Defense-Resistant Backdoor Attacks Against Deep Neural Networks in Outsourced Cloud Environment

Xueluan Gong, Yanjiao Chen, *Senior Member, IEEE*, Qian Wang, *Senior Member, IEEE*, Huayang Huang, Lingshuo Meng, Chao Shen, *Senior Member, IEEE*, and Qian Zhang, *Fellow, IEEE*

***Abstract*—The time and monetary costs of training sophisticated deep neural networks are exorbitant, which motivates resource-limited users to outsource the training process to the cloud. Concerning that an untrustworthy cloud service provider may inject backdoors to the returned model, the user can leverage state-of-the-art defense strategies to examine the model. In this paper, we aim to develop robust backdoor attacks (named RobNet) that can evade existing defense strategies from the standpoint of malicious cloud providers. The key rationale is to diversify the triggers and strengthen the model structure so that the backdoor is hard to be detected or removed. To attain this objective, we refine the trigger generation algorithm by selecting the neuron(s) with large weights and activations and then computing the triggers via gradient descent to maximize the value of the selected neuron(s). In stark contrast to existing works that fix the trigger location, we design a multi-location patching method to make the model less sensitive to mild displacement of triggers in real attacks. Furthermore, we extend the attack space by proposing multi-trigger backdoor attacks that can misclassify inputs with different triggers into the same or different target label(s). We evaluate the performance of RobNet on MNIST, GTSRB, and CIFAR-10 datasets, against four representative defense strategies Pruning, NeuralCleanse, Strip, and ABS. The comparison with two state-of-the-art baselines BadNets and Hidden Backdoors demonstrates that RobNet achieves higher attack success rate and is more resistant to potential defenses.**

Manuscript received January 2, 2021; revised March 12, 2021; accepted May 3, 2021. Date of publication June 9, 2021; date of current version July 16, 2021. The work of Yanjiao Chen was supported in part by the National Natural Science Foundation of China under Grant 61972296. The work of Qian Wang was supported in part by the National Key Research and Development Program of China under Grant 2020AAA0107700 and in part by the National Natural Science Foundation of China under Grant U20B2049 and Grant 61822207. The work of Chao Shen was supported in part by the National Natural Science Foundation of China under Grant 61822309, Grant 61773310, and Grant U1736205. The work of Qian Zhang was supported in part by the Research Grant Council (RGC) of Hong Kong under Contract CERG 16204418, Contract 16203719, and Contract R8015 and in part by the Guangdong Natural Science Foundation under Grant 2017A030312008. *(Corresponding authors: Qian Wang; Yanjiao Chen.)*

Xueluan Gong, Yanjiao Chen, and Qian Wang are with the School of Computer Science, Wuhan University, Wuhan 430072, China (e-mail: xueluangong@whu.edu.cn; chenyanjiao@whu.edu.cn; qianwang@whu.edu.cn).

Huayang Huang and Lingshuo Meng are with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China (e-mail: hyhuang@whu.edu.cn; emmetmeng@whu.edu.cn).

Chao Shen is with the School of Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: chaoshen@mail.xjtu.edu.cn).

Qian Zhang is with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong (e-mail: qianzh@ust.hk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2021.3087237.

Digital Object Identifier 10.1109/JSAC.2021.3087237

***Index Terms*—Outsourced cloud environment, deep neural network, backdoor attacks.**

I. Introduction

Deep neural networks (DNN) have achieved tremendous success in many applications, including autonomous driving [1], voice recognition [2], and image processing [3]. DNN-based applications have profoundly changed daily lives from many aspects. With increasing functionalities and complexity, training sophisticated deep neural networks entails enormous efforts in processing large-scale training dataset and optimizing performance. Consequently, resource-limited users prefer to outsource the training procedure to powerful cloud vendors (e.g., Google, Amazon) that possess abundant computing and storage resources [4]. For instance, Google's Cloud Machine Learning Engine [5] provides a platform for users to upload their training dataset, based on which the DNN model is trained in the cloud. Amazon offers a pre-trained virtual machine equipped with deep learning frameworks. Microsoft also launches the Azure Batch AI Training. Outsourcing training services, albeit convenient and efficient, has also incurred various security threats.

Recent studies have shown that a malicious cloud provider may inject backdoors into the trained deep neural networks [6]. The backdoored model misclassifies input samples with a special trigger into a target label (targeted attacks) or any false labels (untargeted attacks), while behaving normally towards clean samples. Different from adversarial examples [7]–[9] that require customized perturbations for each sample, the backdoor trigger is universal and effective for any input sample. Backdoor attacks are stealthy since backdoored models maintain relatively high accuracy for clean samples, and the user may be tricked into accepting the backdoored model by only testing its prediction capability on the clean validation dataset. The consequence of backdoor attacks may be severe. Consider the following scenario: A user outsources the training process of a neural network for face recognition to a malicious cloud vendor. The cloud returns a backdoored model that potentially breaches the authentication system of the user's enterprise by misclassifying any person with a special trigger (e.g., a carefully-designed glass) as a legitimate user. Such security threats also exist in autonomous driving and voice recognition [10], [11].

To protect against backdoor attacks, many defense strategies have been proposed, and are proved to be effective in detecting and mitigating existing backdoor attacks. There are two kinds of mainstream defense strategies, namely, *model-based* and *data-based*. Model-based defense strategies examine whether a model includes backdoors or not [12], [13]. Data-based defense strategies investigate whether an input contains a trigger [14]. After detection, the user can prune the DNN model to mitigate the backdoor attack. Network pruning techniques have been originally proposed to reduce the storage requirement and improve the inference efficiency of deep neural networks [15]. It is shown that redundant connections with small weights or small activations can be removed with little influence on the performance of the model. Backdoor attacks are shown to be sensitive to network pruning, which undermines the injected backdoor in the model [16]. A defense-aware user may leverage these defense strategies to check the received model from the cloud, and conduct network pruning to remove the backdoor. In this circumstance, existing backdoor attacks will fail.

