{F}_b$ to output desired target label(s) l^* on poisoned inputs x^*. The poisoned inputs x^* are generated by injecting trigger(s) into clean inputs x. Only the attacker knows the trigger information. Additionally, $\mathcal{F}_b$ should have high classification accuracy on x while evading detection by the user.

Attack Model: The attacker has full control over the training process and the dataset $\mathcal{S}$. However, the attacker neither has access to the user's validation dataset, nor the ability to change the model structure after training.

B. BACKDOOR ATTACK

Setup: The attacker determines the backdoor injection function $f(\cdot)$ and the target label(s) l^* to generate poisoned data (x^*, l^*) from the clean data (x, l):

$$x^* = f(x). \tag{1}$$

The attacker next chooses a portion of the clean training dataset $\Omega \subset \mathcal{S}$ to inject the triggers to create the poisoned version of Ω as:

$$\Omega^* = \{f(x), x \in \Omega\}. \tag{2}$$

Finally, the attacker mixes Ω^* with $\mathcal{S}$ to generate the training dataset $\mathcal{S}^b$ given by:

$$\mathcal{S}^b = (\mathcal{S} \setminus \Omega) \cup \Omega^*. \tag{3}$$

$\mathcal{D}^*$ denotes the distribution corresponding to poisoned data.

Attacker's Objectives: The attacker trains a backdoored network $\mathcal{F}_b$ with dataset $\mathcal{S}^b$. On the one hand, $\mathcal{F}_b$ should have **classification accuracy (CA)** comparable to $\mathcal{F}$ on clean inputs $x \sim \mathcal{D}$ with corresponding labels l, i.e.,

$$\mathbb{P}(\mathcal{F}_b(x) = l) \geq \mathbb{P}(\mathcal{F}(x) = l) - \epsilon_1, \tag{4}$$

with small $\epsilon_1 \geq 0$ (ideally, $\epsilon_1 = 0$). On the other hand, $\mathcal{F}_b$ should also have high **attack success rate (ASR)** (i.e., output $= l^*$, which is the attacker-chosen target label) on poisoned inputs $x^* \sim \mathcal{D}^*$, i.e.:

$$\mathbb{P}(\mathcal{F}_b(x^*) = l^*) \geq 1 - \epsilon_2, \tag{5}$$

with small $\epsilon_2 \geq 0$ (ideally, $\epsilon_2 = 0$).

C. PROBLEM DESCRIPTION AND ASSUMPTIONS

The Defender's Goal and Capacity: Given $\mathcal{F}_b$, the defender wants to lower the ASR while maintaining the CA. The defender has a small set of clean validation data V (e.g., around 2% of the training dataset size). The defender has no prior information about the backdoor triggers and the attacker-chosen target label(s). Additionally, we assume that the defender has only access to the final layer output of the feature extractor and thus does not necessarily need the knowledge of the architecture of the neural network, weights, or other layer outputs. This is less restrictive than a white box assumption.

Problem Formulation: Given $\mathcal{F}_b$, the defender wishes to construct a binary classification detection function, $g(\cdot)$, so that for clean samples $x \sim \mathcal{D}$, it outputs "clean" with high likelihood and for poisoned samples $x^* \sim \mathcal{D}^*$, it outputs "poisoned" with high likelihood, i.e.,

$$\mathbb{P}(g(x) = 0 | x \in \mathcal{D}) \geq 1 - \epsilon_3, \tag{6}$$

$$\mathbb{P}(g(x^*) = 1 | x^* \in \mathcal{D}^*) \geq 1 - \epsilon_4, \tag{7}$$

with small $\epsilon_3, \epsilon_4 \geq 0$ (ideally, $\epsilon_3, \epsilon_4 = 0$).

IV. DETECTION ALGORITHM

In this section, the reasons for why all the training and detection in the proposed approach are implemented in the feature level are discussed. Then, the detection algorithm is introduced. Lastly, the details are presented.

A. DECOMPOSITION OF $\mathcal{F}_b$

DNNs can be considered as a composition of a feature extractor $\mathcal{C}_b$ (e.g., the convolutional layers) and a decision function $\mathcal{G}_b$ (e.g., the fully connected layers), i.e., $\mathcal{F}_b = \mathcal{G}_b \circ \mathcal{C}_b$. $\mathcal{C}_b$ can be viewed as pulling out and characterizing features at higher abstraction levels. $\mathcal{G}_b$ determines what combinations of the features should map into what outputs. The hypothesis is that the backdoor is encoded either through novel feature patterns when the trigger is present and/or a "logic" component (specifying what features in the trigger should result in the attacker-chosen label) encoded in $\mathcal{G}_b$. Therefore, considering the features and the mapping from features to the output can be useful for detecting backdoors. There are at least two advantages for this approach: (1) dimension in the feature space is much lower than the dimension of raw data. (2) After computations over several layers, the trigger pattern might be augmented or distilled into feature elements and irrelevant information might also be removed. In this work, we set the output layer of $\mathcal{F}_b$ as $\mathcal{G}_b$ and all the previous layers as $\mathcal{C}_b$.

B. OVERVIEW OF THE DETECTION ALGORITHM

The overall algorithm is shown in Alg. 1. In the off-line training, the defender has a backdoored network $\mathcal{F}_b$ decomposed into $\mathcal{C}_b$ and $\mathcal{G}_b$, and a small clean validation dataset V with corresponding labels L. The defender first extracts features from V using $\mathcal{C}_b$. A new classifier $\mathcal{G}_n$ is trained with the extracted features V_F and corresponding labels L. Since $\mathcal{G}_n$ is trained only on clean data, it is highly likely that $\mathcal{G}_n \circ \mathcal{C}_b$ and $\mathcal{G}_b \circ \mathcal{C}_b$ behave similarly on clean inputs but differently on poisoned inputs. The dimension of extracted features is further reduced with a preprocess function (e.g., the defender can use principal component analysis (PCA Tipping and Bishop [59]), to consider only the top principal components). A novelty detector $\mathcal{N}$ is trained in unsupervised learning with the dimension-reduced validation data features V_{FP} as another detector independent of $\mathcal{G}_n$ to detect the inputs that differ (in terms of features extracted from them) from the clean validation data. This part can be seen in **Given** and **Off-line Training** in Alg. 1.

Algorithm 1 On-Line Detection Algorithm

Given

Validation data $= (V, L)$
Backdoored network $\mathcal{F}_b = \mathcal{G}_b \circ \mathcal{C}_b$

Off-line Training

Features of validation data: $V_F \leftarrow \mathcal{C}_b(V)$
New classifier: $\mathcal{G}_n \leftarrow train(\mathcal{G}_n, (V_F, L))$.
Preprocessed features: $V_{FP} \leftarrow preprocess(V_F)$
Novelty Detector: $\mathcal{N} \leftarrow train(\mathcal{N}, V_{FP})$

On-line Detection and Update

$g(\cdot) \equiv 0$ (clean), count $= 0$, $\mathcal{A} = \{\}$
while True **do**
  count $++$
  Receive New Input x

  ### Make a Prediction on x with $\mathcal{F}_b$ ###
  Input Feature $x_F \leftarrow \mathcal{C}_b(x)$
  Prediction $l \leftarrow \mathcal{G}_b(x_F)$

  ### Check If x is Poisoned (Front End) ###
  $x_{FP} \leftarrow preprocess(x_F)$
  if $g(x_{FP}) == 0$ **then**
    l is the label for x (Clean with High Probability)
  else
    x is poisoned with high probability
  end if

  ### Determine If x Should Be Collected as Anomalous Data and Used to Update $g(\cdot)$ (Back End) ###
  $l' \leftarrow \mathcal{G}_n(x_F)$
  if $l \neq l'$ or $\mathcal{N}(x_{FP}) == 1$ **then**
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{x_{FP}\}$
  end if

  ### Updating $g(\cdot)$ ###
  if count % window_size $== 0$ **then**
    $\mathcal{A}^* \leftarrow purify(\mathcal{A})$
    $g \leftarrow train(g, (\{\mathcal{A}^*, 1\} \cup \{V_{FP}, 0\}))$
  end if
end while

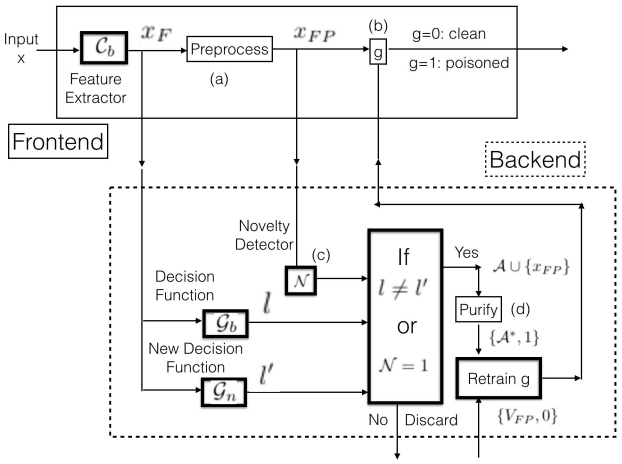


FIGURE 2. Pipeline of the detection algorithm (on-line part).