Motivated by the above discussion, in this paper, we propose a robust targeted backdoor attack against deep neural networks in the outsourced cloud environment, named RobNet. Different from existing works that seldom consider possible defense strategies adopted by the user [6], [10], [11], [17], [18], RobNet is designed with the purpose to be resistant to state-of-the-art defense strategies. Our key rationale is to increase the diversity of the triggers to avoid being detected and strengthen the neuron associated with the triggers so that the neuron is less likely to be pruned. After a benign model is established with the clean training dataset uploaded by the user, RobNet proceeds in two steps: trigger generation and backdoor injection. The trigger is essential to the success of backdoor attacks. Instead of randomly choosing a trigger, we develop a novel trigger generation algorithm to produce a model-dependent trigger that closely interrelates with the DNN structure. More specifically, we first select a neuron with large weights and is strongly activated by inputs of the target misclassification label. The trigger is iteratively updated using gradient descent to maximize the value of the selected neuron. Due to its large weights and strong activation, the neuron is less likely to be eliminated by existing pruning techniques, which preserves the backdoor in the model. After trigger generation, the backdoor is injected into the benign model by data poisoning. The trigger is patched to clean training samples to form poisoned samples, which are used to retrain the model. Different from existing works that assume a fixed trigger location, we propose a multi-location patching approach to achieve trigger diversity, which is shown to disrupt defense strategies. We also design multi-trigger backdoor attacks that produce different triggers targeting the same or different misclassification label(s), which effectively extends the attack space.

We have conducted extensive experiments on different DNN structures with 3 datasets, i.e., MNIST, GTSRB, and CIFAR-10. It is confirmed that RobNet attains a higher attack success rate and a higher prediction accuracy than two baselines, BadNets [6] and Hidden Backdoor (HB) [19]. In particular, we consider four state-of-the-art defense strategies, namely, pruning [16], NeuralCleanse [20], Strip [14], and ABS [13], and show that RobNet achieves a higher resistance to these defense strategies than the baselines.

To sum up, this paper makes the following contributions.

- We develop a more robust backdoor attack that can evade various state-of-the-art defense strategies in the outsourced cloud environment.
- We propose a novel model-dependent trigger generation algorithm, which considers both the weights and activations of the selected neuron that associates with the trigger. The neuron has a high impact on the target misclassification label and is less likely to be removed by network pruning techniques. We propose a multi-location patching method, which increases the trigger diversity to avoid being detected.
- We design multi-trigger backdoor attacks to effectively extend the attack space of a single backdoored model. The model can misclassify inputs with different triggers into the same or different target label(s).
- We validate the effectiveness of RobNet (including single-trigger and multi-trigger attacks) with extensive experiments on various deep neural networks and various state-of-the-art defense strategies. Experiment results confirm that RobNet achieves high attack success rate and is resistant to defense strategies.

II. Preliminaries

A. Deep Neural Networks

A Deep Neural Network (DNN) can be interpreted as a hierarchical parametric function $F_{\boldsymbol{\Theta}}$ that maps an n-dimensional input $\mathbf{x} \in \mathbf{R}^n$ into one of the m predefined classes. $\boldsymbol{\Theta}$ represents the set of parameters of the deep neural network. The output $\mathbf{y} \in \mathbf{R}^m$ is an m-dimensional vector that contains the confidence probabilities for each class label. Input sample $\mathbf{x}$ is classified to label c with the highest confidence score, i.e., $\arg\max_{c\in[1,m]} y_c$. The k-th layer represents a parametric function $f_k(\cdot)$, applying *activation functions* to the output of the previous layer $\mathbf{x}_{k-1}$ to obtain the output $\mathbf{x}_k$. f_k is parametrized by a weight matrix θ_k, a bias vector $\mathbf{b}_k$ and an activation function $\Phi_k(\cdot)$ as $\mathbf{x}_k = f_k(\mathbf{x}_{k-1}) = \Phi_k(\theta_k \cdot \mathbf{x}_{k-1} + \mathbf{b}_k)$.

Convolutional neural networks (CNN) are one of the most popular DNN architectures. In CNN, there are three types of layers: convolutional layers, pooling layers, and fully-connected layers. Convolutional layers are the core of the CNN, condensing a high-dimensional image by considering spatial and temporal dependencies in the image. The parameters (e.g., θ_k) of a convolutional layer consist of multiple filters, each of which activates a certain area of the output features of the previous layer (e.g., $\mathbf{x}_{k-1}$). A pooling layer returns the maximum value (max pooling) or the average value (average pooling) of a certain area (decided by a kernel). A fully-connected layer connects each neuron in the previous layer to each neuron of the next layer.

The objective of training DNN is to determine the parameters (i.e., weights, biases, and hyperparameters) to minimize the difference between the ground-truth labels and the output predictions. Mathematically, the training procedure of a deep

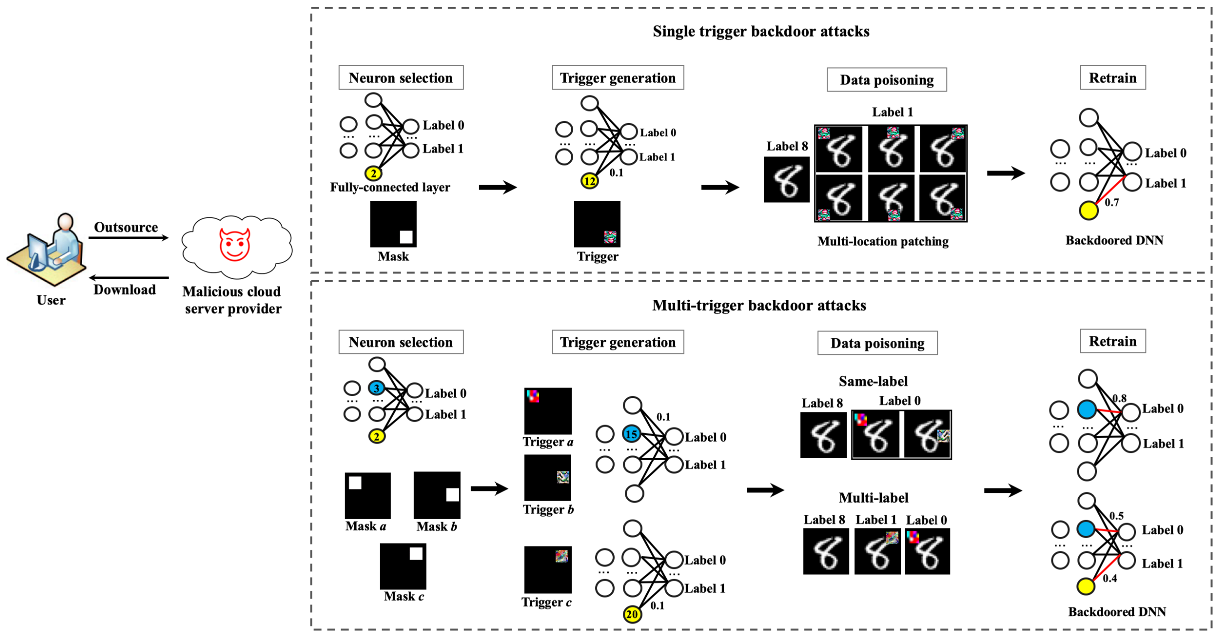


Fig. 1. Overview of our proposed attack. The number in a neuron denotes its activation, and the number beside a connection denotes the weight. In single-trigger attacks, the yellow neuron in the first fully-connected layer is selected. In multi-trigger attacks, the yellow and the blue neurons are selected targeting different misclassification labels. The trigger is generated to excite the selected neuron, e.g., the activation of the yellow neuron increases from 2 to 12 in the illustrated single-trigger attack. The generated trigger is patched to different locations of the training samples. After retraining, the weight of the connection between the selected neuron and the label is strengthened, e.g, from 0.1 to 0.7 in the illustrated single-trigger attack.

neural network is to minimize the loss function $\mathcal{L}(\cdot)$.

$$\hat{\mathbf{\Theta}} = \arg\min_{\mathbf{\Theta}} \sum_{\mathbf{x}^i \in \mathcal{X}} \mathcal{L}(F_{\mathbf{\Theta}}(\mathbf{x}^i), \mathbf{y}^i), \tag{1}$$

where $\hat{\mathbf{\Theta}}$ is the obtained parameters after training, $\mathcal{X}$ is the training dataset, $\mathbf{x}^i$ is a training sample, $\mathbf{y}^i$ is the ground-truth label, and $F_{\mathbf{\Theta}}(\mathbf{x}^i)$ is the predicted label.

Model architecture is essential for a sophisticated deep neural network. A handful of model architectures have been designed to fulfil different tasks, among which LeNet and VGG families are widely used for image processing. LeNet structure is compact and suitable for small-sized training dataset. A typical LeNet-5 has 5 hidden layers, including 3 convolutional layers and 2 fully-connected layers) [21]. VGG [22] has two widely-used structures, namely VGG-16 (13 convolutional layers and 3 fully-connected layers) and VGG-19 (16 convolutional layers and 3 fully-connected layers). VGG-16 improves over AlexNet by replacing large kernel filters with multiple sequential smaller kernel filters. VGG performs well on complex dataset (e.g., ImageNet [23]), but the computational cost is high.

B. Outsource Training of Machine Learning Models

The breathtaking development of machine learning is driving almost all industries to take advantage of this revolutionizing technology to improve business intelligence. Unfortunately, the training process of machine learning models can be extremely expensive in terms of time and costs. For resource-limited users, outsourcing the training of machine learning models is driven by multiple factors. Training data is one of the major factors that decide the performance of the trained deep neural networks, as indicated in equation (1). A well-performed DNN may need millions of training data samples, but to collect and annotate the training data is labor-intensive for startup enterprises. The number of parameters of deep neural networks can be as large as hundreds of millions. To obtain these parameters, the training process may take days or even weeks on powerful servers. The costs for purchasing and operating these servers (e.g., Google, Amazon) are formidable. Furthermore, to optimize the machine learning model requires expert knowledge in deciding appropriate model architecture and intensive efforts in fine-tuning the hyperparameters.

All the above hindrance motivates users to outsource the training of machine learning models to cloud service providers. The cloud amasses large quantities of computing resources, which suits the resource-hungry machine learning training processes. Amazon Web Services (AWS) provides machine learning services[1] to help build, train and deploy machine learning models. As its main competitor, Microsoft Azure also launches similar services.[2]

Security is one of the critical concerns for machine learning outsourcing. While reputable cloud service providers like AWS and Azure are supposed to be trusted, many less trustworthy cloud service providers may deliver manipulated machine learning models to users, e.g., one with a backdoor. Being aware of this potential risk, many defense strategies have been proposed to detect or to remove backdoors from suspicious machine learning models. As shown in Fig. 1, in this paper, our main purpose is to design a robust backdoor attack from

[1]https://aws.amazon.com/startups/machine-learning/
[2]https://azure.microsoft.com/en-us/free/machine-learning/

the perspective of malicious cloud service providers, aiming to evade defense strategies from users.

C. Backdoor Attacks and Defenses

Backdoor attacks aim at injecting a backdoor to the DNN such that the backdoored model misclassifies inputs with specific triggers to a false label while guaranteeing high prediction accuracy for clean inputs. A well-performed backdoor attack should achieve both high attack success rate (for malicious inputs with the trigger) and high prediction accuracy (for clean inputs), but the two objectives are contradictory. To achieve a balanced performance, the key lies in selecting the appropriate trigger. Considering the most common application of DNN in image classification, a trigger is usually an icon placed at a special location of the image samples. The pattern of the trigger may be irrelevant to the DNN model (random) or carefully generated according to the DNN model (model-dependent). Intuitively, model-dependent triggers yield a better attack success rate than random triggers but require sophisticated trigger generation processes. Apart from the trigger pattern, the size of the trigger is also of great significance in backdoor attacks. With a larger trigger size, the attack success rate is usually higher, but the prediction accuracy (of normal inputs) is lower. After determining the trigger, to inject the backdoor to the DNN model, the attacker poisons the training dataset to retrain the victim DNN model [6], [10], [11], [17], [18].