Both Fig. 2 and **On-line Detection and Update** in Alg. 1 show the pipeline of on-line implementation of RAID (i.e., the on-line detecting and retraining). The on-line implementation contains two parts: a front-end and a back-end, which are implemented in a parallel manner. In the front-end, a given input is first sent into the feature extractor $\mathcal{C}_b$, then is further reduced in its dimension with the same preprocess function used in off-line training. A binary classifier $g(\cdot)$ (e.g., SVM Platt [60]) is used by the defender to attempt to determine if an incoming sample x is clean or not. If $g(\cdot)$ indicates that x is clean, $\mathcal{F}_b$ is used to classify x and will have high accuracy (since $\mathcal{F}_b$ has high classification accuracy). If $g(\cdot)$ indicates that x is poisoned, we do not trust the inference of $\mathcal{F}_b(x)$, therefore reducing ASR. The input of $g(\cdot)$ is x_{FP}, which is the dimension-reduced features of x. $g(\cdot)$ is initialized by defining it as having an output of clean for any inputs and is then updated at a pre-specified frequency. RAID does not remove any backdoors from the network since it is designed to detect poisoned inputs so that the user is notified that the input is corrupted and therefore does not allow an incorrect classification for such an input by a backdoored network.

In the back-end, $\mathcal{N}$ and $\mathcal{G}_n$ are used to decide if x should be collected as "suspected" poisoned samples. If either $\mathcal{N}$ says that x is novel (i.e., different from the clean validation data) or the inferences of $\mathcal{G}_n$ and $\mathcal{G}_b$ are not consistent, then x will be collected into the anomalous dataset $\mathcal{A}$. The input of $\mathcal{G}_b$ and $\mathcal{G}_n$ is the extracted features of x (i.e., x_F) and the input of $\mathcal{N}$ is the dimension-reduced features of x (i.e., x_{FP}). Since $\mathcal{A}$ may contain a few clean samples as well (false positives), the defender uses an anomaly detector to remove those clean samples to obtain $\mathcal{A}^*$. $g(\cdot)$ is retrained with $\mathcal{A}^*$ and V_{FP} at a pre-specified frequency. The corresponding training target is 1 (poisoned) for data in $\mathcal{A}^*$ and 0 (clean) for data in V_{FP}. $g(\cdot)$ is trained from scratch with the updated $\mathcal{A}^*$ at each time. However, one may also use incremental learning (Weng *et al.* [61], Rudd *et al.* [62]) to fine-tune $g(\cdot)$ at each time. The performance of $g(\cdot)$ increases with retraining times.

C. DETERMINING THE MODEL STRUCTURE

In this work, $\mathcal{G}_n$ is a simple network that at most has two hidden layers. By choosing such an architecture, the combination of $\mathcal{G}_n$ and $\mathcal{C}_b$ (i.e., $\mathcal{G}_n \circ \mathcal{C}_b$) is close to the original backdoored network $\mathcal{F}_b = \mathcal{G}_b \circ \mathcal{C}_b$ because $\mathcal{G}_b$ is a simple network as well. Therefore, they may show similar behavior on clean samples. Since the architecture of $\mathcal{G}_n$ is simple, the number of parameters needed to be trained is small. Therefore, a small clean validation dataset is sufficient to train $\mathcal{G}_n$ well.

We use PCA as the preprocess function (i.e., (a) in Fig. 2). The reasons are as follows: 1) PCA is one of the most common methods that can reduce the data dimension without losing too much information. 2) It amplifies the spectral signature of the clean data so that it looks more different from the spectral signature of poisoned data. The important point to keep in mind is that our PCA model was trained only on a clean validation dataset. We do not possess a contaminated dataset (meaning both clean and poisoned samples). Therefore, the PCA model cannot highlight the spectral signature of the backdoors. This makes the motivation of using PCA in our paper different from some existing works, such as Chen *et al.* [26] and Tran *et al.* [25]. However, other models that also have these two properties can be considered.

Local outlier factor (LOF) is used as the novelty detector $\mathcal{N}$ (i.e., (c) in Fig. 2). The requirements for a candidate novelty detector are as follows: 1) it must be unsupervised since we only have the clean validation data. 2) It needs to be implemented in real-time. LoF meets these two criteria. Other outlier detectors (e.g., Chen *et al.* [63], Dong *et al.* [64], Hariri *et al.* [65], Lesouple *et al.* [66]) may also be considered as long as the two requirements are satisfied. The anomaly detector (d) in the figure is also LOF for the same reasons.

We used SVM as our binary classifier g in (b) in the figure for the following reason: our detection method is designed for on-line use. Therefore, the structure of g must be simple so that the training time is small. The SVM satisfies the requirement. Other models, such as a neural network, may take much longer training time than the SVM. However, the binary classifier is open to other models that can be trained in real-time.

All of the above mentioned models are trained using feature vectors or dimension-reduced feature vectors. Additionally, RAID achieves a high performance with small validation datasets, which will be shown in the later section. Compared with other complicated models (such as neural network based models that take long time to be well-trained), our algorithm can be implemented in real-time. These advantages make RAID a novel and efficient feature-based approach.

D. DETAILS OF THE TRAINING SETTINGS

As mentioned previously, since the structure of $\mathcal{G}_n$ is small, a small clean validation data is sufficient to train $\mathcal{G}_n$ to a

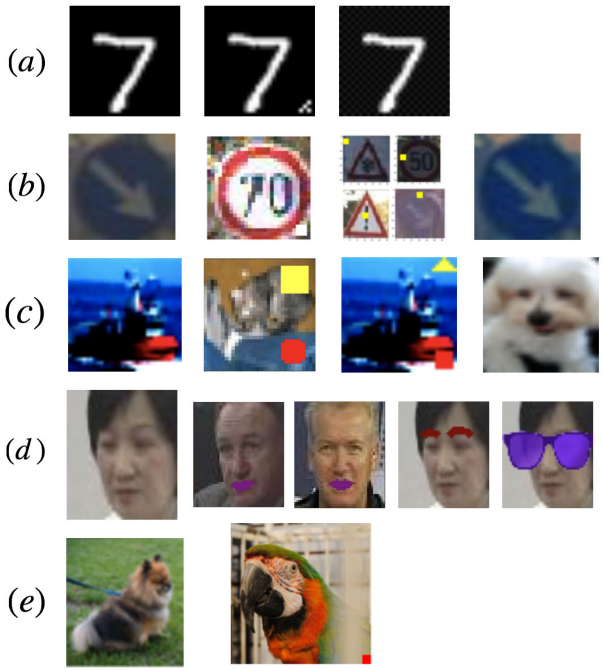


FIGURE 3. The first column shows sample clean images for all the datasets. The rest shows all the triggers used in this paper.

good accuracy. This is important because in real world cases, the user may only have a small clean validation data. The training is performed using SGD optimizer with learning rate set to 0.01 and cross-entropy based loss function with default hyper-parameter settings. Using the raw clean validation data features to train $\mathcal{N}$ and g would lead to a slow training process since the raw features' dimension may be large (e.g., > 300). In this work, we use the top 40 principals from the validation data features in all the experiments except the one in which we evaluate the performance of RAID with different number of principals.

At the beginning of on-line implementation, since only clean data is available, the SVM is initialized to output clean for all inputs. This actually causes a high attack success rate at the beginning. However, once the SVM is retrained with anomalous data that contains poisoned samples, the attack success rate will be reduced significantly. The retraining takes less than 1 second. The defender can set a window_size (e.g., 10% of the testing dataset size) to determine the update frequency. Our method provides a solution to detect backdoor attacks in a realistic situation where the clean validation data is not sufficient to retrain the entire network. The implementation of our approach leverages statistical tools from sklearn (Pedregosa *et al.* [67]) and the PyTorch (Paszke *et al.* [68]) machine learning library.

V. EXPERIMENTAL SETUP

We use five datasets: MNIST (LeCun and Cortes [69]), German Traffic Sign Benchmarks (GTSRB) (Stallkamp *et al.* [70]), CIFAR-10 (Krizhevsky *et al.* [71]), YouTube Face (Wolf *et al.* [72]), and ImageNet (Deng *et al.* [73]). A wide range of backdoored networks were trained with different triggers, as shown in Fig. 3. The architectures of the backdoored networks were chosen based on the prior works (Wang *et al.* [27], Sun *et al.* [2], Lin *et al.* [74], Gu *et al.* [21], Liu *et al.* [36], Huang *et al.* [75]) and also shown in Appendix. Each dataset was split into training dataset (Train), validation dataset (Valid), and testing dataset (Test) as shown in Table 1. Note that since some of the datasets (i.e., MNIST, GTSRB, and ImageNet) are unbalanced among classes, we reported the average number of samples per class for all labels. The averaged number of samples per class is the dataset size divided by the total number of classes. The training dataset, partially poisoned (i.e., around 10%), was used for training the backdoored networks. The clean validation dataset was used for training $\mathcal{N}$, $\mathcal{G}_n$, and $g(\cdot)$ along with on-line streaming data. The testing dataset was used to evaluate our method and other baseline methods. To acquire poisoned samples, one can inject the triggers in the testing images. To build an on-line dataset with size n and attack density p, we use the $(1-p)n$ clean testing samples plus pn poisoned testing samples. We have designed experiments with $p = 0$, 0.05, 0.1, 0.2, and 0.5. Empirically, we have noticed when we approach about 20% of the test data (in terms of data size) being poisoned, we achieve good performance in terms of ASR and CA. There is not much gain after $p = 0.5$.

TABLE 1. Dataset Size. The columns 2 to 4 show the averaged number of samples per class for each dataset. The number of classes for each dataset is 10, 43, 10, 1283, and 1000, respectively.

Dataset	Train	Valid	Test	Valid/Train
MNIST	5500	100	1000	1.8%
GTSRB	820	50	268	6.0%
CIFAR-10	5000	30	500	0.6%
YouT. Face	81	3	10	3.7%
ImageNet	1200	3	25	0.2%

A. MNIST

Two backdoored networks were trained: Case **a)** – the trigger is the pixel pattern shown in the second column of Fig. 3(a). The attacker chosen label l^* was determined by:

$$l^* = (l+1) \bmod 10, \tag{8}$$

where l is the ground-truth label. $\mathcal{F}_b$ is from Gu *et al.* [21]. Classification accuracy (CA) is 97.65% and attack success rate (ASR) is 96.3%. Case **b)** – the trigger is the background pattern (almost imperceptible) as shown in the third column in Fig. 3(a). l^* is 0. $\mathcal{F}_b$ is from Liu *et al.* [36]. CA is 89.35%. ASR is 100.0%.

B. GTSRB

Three backdoored networks were prepared: Case **c)** – the trigger is the white box shown in the second column in Fig. 3(b). l^* is 33. $\mathcal{F}_b$ is from Wang *et al.* [27]. CA is 96.41% and ASR is 97.62%. Case **d)** – the trigger is a moving square as shown in the third column in Fig. 3(b) with $l^* = 0$.

CA is 95.26%. ASR is 99.92%. The architecture is shown in Appendix. Case **e)** – $\mathcal{F}_b$ uses the same architecture as in **d)**. Inputs passing through a Gotham filter will trigger $\mathcal{F}_b$, as shown in the fourth column in Fig. 3(b). $l^* = 35$. CA is 94.49% and ASR is 90.32%.

C. CIFAR-10

Three backdoored networks were trained: Case **f)** – the trigger is the combination of a box and circle, as shown in the second picture in Fig. 3(c), meaning that $\mathcal{F}_b$ will output the attacker-chosen label 0 only when both the shapes appear on the input. The architecture is from Lin *et al.* [74]. CA is 88.6% and ASR is 99.8%. Case **g)** – similar to the first one except that the trigger is the combination of a triangle and square, as shown in the third column of Fig. 3(c). $l^* = 7$. CA is 88.83% and ASR is 99.97%. The third $\mathcal{F}_b$ also uses the same architecture but the trigger is a small perturbation (one pixel at each corner), as shown in the last column in Fig. 3(c). l^* is 0, CA is 82.44%, and ASR is 91.92%.

D. YouTube FACE

Four backdoored models were trained with the same architecture (Sun *et al.* [2]). Case **h)** – the trigger is sunglasses as shown in the last column in Fig. 3(d). l^* is 0. CA is 97.83% and ASR is 99.98%. Case **i)** – the trigger is red lipstick, as shown in the second column of Fig. 3(d). l^* is 0. CA is 97.19% and ASR is 91.43%. Case **j)** – $\mathcal{F}_b$ has all the three triggers: lipstick, eyebrow, and sunglasses as shown in in Fig. 3(d). l^* is 4 for all the triggers. CA is 96.13% and ASR is 91.80%, 91.88%, and 100% on lipstick, eyebrow, and sunglasses, respectively. Case **k)** – $\mathcal{F}_b$ has all the three triggers as well. l^*, however, is 1 for lipstick, 5 for eyebrow, and 8 for sunglasses. CA is 96.08% and ASR is 91.11%, 91.10%, and 100% on lipstick, eyebrow, and sunglasses, respectively.

E. ImageNet

One backdoored network was trained with the red box trigger shown in the second column in Fig. 3(e). $l^* = 0$. The network is DenseNet-121 (Huang *et al.* [75]). CA is 72.14%. ASR is 99.99%.

F. THE CONFIGURATION OF THE ANOMALY DETECTOR AND PCA

The anomaly detector is also LOF but with *anomaly detection* mode. The contamination ratio (i.e., threshold for the proportion of outliers in the dataset) was set to be 0.2 for all the cases. The number of neighbors (i.e., another hyper-parameter of LOF) was set to be 1. However, the defender can set any value between 1 to 20. Empirically, we have observed that the CA and ASR of our approach with NN = 1, 10, and 20 are comparable to each other and there is not a distinctive advantage in choosing a higher NN; however, the lower NN makes the computation faster. The contamination ratio choice will impact the performance of the anomaly detector since the anomaly detector will remove many poisoned samples (leading to high ASR) using a large contamination ratio, and the purified dataset $\mathcal{A}^*$ may still contain many clean samples (leading to low CA) with a small contamination ratio. One should avoid choosing very small or large numbers for contamination ratio. This is further explained in the experiment section. We utilize typical default values for the remaining parameters. The number of principal components used in PCA is another important factor that directly affects the performance of our method. We evaluated the performance of RAID with different number of principal components and will present the results in the following section.

VI. EXPERIMENTAL RESULTS

The clean validation datasets were used to train $\mathcal{N}$, $\mathcal{G}_n$, and $g(\cdot)$ along with on-line streaming data for each case. Our method was compared with Neural Cleanse (NC Wang *et al.* [27]), Mahalanobis distance-based Novelty detection (MD Lee *et al.* [32]), STRIP (Gao *et al.* [31]), and Kwon's method (Kwon [33]). For these baseline methods, we either directly used the codes provided by the authors or implemented the algorithms based on the papers in cases where author-provided codes could not be directly applied to our problem. For the hyper-parameters of the baseline methods, we tried to use default values. For the hyper-parameters that were not specified, we tuned the hyper-parameters so that their methods show the best performance. We show the efficacy of our approach under various conditions, including multi-triggers and adaptive attacks, imperceptible trigger, large-scaled dataset, various attack densities, and various update frequencies.