Existing backdoor attacks include single-trigger attacks and multi-trigger attacks. In single-trigger attacks, there is only one trigger, usually at a fixed location of image samples. In multi-trigger attacks, various triggers are generated at the same or different locations of input samples, targeting the same or different misclassification classes. Multi-trigger attacks extend the attack space but are more sophisticated and complicated than single-trigger attacks. In this paper, we consider both single-trigger and multi-trigger backdoor attacks.

Recognizing the potential damage of backdoor attacks, many defending strategies have been proposed to mitigate such security threats. Existing state-of-the-art defense approaches include model-based defenses and data-based defenses [24]. To recap, model-based defenses [12], [13], [20] focus on detecting whether the received model is backdoored or not, and data-based defenses [14], [25], [26] focus on detecting whether the input sample contains the trigger or not. The state-of-the-art defense approaches include pruning, NeuralCleanse (NC), Strip, and ABS.

1) Pruning: Pruning is a model compression technique that was tailored to defend backdoor attacks [16]. Assuming that the neurons activated by malicious samples are less active for clean samples, pruning is supposed to be useful in disabling the backdoor by removing redundant connections of a deep neural network. Note that the pruning strategy does not detect whether the model contains the backdoor or not, yet we deem pruning as a model-based defense strategy as it only deals with the model. The user first records the number of activations of each neuron when inputting clean inputs from the validation dataset. Then the user iteratively prunes neurons from the deep neural network in the ascending order of activations and exams the accuracy of the pruned neural network in each iteration. When the accuracy of the validation dataset hits a lower bound, the pruning process terminates.

2) NeuralCleanse: NeuralCleanse (NC) [20] is a model-based defense method that attempts to check whether the model contains backdoors. It is assumed that for a backdoored model, to modify the inputs to be misclassified to the target label of the backdoor attack is much easier than other non-target labels. Therefore, NC attempts to recover a trigger for each label, and check whether the smallest trigger is significantly smaller than others. If so, the model is deemed as having been backdoored, and this trigger is considered as the actual trigger. NC is effective in detecting backdoor attacks with small-sized triggers, e.g., 18% of the image for MNIST, but may fail for those with larger triggers.

3) Strip: Strip [14] is a data-based defense method that examines whether the incoming inputs contain triggers. An input image sample is duplicated for multiple times, and each replica is merged with a different image sample to form a perturbed sample. The prediction results of these perturbed samples are expected to have a high entropy value if the model is benign due to randomness, and a low entropy value if the model is malicious since the input with the trigger is strongly associated with the target misclassification label. The key is to determine the entropy threshold to differentiate benign inputs and inputs with triggers.

4) ABS: Artificial Brain Stimulation (ABS) [13] is a most recent model-based defense strategy that scans deep neural networks to determine whether there are backdoors. It is inspired by Electrical Brain Stimulation (EBS) that is used to analyze human brain neurons. Given the deep neural network, ABS changes the activation of a neuron and observes the corresponding output differences. When appropriate stimuli are provided, neurons affected by the backdoor will manifest themselves by significantly increasing the activation of the target label and possibly inhibiting the activation of other labels. ABS utilizes stimulus analysis to reverse engineer backdoor triggers. ABS is effective in detecting single-trigger attacks but not multi-trigger attacks.

With our target of a robust backdoor attack for malicious cloud service providers, we demonstrate that our proposed RobNet is resistant to all these state-of-the-art defense strategies via intensive experiments in Section V.

III. Threat Model

In the outsourced cloud environment, there are two parties, namely, a user who outsources the training of a deep neural network and a cloud service provider who trains the deep neural network. More specifically, the user negotiates with the cloud service provider on the specifics of the neural network, including the depth (the number of layers) of the DNN, the size of each layer, and the activation function. The user may upload a training dataset, or entrust the cloud service provider to collect the training dataset. The cloud provider trains and returns a deep neural network to the user. After receiving the model from the cloud, the user tests its accuracy on a

TABLE I
APPLICATION SCENARIOS OF DIFFERENT ATTACKS

Attack type	# of triggers	# of target labels	Facial recognition attack scenario
Single-trigger single-location	single	single	Any user wearing special glasses (trigger) is misclassified as Alice.
Single-trigger multi-location	single	single	Special glasses can be worn at different places to be misclassified to Alice.
Multi-trigger same-label	multiple	single	Any user wearing special glasses or a special hat is misclassified to Alice.
Multi-trigger multi-label	multiple	multiple	Any user wearing special glasses is misclassified to Alice. Any user wearing a special hat is misclassified to Bob.

validation dataset. The user accepts the model only if the test accuracy meets a target threshold determined by domain knowledge or performance requirements.

We consider the threat model in which the cloud provider is the adversary who aims to return a backdoored but well-performed deep neural network. Unlike the prior work [6] that assumes the user only tests the prediction accuracy of the received model, we consider a much stronger case where the user leverages various advanced defense strategies to check whether the model is backdoored or not.

IV. RobNet: Construction Details

A. Overview

Our proposed backdoor attack RobNet consists of two key steps: trigger generation and backdoor injection.

1) Trigger Generation: The trigger plays an important role in backdoor attacks. There are two kinds of triggers, namely, random triggers and model-dependent triggers. Since random triggers yield a low attack success rate and are easy to be detected by defense strategies, we adopt model-dependent triggers. Model-dependent triggers are generated based on the deep neural network. We develop a novel model-dependent trigger generation algorithm, which not only enhances the attack capabilities but also evades most defense strategies.

The main idea of our trigger generation algorithm is to decide value assignment in an empty mask (the mask covers the location of the trigger on the image) so that certain neuron(s) is excited the most. As shown in Fig. 1, the mask is rectangular, and the selected neuron is highlighted in yellow or blue. The final value assignment of the mask yields the model-dependent trigger. In single trigger attacks, the trigger raises the activation value of the selected neuron from 2 to 12. The key of trigger generation lies in selecting appropriate neurons that can best facilitate backdoor attacks. Since targeted backdoor attacks aim to misclassify malicious inputs into the target label, we seek for neurons that are closely associated with the target label. More specifically, we choose the neuron(s) with the highest weights and activations when the network is fed (clean) inputs that belong to the target label. We show in Section V that our selected neuron(s) will not be removed by network pruning operations, thus the proposed attacks are robust to pruning-based defense strategies.