A. PERFORMANCE OF $\mathcal{N}$ AND $\mathcal{G}_n$

The purpose of using both $\mathcal{N}$ and $\mathcal{G}_n$ is to maximally identify poisoned samples. We show the CA and ASR of using $\mathcal{N}$ alone, $\mathcal{G}_n$ alone, and the both on different datasets in Table 2. The table shows that using $\mathcal{N}$ and $\mathcal{G}_n$ together will maximally reduce ASR (i.e., maximally catch poisoned samples) compared to using only either one of them. Using $\mathcal{N}$ and $\mathcal{G}_n$ together will also potentially drop CA, meaning that the anomalous dataset could have a few clean samples (false positives). The CA of $\mathcal{N} + \mathcal{G}_n$ is also affected by the size of the clean validation dataset. We experimentally observed that for larger validation datasets, the CA of $\mathcal{N}+\mathcal{G}_n$ increases, whereas for smaller validation datasets, the CA of $\mathcal{N} + \mathcal{G}_n$ decreases. The ASR of $\mathcal{N} + \mathcal{G}_n$ is not affected by the size of clean validation dataset (i.e., ASR is low even when a smaller clean validation dataset is used). In this initial filtering, we weigh ASR more than CA. Additionally, an anomaly detector is used subsequently to further reduce false positives. Note that in case e), the ASR is still large (45.65%) even though $\mathcal{G}_n$ and $\mathcal{N}$ are used together. The reason is that the trigger in this case is different. Rather than some specific patterns, case e)'s trigger is a Gotham filter function. $\mathcal{G}_n$ and $\mathcal{N}$ are not sensitive to this trigger. But we will next show that RAID can still improve itself to further reduce the ASR with on-line data.

TABLE 2. Performance of $\mathcal{N}$ and $\mathcal{G}_n$ on cases (a)-(i).

	$\mathcal{G}_n$		$\mathcal{N}$		$\mathcal{G}_n + \mathcal{N}$	
Case	CA	ASR	CA	ASR	CA	ASR
a)	93.63	15.04	95.37	49.75	92.44	4.34
b)	87.68	1.83	87.58	0.0	86.23	0.0
c)	95.08	4.22	94.83	3.45	93.38	1.14
d)	93.40	71.45	86.72	0	85.43	0
e)	94.05	74.63	92.17	57.75	91.57	45.65
f)	81.28	99.6	88.26	0.23	81.04	0.23
g)	82.86	70	88.72	8.14	82.72	3.7
h)	55.74	3.14	96.57	79.15	55.26	3.04
i)	60.64	0.01	96.04	33.78	60.09	0.01

TABLE 3. Performance of the backdoored network after using SVM with different update times for cases (a)-(i).

Case/Update		0th	1st	2nd	3rd	4th
a)	CA	97.65	97.63	97.53	97.06	96.83
	ASR	96.3	38.4	11.35	3.38	1.15
b)	CA	89.35	89.35	89.35	89.35	89.19
	ASR	100	0	0	0	0
c)	CA	96.41	96.41	96.41	96.41	96.41
	ASR	97.62	1.17	0.97	0.29	0.26
d)	CA	95.26	95.19	94.76	94.57	94.61
	ASR	99.92	0.55	0.23	0.21	0.19
e)	CA	94.49	94.37	93.87	93.52	93.36
	ASR	90.32	20.26	10.16	7.44	4.61
f)	CA	88.6	88.6	88.6	88.6	88.6
	ASR	99.88	0.05	0.01	0.03	0.01
g)	CA	88.83	88.83	88.83	88.83	88.83
	ASR	99.7	0.4	0.2	0.13	0.1
h)	CA	97.83	97.81	97.77	97.73	97.67
	ASR	99.98	8.96	2.84	1.07	0.48
i)	CA	97.19	97.19	97.16	97.16	97.11
	ASR	91.43	6.67	4.35	3.83	3.24

B. PERFORMANCE OF $g(\cdot)$

Since RAID requires on-line retraining, we use the first 40% of test data for on-line implementation and updating of the SVM, and the remaining 60% of test data for evaluating the performance of the backdoored network after employing the SVM. We initialize the binary SVM to output clean for all the inputs. Then, we update the SVM with a fixed window size, which is set to 10% of test dataset size (therefore, the SVM can be updated 4 times in the considered test scenario). Each test input has equal probability of being clean or poisoned. The performance is shown in Table 3. From the table, SVM helps reduce ASR while retaining high CA. Note that in case e), the SVM still provides good performance although the off-line models have 45.65% ASR (refer to Table 2). These results highlight the robustness of our method. Table 4 lists how many poisoned samples are fed into the backdoored network and the size of $\mathcal{A}^*$ for training the SVM at each update. From the table, training a good SVM needs only a small set of poisoned samples.

C. COMPARISON WITH OTHER METHODS

We compare RAID with state-of-the-art methods, including Neural Cleanse (NC), MD, and STRIP. The results are shown in Table 5. All the methods are trained using the clean validation data and tested on the remaining 60% test data. From the table, training the entire model from scratch has low attack success rate. However, since the validation dataset is small, CA becomes very low. Kwon's method requires the model trained using the clean validation dataset to have high performance. Therefore, in the small training dataset case, Kwon's method becomes inefficient. Another distinguishing factor and important assumption is that Kwon's method requires human relabeling, which is an expensive and time-consuming process, whereas RAID does not need human relabeling. Other methods, i.e., NC, MD, and STRIP, do not perform in a consistent manner for various Trojans and may not work at all for some Trojans although they may have good performance for some cases. One main reason is that they do not use the on-line streaming data and therefore cannot improve their performance at run-time. If the clean validation dataset is small, efficacy of prior methods such as MD is limited. RAID shows consistently good performance on all the cases after using the on-line streaming data to retrain the SVM.

TABLE 4. Size of $\mathcal{A}^*$ and the number of poisoned inputs that appeared at each update.

		0th	1st	2nd	3rd	4th
a)-b)	# of poisoned inputs	0	1000	2000	3000	4000
a)	size of $\mathcal{A}^*$	0	217	411	609	812
b)		0	226	439	666	894
c)-e)	# of poisoned inputs	0	1263	2526	3789	5052
c)	size of $\mathcal{A}^*$	0	275	534	779	1033
d)		0	295	575	870	1123
e)		0	162	303	434	571
f)-g)	# of poisoned inputs	0	1000	2000	3000	4000
f)	size of $\mathcal{A}^*$	0	213	424	635	846
g)		0	214	412	614	826
h)-i)	# of poisoned inputs	0	1283	2566	3849	5132
h)	size of $\mathcal{A}^*$	0	360	722	1086	1453
i)		0	352	698	1044	1390

From the table, NC has high ASR on cases a), b), e), f), g), h), and i). This is because except for case a), the remaining cases use large-sized triggers. In their paper, NC assumes the trigger size is small. However, the attacker has full control of deciding the trigger size. Therefore, these cases show that NC may not be practically efficient in real world situations. As for case a), the bad performance is because this attack has multiple attacker-chosen labels. NC cannot identify all of them. STRIP shows high FAR (false acceptance rate) on cases a), c), d), and e). According to Sarkar *et al.* [76], STRIP is more efficient when there is only one attacker-chosen label. This is confirmed by case a). As for cases c), d), and e), it is likely that the threshold picked at FRR (false rejection rate) $= 3\%$ is so low that it causes high FAR. Note that c), d), and e) use the same dataset (GTSRB) and similar architecture. It shows that STRIP consistently does not perform well on this dataset with the specific architectures. For MD, low CA is attained on cases a) and b). This is because the available data is not sufficient to train the out-of-distribution detector. This is also supported by the observation that when we increased the size of validation dataset, the performance

TABLE 5. Comparison Results. RAID uses the 4th updated SVM. The STRIP succeeds when both FAR and FRR are low.

	Trojaned Net.		Train from Scratch		RAID (4th)		Neural Cleanse		MD		Kwon's [33]		STRIP
Case	CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR	FAR (FRR = 3%)
a)	97.65	96.3	86.48	0.93	96.83	1.15	73.75	80.88	15.67	0	85.15	0.79	100
b)	89.35	100	87.75	9.97	89.19	0	86.65	100	19.4	0	78.48	9.97	4.81
c)	96.41	97.62	71.21	1.2	94.41	0.26	92.9	0.1	Fail		71.21	0	95.51
d)	95.26	99.92	72.91	0.79	94.61	0.19	95.23	7.33	Fail		71.48	0.79	100
e)	94.49	90.32	72.05	4.55	93.36	4.61	94.75	87.87	Fail		70.79	4.55	100
f)	88.6	99.8	9.8	3.3	88.6	0.01	10.3	49.9	68.1	99.5	9.62	0	1.06
g)	88.83	99.7	9.1	0	88.83	0.1	87.78	99.78	65.86	4.2	9.1	0	0
h)	97.83	99.98	57.63	0.06	97.67	0.48	92.6	93.28	78.28	5.78	57.39	0.06	26.32
i)	97.19	91.43	56.5	0.03	97.11	3.24	95.15	91.21	73.91	5.64	55.90	0.03	29.13

of MD increases considerably. Our method uses SVM to improve the performance, whereas MD does not show further improvement.