2) Backdoor Injection: To inject the backdoor into the DNN, we poison the training dataset (uploaded by the user) with malicious samples to retrain the DNN model. For a given clean sample $(\mathbf{x}, c)$, where c is the ground-truth label, we construct the malicious sample(s) $(\mathbf{x}^*, c^*)$, where $\mathbf{x}^*$ is the clean sample with the trigger and c^* is the target label. Multiple malicious samples can be constructed since the trigger can be placed at different locations of the image.

Most existing works assumed that the location of the trigger is fixed and did not consider the influence of trigger location on the performance of backdoor attacks. Nonetheless, we find that the attack success rate is highly sensitive to the trigger location. If the trigger location during testing is slightly different from that during training, the attack success rate will drop sharply. To make our attacks more robust, we place the same trigger at different locations of the clean sample to form multiple malicious samples to poison the training dataset and retrain the benign model to inject the backdoor. In this way, the attacks are more robust to trigger locations.

In the retraining procedure, both the clean training samples and corresponding malicious samples are utilized to retrain the deep neural network, aiming to strengthen the connection between the selected neuron and the targeted label. As shown in single trigger backdoor attacks in Fig. 1, after retraining, the weight between the selected neuron and the output neuron (label 1, targeted misclassification label) rises from 0.1 to 0.7. Note that we only retrain the layers between the layer of the selected neuron in trigger generation and the output layer, in the aim of preserving high prediction accuracy for clean samples. After retraining, the backdoored model will output the target label once the trigger appears in the input but will behave normally in the absence of the trigger.

Apart from single-trigger backdoor attacks, we also develop and evaluate multi-trigger attacks where the adversary generates multiple triggers that target at the same or different labels. Note that in multi-trigger attacks aiming for a single label, the triggers generated at different patching locations are different, while in single-trigger attacks, we patch the same trigger generated at a specific location to different locations to strengthen the robustness of the attack. As shown in Fig. 1, trigger a and b are generated based on the same neuron (highlighted in blue) and target the same label 0 but are placed at different locations. Trigger c is generated based on the yellow neuron targeting label 1. Note that trigger c may have the same location as a or b but target different labels. Multi-trigger attacks extend the attack space of backdoor attacks by enabling the adversary to achieve various attack targets with a single backdoored model. For example, a backdoored traffic sign recognition model may misclassify all inputs with trigger a as a *stop* sign and all inputs with trigger b as a *right-turn-only* sign.

Taking the facial recognition task as an example, we illustrate the attack scenarios to which different attacks are applicable in Table I.

Algorithm 1 Backdoor Trigger Generation Algorithm

```
Require: Pretrained deep neural network $F$, the mask $M$, the threshold $h$, the maximum number of iterations $I$, and the learning rate $lr$.
Ensure: The trigger $Trigger$.
1: $l = layer\_selection(F)$.
2: $Trigger = Mask\_Initialize(M)$.
3: $n^* = neuron\_selection(F, l)$, $i = 0$.
4: $c^*_{n,i} = neuron\_activation(F, l, n^*)$.
5: $t^*_{n,i} = max\_value(F, l)$.
6: $cost_i = |c^*_{n,i} - t^*_{n,i}|^2$
7: $dif_i = c^*_{n,i}$.
8: $flag = \max(cost_i, dif_i)$.
9: while $flag > h$ and $i < I$ do
10:    $\Delta = \partial cost / \partial Trigger$
11:    $\Delta = \Delta \circ M$
12:    $Trigger = Trigger - lr \cdot \Delta$.
13:    $i = i + 1$.
14:    $cost_i = |c^*_{n,i} - t^*_{n,i}|^2$
15:    $dif_i = c^*_{n,i} - c^*_{n,i-1}$.
16:    $flag = \max(cost_i, dif_i)$.
17: end while
18: return $Trigger$.
```

B. Step I: Trigger Generation

The most critical step in backdoor attacks is to generate an effective trigger. Our proposed model-dependent trigger generation approach is summarized in Algorithm 1. The trigger generation process is equivalent to finding the value assignment of an empty mask to strongly activate certain neuron(s) in the neural network. Therefore, we need to first determine what the mask is and which neurons to activate, then produce the trigger accordingly.

1) Mask Determination: The mask $\mathbf{M}$ is a matrix in which the number of rows and the number of columns conform to the height and the width of the image sample. The elements in the matrix have a value of 1 in the trigger region and 0 in the remaining region. The trigger region in the mask is formed by determining the shape, size, and position of the trigger. For the shape and the position of the trigger, we follow existing works on backdoor attacks [10]. The shape of the trigger is rectangular, and the position of the trigger is at the lower right corner of the image sample. To choose the appropriate size of the trigger is a trade-off between the attack success rate and attack stealthiness. A larger trigger boosts the attack success rate but is more observable and easier to be detected. Through our extensive experiment, we find that a trigger size of 7% of the image brings an ideal performance. Note that 7% falls into the range that can be detected by NeuralCleanse [20], yet we show in Section V that NeuralCleanse is ineffective in defending against RobNet.

2) Neuron Determination: Following [10], we select a single neuron to activate. To find the most effective neuron, we first decide which layer the neuron should reside, then pinpoint the neuron in the selected layer.

Layer Selection: Convolutional layers are not selected, since a neuron in a convolutional layer only connects to a small set of neurons of the preceding layer, thus the response to the input trigger is weak. For similar reasons, we do not select pooling layers. Fully-connected layers are ideal since each neuron connects all neurons from the previous layer to all neurons of the next layer, exerting high influence on the output.

Since the benign deep neural network will be retrained between the selected layer and the output layer for backdoor injection, we do not select layers close to the output layer due to a lack of room for retraining. Taking all these factors into consideration, we choose the first fully-connected layer of the DNN for neuron selection.