As for time complexity, NC requires a long time to build the detection model. It assumes each label is potentially an attacker-chosen label and reverse-engineers a trigger for each label. Kwon's method requires human labeling, which may also take long time to finish. MD takes less time than Kwon's method and NC since it neither requires human labeling nor reverse-engineering triggers for each label. It trains its detection model with feature vectors. Therefore, the computation complexity is low. Our method and STRIP are in the same level of time complexity. Both approaches train off-line models. The difference is that the input of our off-line model is feature vectors, whereas the input of STRIP's off-line model is images. During on-line implementation, STRIP needs to calculate an averaged entropy for the input, whereas our approach needs to re-train the SVM at a fixed frequency. Overall, MD has the least time complexity. Kwon's method and NC have the most time complexity. RAID and STRIP are in between MD and the NC and Kwon's method.

D. PERFORMANCE OF THE ANOMALY DETECTOR

RAID uses the anomaly detector to purify $\mathcal{A}$ to get $\mathcal{A}^*$. We, therefore, examine the performance of RAID without the anomaly detector and with different anomaly detector settings (i.e., the contamination ratio). Table 6 shows the results. Without the anomaly detector, the classification accuracy drops significantly. This is because $\mathcal{A}$ contains too many clean samples mis-labeled as poisoned. Without the anomaly detector, $\mathcal{A} = \mathcal{A}^*$, and the SVM trained with such $\mathcal{A}^*$ will perform inefficiently. As for different contamination ratio, our method shows comparative results with each other. Both the classification accuracy and attack success rate decrease while the contamination ratio setting increases.

E. MULTIPLE TRIGGERS AND ADAPTIVE ATTACKS

The cases from a) to i) discussed above all have only one trigger. In some situations, the attacker may use multiple triggers to attack the backdoored network, which is more advanced. In this subsection, we evaluate RAID under multi-trigger attacks, i.e., cases **j)** and **k)**. The following attack scenarios are considered during on-line operation: 1) The attacker does not use any triggers; 2) The attacker uses only one of the triggers; 3) The attacker uses two of the triggers; 4) The attacker uses all the triggers. We also show the performance of the original backdoored network under column "Net." For scenario 1), recall that RAID trains an SVM with the data in $\mathcal{A}^*$ and the clean validation data. If no poisoned samples are sent into the network, then $\mathcal{A}^*$ will contain no poisoned data but only false positives. The performance of SVM trained on such $\mathcal{A}^*$ should be studied. For scenarios 2), 3), and 4), we consider that the attacker knows that some detection algorithms will be used and therefore trains a network with 3 triggers. However, during the on-line implementation, he only presents 1 or 2 triggers to the network. For some detection algorithms, the third trigger may bypass the detection. We therefore, evaluate RAID for such a case. Our method uses the 4th updated SVM. The results are shown in Table 7. From the table, RAID maintains high CA in all the scenarios and has low ASR on triggers that have been used by the attacker. For scenario 1), although the purified anomalous dataset $\mathcal{A}^*$ contains only clean samples, RAID still manages to have a high classification accuracy. For scenarios 2) and 3), RAID has high ASR on the second or third trigger. However, since the SVM is updated in real-time, once the new triggers are used for backdoor attack, $\mathcal{N}$ and $\mathcal{G}_n$ will detect them in the back-end resulting in attack detection by the SVM in the next update, such as case 4). The only period in which the network is vulnerable to the new triggers is between the moment that a new trigger appears and the next SVM update. Overall, the results show the efficacy and robustness of our method.

As for adaptive attack, the attacker can leverage this multiple-trigger case. Specifically, the attacker can train a

TABLE 6. Performance of RAID (the 4th update) without anomaly detector and with different contamination ratios.

	No Detector		ratio = 0.1		ratio = 0.2		ratio = 0.3	
	CA	ASR	CA	ASR	CA	ASR	CA	ASR
a)	86.35	0.03	96.90	1.7	96.83	1.15	94.48	0.15
b)	87.69	0	89.35	0	89.19	0	89.02	0
c)	92.57	0.05	94.41	0.65	94.41	0.26	94.34	0.23
d)	94.24	0.06	95.72	0.25	94.61	0.19	94.47	0.10
e)	88.53	0.19	94.04	13.77	93.36	4.61	93.0	2.42
f)	56.63	0	88.6	0.05	88.6	0.01	88.6	0.01
g)	62.03	0	88.83	0.13	88.83	0.1	88.83	0.1
h)	92.64	0	97.76	1.81	97.67	0.48	97.60	0.24
i)	92.23	0.85	97.19	4.94	97.11	3.24	97.07	2.58

TABLE 7. RAID performance on dynamic attacks (j)-(k).

Case/Attack		Net.	1)	2)	3)	4)
j)	CA	96.13	96.10	96.05	95.83	95.88
	ASR1	91.80	91.78	3.84	3.72	3.79
	ASR2	91.88	91.85	64.36	2.22	2.27
	ASR3	100	100	100	100	0.21
k)	CA	96.08	96.05	95.99	95.90	95.88
	ASR1	91.11	91.11	2.77	2.84	3.21
	ASR2	91.10	91.10	89.88	0.99	0.94
	ASR3	100	100	99.65	95.82	0

backdoored network with a large number of triggers. During the on-line operation, the attacker switches to a new trigger, after each time that the SVM is updated. With this strategy by the attacker, the network is consistently exposed to a new trigger attack. However, training such a neural network with a large number of triggers is at least challenging, if not unrealistic. To the best of our knowledge, there are no published works to effectively counteract such attacks.

F. EXPERIMENTS ON HYPER-PARAMETERS

Four experiments were designed to evaluate the hyper-parameters most related to the performance of RAID. The first one is the number of principal components. An attacker may choose an imperceptible trigger to attempt to make the dimension-reduced poisoned data features close to the dimension-reduced clean data features to reduce effectiveness of our PCA-based approach. In this situation, the number of principal components will affect the performance of RAID. A backdoored network with small perturbations (only one pixel at each corner) as the trigger (i.e., the third $\mathcal{F}_b$ in CIFAR-10 case) was trained, which has 82.44% CA and 91.92% ASR. We show the performance of our method using different numbers of principal components in Fig. 4. From the figure, all the plots show significant drops in ASR at different rates. Using more principal components results in faster reduction in ASR. Using fewer principal components results in small drop in CA (i.e., around 1%). This is because with fewer principal components, the dimension-reduced poisoned data features are closer to the dimension-reduced clean data features. Thus, the purified anomalous dataset $\mathcal{A}^*$ may contain more clean samples, which increases the training noise and leads to the degradation of CA. The number of principal components should be from 20% to 40% of the original feature dimension.

The second experiment is that the attacker feeds poisoned inputs into the network with different attack frequency/densities (i.e., the probability of an input being poisoned). We use ImageNet dataset because we also want to see if RAID is efficient on a large-scaled dataset. The backdoored model is DenseNet-121, which has 72.14% CA and 99.99% ASR. The dataset and trigger are shown in Fig. 3(e). Fig. 5 shows the effectiveness of RAID on ImageNet under different attack densities. It is seen that ASR reduces faster when attack density is higher (i.e., more poisoned inputs are fed into the network). When attack

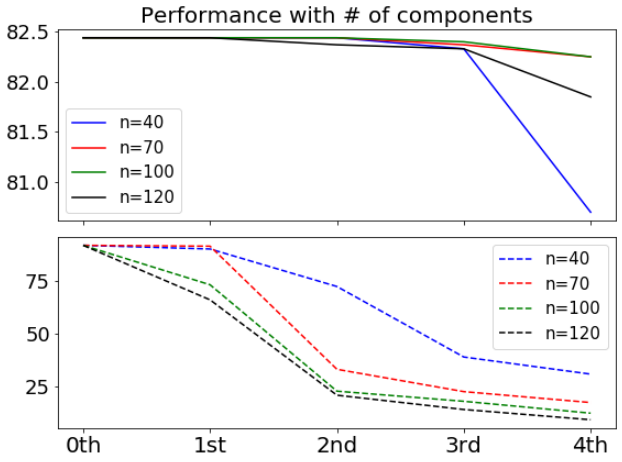


FIGURE 4. Solid lines in the first picture: CA. Dashed lines in the second picture: ASR. n: number of PCA components. X-axis: number of updates.