Neuron Selection: To achieve the best attack performance, the selected neuron should be sensitive to malicious inputs and has a strong relationship with the target misclassification label. Given the selected layer l, let $\mathcal{N}$ represent the set of neurons in layer l, $\mathcal{J}$ represent the set of neurons in the preceding layer $l-1$, and $w^{l-1,l}_{n,j}$ represent the weight between neuron $n \in \mathcal{N}$ and neuron $j \in \mathcal{J}$ across the two layers. The existing method [10] selects the neuron n^* that is most affected by the preceding layer (i.e., with the largest sum of weights to the preceding layer).

$$n^* = \arg\max_{n\in\mathcal{N}} \sum_{j\in\mathcal{J}} w^{l-1,l}_{n,j}. \tag{2}$$

However, the selected neuron in (2) is indiscriminately sensitive to both malicious inputs and benign inputs, ignoring the impact of the selected neuron on the target misclassification label. Moreover, the neuron selection method of [10] did not consider activation-based pruning defense strategies. The selected neuron may have a low activation and will be pruned, leading to the failure of backdoor attacks.

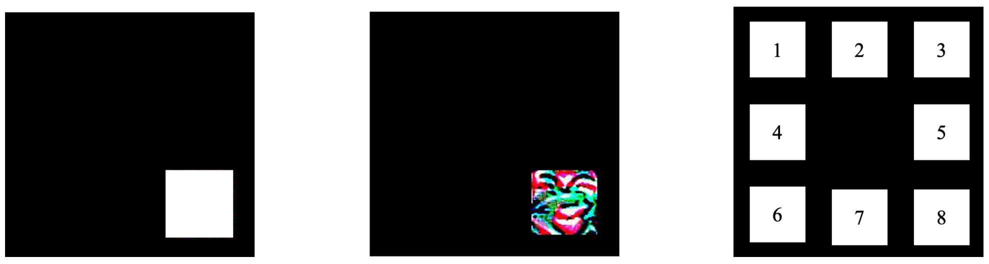
After carefully studying network pruning techniques, we propose a robust neuron selection mechanism. It is shown that certain neurons are more active when inputting specific classes of inputs [26], [27]. Our idea is to find the neuron with both large weights and a large number of activations when the neural network is fed with clean inputs of the target misclassification label. In this way, the chosen neuron has a strong association with the target misclassification label, and is less likely to be pruned due to its large weights.

To achieve this goal, we input massive clean samples of the target misclassification label into the benign DNN model and record the number of activations and the weights of each neuron in the selected layer. The neuron with the highest weighted sum of the number of activations and weights is selected.

$$n^* = \arg\max_{n\in\mathcal{N}}(\lambda \sum_{\mathbf{x}\in\mathcal{X}^c} \mathbf{I}_{[F(x)=n]} + (1-\lambda)\sum_{j\in\mathcal{J}} w^{l-1,l}_{n,j}), \tag{3}$$

where $\mathcal{X}^c$ is the set of benign samples of the target misclassification label c, and $\mathbf{I}_{[F(\mathbf{x})=n]}$ denotes that neuron n is activated by input $\mathbf{x}$.

λ balances the weights and the number of activations. The activations reflect the connection between the neuron and the inputs of the target label, and the weights characterize the influence of the neuron on the neurons of the next layer. With



(a) Empty mask (b) Trigger (c) Trigger locations

Fig. 2. Illustration of the backdoor attack mask, trigger, and location. There are eight different trigger positions for generating the multi-trigger.

a larger λ, we prefer a neuron that is more activated by the inputs of the target label. With a smaller λ, we prefer a neuron that is more influential on the following neurons. According to empirical studies, we set λ as 0.65 to balance the two aspects. A well-balanced λ will lead to a high attack success rate (ASR), while an ill-balanced one will result in low ASR. The prediction accuracy (PA) of clean inputs is less affected by λ since λ is mostly related to the trigger and the poisoned inputs.

3) Trigger Formation: Given the mask and the selected neuron, we adopt gradient descent to iteratively update the value assignment of the mask to minimize the cost function $|v_{n,t}-u_t|^2$, where $v_{n,t}$ is the current activation value of neuron n in the t-th iteration, and u_t is the target value. Given that layer $l-1$ consists of K neurons, the activation a_n^l of the n-th neuron in layer l is

$$a_n^l = \Phi^l\Big(\sum_{j=1}^{K} w_{n,j}^{l-1,l} \cdot a_j^{l-1} + b_n^l\Big), \tag{4}$$

where Φ^l is the activation function of layer l, b_n^l is the bias.

The target activation value u_t is set as the highest activation value of the neurons in the selected layer. If the value of the selected neuron is already the highest in the chosen layer, we continue to increase its value until convergence or until the number of iterations hits the limit. In the last iteration, the trigger is extracted from the mask area.

C. Step II: Backdoor Injection

The process of backdoor injection consists of two steps: data poisoning and model retraining.

1) Data Poisoning: Given the generated trigger mask $\mathbf{M}$, the adversary patches the trigger to clean data samples to construct poisoned data samples.

$$\mathbf{x}^* = \mathbf{x} + \mathbf{trigger} \odot \mathbf{M}. \tag{5}$$

where $\mathbf{x}$ is a clean sample, $\mathbf{x}^*$ is the corresponding poisoned malicious sample, $\mathbf{M}$ is the mask to locate the trigger, and $\odot$ is element-wise product. The label of the poisoned data samples is the target misclassification label.

Existing methods patch the trigger to clean samples to generate poisoned retraining dataset at exactly the same location as in the trigger generation process. Nevertheless, the attack success rate of the backdoored model will sharply decrease if the location of the trigger in the test samples is slightly different from that in the poisoned retraining dataset. Furthermore, many state-of-the-art defense strategies rely on the assumption that the position of the trigger is fixed (e.g., NeuralCleanse). To increase the robustness of RobNet, we propose a multi-location patching method by patching the trigger to different locations of the clean samples to generate multiple poisoned data samples. For example, the same generated trigger is patched to both location 6 and location 8 of a clean sample in Fig. 2 to construct two poisoned samples with the same target class. These two types of poisoned samples are both added to the poisoned dataset.

2) Model Retraining: In the outsourced cloud environment, the malicious cloud first trains a benign neural network using a clean training dataset. Then, the cloud performs trigger generation algorithm based on the benign neural network. Finally, the cloud constructs poisoned training dataset using the generated trigger, and retrains the benign neural network with the combined clean and poisoned datasets to obtain the backdoored model. In this way, the backdoored model maintains a high prediction accuracy for clean inputs, but misclassifies malicious inputs to the target label.

It is worth noting that only the layers between the selected layer in trigger generation and the output layer are retrained. Besides, not all clean samples are poisoned to avoid high retraining overhead. We only poison a fraction of the clean training dataset. In Section V, we evaluate the influence of the poison ratio on the performance of RobNet.

D. Multi-Trigger Attacks

Most existing backdoor attacks only consider a single trigger that leads to misclassification to a single target label. Multiple triggers can enrich the attack effect of malicious inputs without affecting the prediction accuracy of clean inputs (the clean inputs do not contain the triggers). Fig 2 shows typical locations that we use in our experiments, but the possible locations are not limited to those in the figure. We consider two scenarios, i.e., multi-trigger attacks targeting a single or multiple misclassification classes.

For multiple triggers that target at the same class label, we generate different triggers at different locations. Each trigger is generated in the same way as single-trigger attacks. To construct poisoned data samples, we patch each trigger to the corresponding location to build multiple poisoned data samples. For instance, we generate trigger $\mathbf{A}$ with mask $\mathbf{M_A}$ at location 6 (as in Fig. 2) and trigger $\mathbf{B}$ with mask $\mathbf{M_B}$ at location 8, both targeting at the same misclassification label c. We can construct two poisoned samples $(\mathbf{x}+\mathbf{A}\odot\mathbf{M}_\mathbf{A}, c)$ and $(\mathbf{x}+\mathbf{B}\odot\mathbf{M}_\mathbf{B}, c)$ for model retraining. During attacks, multiple triggers can be present in test samples, i.e., $\mathbf{x}+\mathbf{A}\odot\mathbf{M}_\mathbf{A}$, $\mathbf{x}+\mathbf{B}\odot\mathbf{M}_\mathbf{B}$ and $\mathbf{x}+\mathbf{A}\odot\mathbf{M}_\mathbf{A}+\mathbf{B}\odot\mathbf{M}_\mathbf{B}$ will all be misclassified as c. Note that different from multi-location patching in single-trigger attacks that patch the same generated trigger to different locations of the image for data poisoning, multi-trigger attacks (targeting one misclassification label) patch different generated triggers to different locations of the image for data poisoning. Both multi-trigger attacks and multi-location patching make the attacks more robust to the location of the trigger in test samples.

For multiple triggers that target at different class labels, we generate different triggers at the same or different locations. Each trigger targets a different misclassification label, and is generated in the same way as single-trigger attacks. To construct poisoned data samples, we patch each trigger to the corresponding location to build multiple poisoned data samples. For instance, we generate trigger $\mathbf{A}$ with mask $\mathbf{M}$ at location 6 targeting label c_1, trigger $\mathbf{B}$ with the same mask $\mathbf{M}$ at location 6 targeting label c_2, trigger $\mathbf{C}$ with mask $\mathbf{M}'$ at location 8 targeting label c_3. We can construct poisoned samples $(\mathbf{x}+\mathbf{A}\odot\mathbf{M}, c_1)$, $(\mathbf{x}+\mathbf{B}\odot\mathbf{M}, c_2)$, and $(\mathbf{x}+\mathbf{C}\odot\mathbf{M}', c_3)$ for model retraining. During attacks, only one trigger will be present in test samples, i.e., $\mathbf{x}+\mathbf{A}\odot\mathbf{M}$ will be misclassified as c_1, $\mathbf{x}+\mathbf{B}\odot\mathbf{M}$ will be misclassified as c_2 and $\mathbf{x}+\mathbf{C}\odot\mathbf{M}'$ will be misclassified as c_3. The adversary will not use samples such as $\mathbf{x}+\mathbf{A}\odot\mathbf{M}+\mathbf{C}\odot\mathbf{M}'$ for attacks.

Note that the selected neuron is related to the target misclassification label but not the trigger location, and the generated trigger is related to the selected neuron and the trigger location. In multi-trigger same-label attacks, the same neuron is selected to generate different triggers at different locations. In multi-trigger multi-label attacks, it is possible that the same neuron may be selected for different target labels (the neuron is strongly activated by both labels). In this case, the same trigger will be generated for different target labels at the same trigger location. A simple way to solve this problem is to choose different trigger locations for different target labels. Another way is to refine the neuron selection process. For instance, we target label c_A and label c_B, and neuron A has the highest weighted sum of weights and the number of activations for both labels. We may select neuron A for label c_A and select the second-best neuron for label c_B. In the experiments, we demonstrate the effectiveness of multi-trigger attacks.

V. Implementation and Evaluation

A. Experiment Setup

We consider two state-of-the-art backdoor attacks BadNets [6] and Hidden-Backdoor (HB) [19] as the baselines, which have the same threat model as ours. BadNets uses random triggers, and HB uses invisible random triggers. Although BadNets can achieve both high prediction accuracy and attack success rate, our experiments show that BadNets can be defended by all four defense strategies [13], [14], [16], [20] that we evaluate. The invisibility concern of HB affects its performance in attack success rate, and we show that HB is detected by three of the four defense strategies that we evaluate even if the trigger is imperceptible by sight. Note that we did not experiment on [10] that has model-dependent triggers as ours, since the threat model of [10] is different from ours.

All experiments are carried out on an Ubuntu 16.04 system with Intel CPU of 4 cores and 4 GeForce RTX 2080 Ti GPU. We consider 3 most commonly-used datasets.

1) MNIST: MNIST [28] consists of 70,000 gray-scale images that belong to 10 classes (classes $0 \sim 9$). According to existing works, we select 60,000 samples as the training dataset, and the remaining 10,000 image samples as the test dataset. We assume that the user specifies the structure of the trained deep neural network as LeNet-5 [21], i.e., two convolutional layers with pooling and two fully-connected layers. In the training process, the base learning rate is set as 0.01 with the inverse decay policy. The max iteration is set as 10,000. The momentum of stochastic gradient descent (SGD) is set as 0.9 and the benign model achieves an accuracy of 99.12%.