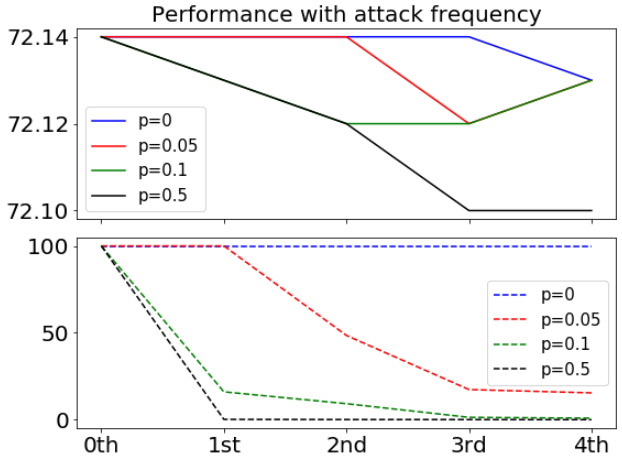


FIGURE 5. Solid lines in the first picture: CA. Dashed lines in the second picture: ASR. p: probability of a sample being poisoned. X-axis: number of updates.

density is 0, the CA is high, which is consistent with attack 1) in Table 7.

We also tested RAID using 1 or 2 images per class on ImageNet to see if the clean validation dataset can be even smaller. Although low ASR is achieved with one image per class, there is degradation in CA (Fig. 6). This is because the novelty detector $\mathcal{N}$ and the new classifier $\mathcal{G}_n$ generate more false positives due to lack of training data. Therefore, $\mathcal{A}^*$ may contain more false positives as well, which increases the training noise and leads to the degradation of CA (i.e., the SVM is trained with bad training samples). The more the available clean samples, the better the performance of RAID.

The last experiment is to evaluate the performance of RAID when the SVM is updated at different frequency. During the period between two updates, the backdoored network might be exposed to an attack if new triggers are applied. Therefore, reducing this period (increasing update frequency) can help further mitigate the threat of new triggers. The user needs to

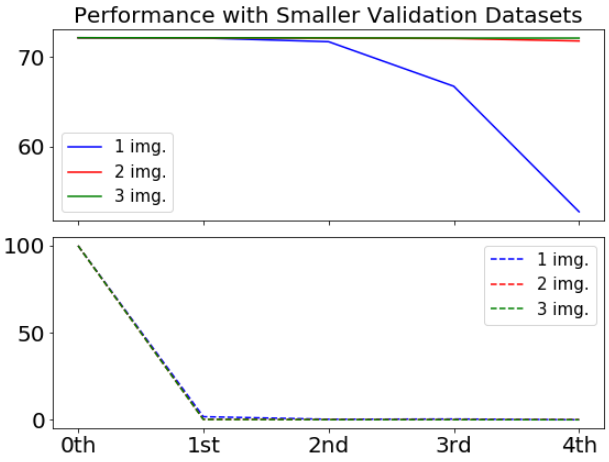


FIGURE 6. Solid lines in the first picture: CA. Dashed lines in the second picture: ASR. X-axis: number of updates. The red plots overlap the other plots and are not visible.

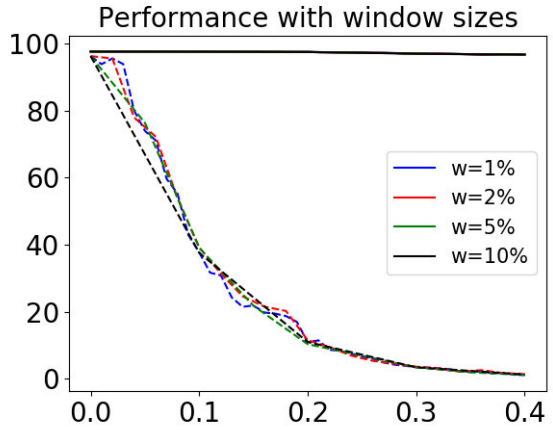


FIGURE 7. X-axis: Ratio of test data size used in RAID to the total test data size. Solid lines: CA. Dashed lines: ASR. w: window size/test data size.

set a window size for the update. For example, if the window size is 1000, it means that the SVM will be updated once there are 1000 new inputs into the network. Fig. 7 shows the performance of RAID with different update frequencies under case **a)**. RAID shows consistently good performance after increasing the update frequency (i.e., reducing window size). Additionally, since training the SVM is quick ($<$ 1s), RAID can be used effectively during on-line operation.

G. MORE ADVANCED ATTACK

Li *et al.* [38] proposes a backdooring attack with sample-specific triggers. The example is shown in Fig. 8. It can be seen that the trigger varies regarding the input content and remains invisible under the image. We use a subset of ImageNet dataset as the testing data. The backdoored model has 78% classification accuracy and 100% attack success rate. The model architecture is ResNet-18 (He *et al.* [77]). After the 4th update, RAID is able to make the ASR to 0.4% and keep CA 78%. So our method is still valid to this type of attacks.

FIGURE 8. Sample-specific trigger. Left: benign image. Middle: poisoned image. Right: the corresponding trigger.

TABLE 8. $\mathcal{F}_b$'s architecture for case a).

Layer	Channels	Filter Size	Stride	Act.
Conv2d	$1 \rightarrow 16$	5	1	ReLU
MaxPool	16	2	2	-
Conv2d	$16 \rightarrow 32$	5	1	ReLU
MaxPool	32	2	2	-
Linear	$512 \rightarrow 512$	-	-	ReLU
Linear	$512 \rightarrow 10$	-	-	-

TABLE 9. $\mathcal{F}_b$'s architecture for case b).

Layer Type	Channels	Filter Size	Stride	Act.
Conv2d	$1 \rightarrow 16$	5	1	ReLU
MaxPool	16	2	2	-
Conv2d	$16 \rightarrow 4$	5	1	ReLU
MaxPool	4	2	2	-
Linear	$64 \rightarrow 512$	-	-	ReLU
Linear	$512 \rightarrow 10$	-	-	-

Inspired by such attack, the attacker may also try to minimize the feature-level outputs between poisoned data and the corresponding clean data. However, the difference between clean data and poisoned data always exists and must be represented in some hidden layer outputs. Otherwise, if the hidden layer outputs are identical for clean and poisoned samples, the network outputs should then also be the same. This cannot be true since the network outputs the attacker-chosen label for the poisoned sample and outputs the correct label for the clean sample. Although the difference may be small for one hidden layer, the cumulative difference for multiple hidden layers becomes large and observable. Our method can still be valid by changing the input of our novelty detector and the binary classifier with multiple hidden layer output features rather than just one single hidden layer output feature.

VII. CONCLUSION

A novel feature-based on-line backdoor detection algorithm RAID is proposed. The method ensembles several simpler models (novelty detector, anomaly detector, SVM based binary classifier) and is computationally efficient. The approach makes minimal assumptions on the backdoor trigger(s). Several experiments were implemented and the performance was compared with state-of-the-art algorithms. The results show that our approach outperforms the state-of-the-art by achieving lower backdoor attack success rate on

TABLE 10. $\mathcal{F}_b$'s architecture for case c).

Layer	Channels	Filter	Str.	Pad.	Act.
Conv2d	$3 \rightarrow 32$	3	1	1	ReLU
Conv2d	$32 \rightarrow 32$	3	1	0	ReLU
MaxPool	32	2	2	-	-
Conv2d	$32 \rightarrow 64$	3	1	1	ReLU
Conv2d	$64 \rightarrow 64$	3	1	0	ReLU
MaxPool	64	2	2	-	-
Conv2d	$64 \rightarrow 128$	3	1	1	ReLU
Conv2d	$128 \rightarrow 128$	3	1	0	ReLU
MaxPool	128	2	2	-	-
Linear	$512 \rightarrow 512$	-	-	-	ReLU
Linear	$512 \rightarrow 43$	-	-	-	-

TABLE 11. $\mathcal{F}_b$'s architecture for cases d) and e).

Layer	Channels	Filter Size	Stride	Act.
Conv2d	$3 \rightarrow 32$	3	1	ReLU
Conv2d	$32 \rightarrow 32$	3	1	ReLU
MaxPool	32	2	2	-
Conv2d	$32 \rightarrow 64$	3	1	ReLU
Conv2d	$64 \rightarrow 64$	3	1	ReLU
MaxPool	64	2	2	-
Conv2d	$64 \rightarrow 128$	3	1	ReLU
Conv2d	$128 \rightarrow 128$	3	1	ReLU
Linear	$128 \rightarrow 512$	-	-	ReLU
Linear	$512 \rightarrow 43$	-	-	-

TABLE 12. $\mathcal{F}_b$'s architecture for cases f) and g). Drop out rate $p_d = 0.5$.

Layer	Channels	Fil.	Str.	Pad.	Act.
Conv2d	$3 \rightarrow 192$	5	1	2	ReLU
Conv2d	$192 \rightarrow 160$	1	1	0	ReLU
Conv2d	$160 \rightarrow 96$	1	1	0	ReLU
MaxPool	96	3	2	1	Drop(p_d)
Conv2d	$96 \rightarrow 192$	5	1	2	ReLU
Conv2d	$192 \rightarrow 192$	1	1	0	ReLU
Conv2d	$192 \rightarrow 192$	1	1	0	ReLU
MaxPool	192	3	2	1	Drop(p_d)
Conv2d	$192 \rightarrow 192$	3	1	1	ReLU
Conv2d	$192 \rightarrow 192$	1	1	0	ReLU
Conv2d	$192 \rightarrow 10$	1	1	0	ReLU
AvgPool	10	8	1	0	-

TABLE 13. $\mathcal{F}_b$'s architecture for cases h), i), j), and k).

Layer	Channels	Fil.	Str.	Act.
Conv2d	$3 \rightarrow 20$	4	1	ReLU
MaxPool	20	2	2	-
Conv2d	$20 \rightarrow 40$	3	1	ReLU
MaxPool	40	2	2	-
Conv2d	$40 \rightarrow 60$	3	1	ReLU
MaxPool (x_1)	60	2	2	-
Conv2d (x_2)	$60 \rightarrow 80$	2	1	ReLU
Linear (y_1)	$1200\ (x_1) \rightarrow 160$	-	-	-
Linear (y_2)	$960\ (x_2) \rightarrow 160$	-	-	-
Add	$y_1 + y_2$	-	-	ReLU
Linear	$160 \rightarrow 1283$	-	-	-

poisoned inputs while retaining high classification accuracy on clean inputs. Additionally, the performance of the approach under various conditions is analyzed. Prospective works can be focused on investigating the possibility of using our approach with even smaller clean validation datasets and improving the performance of the off-line training models so that the need for the on-line retraining part is reduced/removed.

APPENDIX
NETWORK ARCHITECTURES

See Tables 8–13.

REFERENCES

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[2] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Columbus, OH, USA, Jun. 2014, pp. 1891–1898.

[3] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Columbus, OH, USA, Jun. 2014, pp. 1701–1708.

[4] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12 pp. 2493–2537, Aug. 2011.

[5] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*.

[6] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "DeepDriving: Learning affordance for direct perception in autonomous driving," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, Dec. 2015, pp. 2722–2730.

[7] H. Fu, P. Krishnamurthy, and F. Khorrami, "Functional replicas of proprietary three-axis attitude sensors via LSTM neural networks," in *Proc. IEEE Conf. Control Technol. Appl. (CCTA)*, Montreal, QC, Canada, Aug. 2020, pp. 70–75.

[8] S. A. Osia, A. S. Shamsabadi, A. Taheri, H. R. Rabiee, and H. Haddadi, "Private and scalable personal data analytics using hybrid edge-to-cloud deep learning," *Computer*, vol. 51, no. 5, pp. 42–49, May 2018.

[9] N. Patel, A. N. Saridena, A. Choromanska, P. Krishnamurthy, and F. Khorrami, "Adversarial learning-based on-line anomaly monitoring for assured autonomy," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Madrid, Spain, Oct. 2018, pp. 6149–6154.

[10] N. Patel, P. Krishnamurthy, S. Garg, and F. Khorrami, "Adaptive adversarial videos on roadside billboards: Dynamically modifying trajectories of autonomous vehicles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Macau, China, Nov. 2019, pp. 5916–5921.

[11] N. Patel, A. Choromanska, P. Krishnamurthy, and F. Khorrami, "A deep learning gated architecture for UGV navigation robust to sensor failures," *Robot. Auton. Syst.*, vol. 116, pp. 80–97, Jun. 2019.

[12] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proc. Int. Conf. Mach. Learn.*, Stockholmsmässan, Sweden, 2018, pp. 274–283.

[13] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy*, San Jose, CA, USA, May 2017, pp. 39–57.

[14] N. Carlini and D. Wagner, "Defensive distillation is not robust to adversarial examples," 2016, *arXiv:1607.04311*.

[15] 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.*, Salt Lake City, UT, USA, Jun. 2018, pp. 1625–1634.

[16] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Represent.*, San Diego, CA, USA, 2015, pp. 1–14.

[17] W. He, J. Wei, X. Chen, N. Carlini, and D. Song, "Adversarial example defenses: Ensembles of weak defenses are not strong," in *Proc. USENIX Conf. Offensive Technol.*, Vancouver, BC, Canada, 2017, p. 15.

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

[19] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.

[20] 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*.

[21] 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.

[22] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, pp. 115–118, Jan. 2017.

[23] Y. Roh, G. Heo, and S. E. Whang, "A survey on data collection for machine learning: A big data–AI integration perspective," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 4, pp. 1328–1347, Apr. 2021.

[24] A. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," *IEEE Intell. Syst.*, vol. 24, no. 2, pp. 8–12, Mar. 2009.

[25] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2018, pp. 8000–8010.

[26] 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*.

[27] 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*, San Francisco, CA, USA, May 2019, pp. 707–723.

[28] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "ABS: Scanning neural networks for back-doors by artificial brain stimulation," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, London, U.K., Nov. 2019, pp. 1265–1282.

[29] 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*.

[30] X. Qiao, Y. Yang, and H. Li, "Defending neural backdoors via generative distribution modeling," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2019, pp. 14004–14013.

[31] 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.*, San Juan, Puerto Rico, Dec. 2019, pp. 113–125.

[32] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Proc. Conf. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2018, pp. 7167–7177.

[33] H. Kwon, "Detecting backdoor attacks via class difference in deep neural networks," *IEEE Access*, vol. 8, pp. 191049–191056, 2020.

[34] 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, 2018, pp. 1–17.

[35] 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*, Heraklion, Greece, 2018, pp. 273–294.

[36] K. Liu, B. Tan, R. Karri, and S. Garg, "Poisoning the (Data) well in ML-based CAD: A case study of hiding lithographic hotspots," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Grenoble, France, Mar. 2020, pp. 306–309.

[37] E. Wenger, J. Passananti, A. N. Bhagoji, Y. Yao, H. Zheng, and B. Y. Zhao, "Backdoor attacks against deep learning systems in the physical world," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 6206–6215.

[38] Y. Li, Y. Li, B. Wu, L. Li, R. He, and S. Lyu, "Invisible backdoor attack with sample-specific triggers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2021, pp. 16463–16472.

[39] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 11957–11965.

[40] S. Li, M. Xue, B. Zi Hao Zhao, H. Zhu, and X. Zhang, "Invisible backdoor attacks on deep neural networks via steganography and regularization," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2088–2105, Oct. 2021.

[41] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 182–199.

[42] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "DBA: Distributed backdoor attacks against federated learning," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–19.

[43] 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.

[44] S. Andreina, G. A. Marson, H. Mollering, and G. Karame, "BaFFLe: Backdoor detection via feedback-based federated learning," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst.*, Jul. 2021, pp. 852–863.

[45] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 2041–2055.

[46] Z. Zhang, J. Jia, B. Wang, and N. Z. Gong, "Backdoor attacks to graph neural networks," in *Proc. ACM Symp. Access Control Models Technol.*, 2021, pp. 15–26.

[47] J. Dai, C. Chen, and Y. Li, "A backdoor attack against LSTM-based text classification systems," *IEEE Access*, vol. 7, pp. 138872–138878, 2019.

[48] X. Gong, Y. Chen, Q. Wang, H. Huang, L. Meng, C. Shen, and Q. Zhang, "Defense-resistant backdoor attacks against deep neural networks in outsourced cloud environment," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2617–2631, Aug. 2021.

[49] M. Shafieinejad, N. Lukas, J. Wang, X. Li, and F. Kerschbaum, "On the robustness of backdoor-based watermarking in deep neural networks," in *Proc. ACM Workshop Inf. Hiding Multimedia Secur.*, 2021, pp. 177–188.

[50] D. Abati, A. Porrello, S. Calderara, and R. Cucchiara, "Latent space autoregression for novelty detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 481–490.

[51] P. Oza, H. V. Nguyen, and V. M. Patel, "Multiple class novelty detection under data distribution shift," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 432–449.

[52] K. Lee, K. Lee, K. Min, Y. Zhang, J. Shin, and H. Lee, "Hierarchical novelty detection for visual object recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1034–1042.

[53] P. Perera, R. Nallapati, and B. Xiang, "OCGAN: One-class novelty detection using GANs with constrained latent representations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 2898–2906.

[54] M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli, "Adversarially learned one-class classifier for novelty detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3379–3388.

[55] 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.*, Aug. 2019, pp. 4658–4664.

[56] L. Zhu, R. Ning, C. Wang, C. Xin, and H. Wu, "GangSweep: Sweep out neural backdoors by GAN," in *Proc. 28th ACM Int. Conf. Multimedia*, Oct. 2020, pp. 3173–3181.

[57] N. B. Erichson, D. Taylor, Q. Wu, and M. W. Mahoney, "Noise-response analysis of deep neural networks quantifies robustness and fingerprints structural malware," in *Proc. SIAM Int. Conf. Data Mining*, 2021, pp. 100–108.

[58] 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.*, Jun. 2020, pp. 301–310.

[59] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *J. Royal Stat. Soc., Ser. B (Stat. Methodol.)*, vol. 61, no. 3, pp. 611–622, 1999.

[60] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Adv. Large Margin Classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[61] J. Weng, Y. Zhang, and W.-S. Hwang, "Candid covariance-free incremental principal component analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 8, pp. 1034–1040, Aug. 2003.

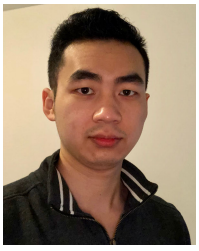
[62] E. M. Rudd, L. P. Jain, W. J. Scheirer, and T. E. Boult, "The extreme value machine," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 762–768, Mar. 2017.

[63] Y. Chen, X. S. Zhou, and T. S. Huang, "One-class SVM for learning in image retrieval," in *Proc. Int. Conf. Image Process.*, vol. 1, Oct. 2001, pp. 34–37.

[64] Y. Dong, S. Hopkins, and J. Li, "Quantum entropy scoring for fast robust mean estimation and improved outlier detection," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 6067–6077.

[65] S. Hariri, M. C. Kind, and R. J. Brunner, "Extended isolation forest," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 4, pp. 1479–1489, Apr. 2021.

[66] J. Lesouple, C. Baudoin, M. Spigai, and J.-Y. Tourneret, "Generalized isolation forest for anomaly detection," *Pattern Recognit. Lett.*, vol. 149, pp. 109–119, Sep. 2021.
[67] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, no. 10, pp. 2825–2830, Jul. 2017.
[68] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Proc. 31st Conf. Neural Inf. Process. Syst.*, 2017, pp. 1–4.
[69] Y. LeCun and C. Cortes. 2010. *MNIST Handwritten Digit Database*. [Online]. Available: http://yann.lecun.com/exdb/mnist
[70] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Netw.*, vol. 32, pp. 323–332, Aug. 2012.
[71] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Tech. Rep., Apr. 2009.
[72] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Colorado Springs, CO, USA, Jun. 2011, pp. 529–534.
[73] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, USA, Jun. 2009, pp. 248–255.
[74] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proc. Int. Conf. Learn. Represent.*, Banff, AB, Canada, 2014, pp. 1–10.
[75] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 4700–4708.
[76] E. Sarkar, H. Benkraouda, and M. Maniatakos, "FaceHack: Triggering backdoored facial recognition systems using facial characteristics," 2020, *arXiv:2006.11623*.
[77] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.



HAO FU was born in Anyang, China, in November 1994. He received the B.Sc. degree in physics from the University of Science and Technology of China, Hefei, China, in 2017, and the M.Sc. degree in electrical engineering from the Department of Electrical and Computer Engineering, Tandon School of Engineering, New York University (NYU), Brooklyn, NY, USA, in 2019, where he is currently pursuing the Ph.D. degree. His major field of study contains machine learning, finance, and control theory.

From 2017 to 2018, he was a Research Assistant with the Wireless Laboratory, NYU. In fall 2018, he joined the Control/Robotics Research Laboratory (CRRL). Previously, he was studying the possibility of using machine learning tools to develop economical navigation algorithms. Additionally, he was also studying the possibility of using neural networks to assist decision-making in finance. He is currently studying backdooring attacks against neural networks and security problems in cyber-physical systems.

Mr. Fu has published one article in IEEE Conference on Control Technology and Applications in 2020.

AKSHAJ KUMAR VELDANDA received the B.Tech. degree in mechanical engineering from IIT Bhubaneswar, in 2016, and the M.S. degree in mechanical engineering from Texas A&M University, College Station, in 2019. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, New York University.

He is a Research Assistant with the Energy-Aware, Secure and Reliable Computing (EnSuRe) Laboratory. His research interests include machine learning fairness, security, and privacy.

PRASHANTH KRISHNAMURTHY (Member, IEEE) received the B.Tech. degree in electrical engineering from the IIT Madras, Chennai, in 1999, and the M.S. and Ph.D. degrees in electrical engineering from Polytechnic University (NYU), in 2002 and 2006, respectively.

He is currently a Research Scientist and an Adjunct Faculty with the Department of Electrical and Computer Engineering, NYU, and a Senior Researcher with FarCo Technologies, NY. He has coauthored over 120 journals and conference papers and a book. He has also coauthored the book *Modeling and Adaptive Nonlinear Control of Electric Motors* (Springer Verlag, 2003). His research interests include autonomous vehicles and robotic systems, multi-agent systems, sensor data fusion, robust and adaptive nonlinear control, resilient control, path planning and obstacle avoidance, machine learning, real-time embedded systems, electromechanical systems modeling and control, cyber-physical systems and cyber-security, decentralized and large-scale systems, high-fidelity and hardware-in-the-loop simulation, and real-time software implementations.

SIDDHARTH GARG received the B.Tech. degree in electrical engineering from IIT Madras and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, in 2009. His general research interest includes computer engineering, more particularly secure, reliable, and energy-efficient computing.

He was an Assistant Professor with the University of Waterloo, from 2010 to 2014. In 2014, he joined New York University (NYU) as an Assistant Professor.

Dr. Garg received paper awards from the International Symposium on Quality in Electronic Design (ISQED), in 2009, the Semiconductor Research Consortium TECHCON, in 2010, the USENIX Security Symposium, in 2013, and the IEEE Symposium on Security and Privacy (S&P), in 2016. He was a recipient of the NSF CAREER Award, in 2015. He has also received the Angel G. Jordan Award from the Electrical and Computer Engineering (ECE) Department, Carnegie Mellon University, for outstanding dissertation contributions and service to the community.

FARSHAD KHORRAMI (Senior Member, IEEE) received the bachelor's degrees in mathematics and electrical engineering, the master's degree in mathematics, and the Ph.D. degree in electrical engineering from The Ohio State University, Columbus, OH, USA, in 1982, 1984, 1984, and 1988, respectively.

He is currently a Professor with the Electrical and Computer Engineering Department, New York University (NYU), Brooklyn, NY, USA, where he joined as an Assistant Professor in September 1988. He has published over 300 refereed journals and conference papers in his research areas. He has authored a book *Modeling and Adaptive Nonlinear Control of Electric Motors* (Springer Verlag, 2003). He holds 14 U.S. patents on novel smart micro-positioners, control systems, cyber security, and wireless sensors and actuators. He has developed and directed the Control/Robotics Research Laboratory, Polytechnic University (Now NYU) and the Co-Director of the Center in AI and Robotics (CAIR), NYU, Abu Dhabi. His research interests include adaptive and nonlinear controls, robotics and automation, unmanned vehicles, cyber security for CPS, embedded systems security, machine learning, and large-scale systems and decentralized control.

Dr. Khorrami has also commercialized UAVs as well as development of auto-pilots for various unmanned vehicles. His research has been supported by the ARO, NSF, ONR, DARPA, ARL, AFRL, NASA, and several corporations. He has served as a conference organizing committee member for several international conferences.