2) GTSRB: GTSRB [29] consists of 39,209 training samples and 12,630 testing samples. The image samples vary in size from 28×28 to 215×215. We utilize the annotation information that describes the number and the type of traffic signs in an image to crop and enlarge the area that contains the traffic sign(s). In this way, we obtain all the samples with a uniform size of 224×224. We train a traffic sign classifier on the VGG-16 structure with pretrained weights of ImageNet [22], [23]. We train the model using SGD with momentum optimizer for 5,000 epochs. The base learning rate is 0.0001, and polynomial decay is adopted with a power of 1.0. The trained benign VGG model achieves a prediction accuracy of 94.98% on the test dataset.

3) CIFAR-10: CIFAR-10 [30] contains 60,000 color images that belong to 10 classes. Each class includes 6,000 images, and each image has a size of 32×32. We use 50,000 image samples for training and the remaining 10,000 samples for testing. We train a CIFAR-10 classifier on the VGG-16 structure with batch normalization. To achieve a better performance, we perform data augmentation by randomly rotating, shifting, and horizontally flipping the training data samples. We use the SGD optimizer, and the base learning rate is 0.1 with a step drop every 50 iterations. We train the model for 200 epochs with a batch size of 128. The obtained benign model has a prediction accuracy of 96.05% on the test set.

We employ *prediction accuracy* (PA) and *attack success rate* (ASR) to evaluate the effectiveness of RobNet. Prediction accuracy measures the ratio of correctly-labeled clean samples in the test dataset. Mathematically, *prediction accuracy* is defined as

$$PA(F^*, T) = \frac{1}{|T|}\sum_{\mathbf{x}\in T}\mathbf{I}[F^*(\mathbf{x}) = c], \tag{6}$$

where T is the set of clean inputs, F^* is the backdoored deep neural network, c is the corresponding ground-truth label of $\mathbf{x}$.

TABLE II
THE IMPACT OF POISON RATIO ON THE ATTACK PERFORMANCE

Poison ratio	MNIST		GTSRB		CIFAR-10	
	ASR	PA	ASR	PA	ASR	PA
0.1%	93.23%	98.79%	99.4%	95.3%	88.17%	97.29%
0.5%	93.66%	98.75%	99.4%	95.4%	89.48%	97.27%
1%	99.21%	98.73%	99.5%	95.41%	92.10%	97.25%
5%	99.36%	98.60%	99.7%	95.36%	92.53%	97.22%
10%	99.55%	98.57%	99.83%	95.28%	96.73%	94.32%
15%	99.44%	98.63%	99.9%	95.30%	98.28%	95.09%
20%	99.59%	98.68%	99.95%	95.25%	99.26%	96.73%
25%	100%	98.45%	100%	94.60%	99.90%	91.44%
30%	99.97%	98.2%	99.9%	94.10%	99.87%	89.39%

TABLE III
SINGLE-ATTACK PERFORMANCE COMPARISON

	MNIST		GTSRB		CIFAR-10	
	ASR	PA	ASR	PA	ASR	PA
[6]	99.07%	98.06%	89.8%	90.6%	88%	93.45%
[19]	72.9%	94.8%	75.44%	80.1%	27.1%	84.5%
Ours	**99.59%**	**98.68%**	**99.95%**	**95.25%**	**99.26%**	**96.73%**

Attack success rate is calculated as the percentage of malicious samples that are misclassified by the backdoored deep neural network into the target label. Mathematically, the *attack success rate* is defined as

$$ASR(F^*, T^*) = \frac{1}{|T^*|} \sum_{\mathbf{x}^* \in T^*} \mathbf{I}[F^*(\mathbf{x}^*) = c^*], \tag{7}$$

where T^* denotes the set of malicious inputs, c^* is the target label.

B. Evaluation Results

1) Effectiveness of Backdoor Attacks: We first validate the effectiveness of ROBNET in terms of single-trigger attacks and multi-trigger attacks, then evaluate the impact of the number of triggers on the performance of ROBNET.

Single-Trigger Attacks: Before comparing ROBNET with other state-of-the-art attacks, we first investigate the impact of poison ratio on the attack performance. Poison ratio refers to the percentage of benign samples that are poisoned in the training dataset. As shown in Table II, as the poisoning rate increases, the attack success rate will increase, but the prediction accuracy will slightly decrease. Even with 1% poison ratio, ROBNET can achieve a mean attack success rate of 97% while maintaining a high prediction accuracy.

Table III summarizes the performance of single-trigger backdoor attacks. With the same settings (e.g., poison ratio (20%), retraining operations), ROBNET outperforms both BadNets and HB with a large margin. Note that we set the poison ratio as 20% since HB [19] is not effective at a lower poison ratio. The high attack success rate can be attributed to our proposed trigger generation algorithm, which takes into account the association of the neuron with the target label. In comparison, both baselines choose a random backdoor trigger, which cannot strongly activate the inner neurons (especially the neurons associated with the targeted label).

Fig. 3 demonstrates the performance of single-trigger attacks when different patching locations are considered. The

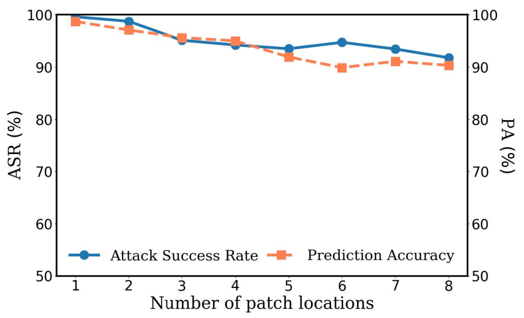


Fig. 3. The robustness of the attacks under different possible patching locations of the trigger.

experiments are conducted using the MNIST dataset. As we have mentioned before, during the attack, patching the trigger to the test sample at exactly the same location as that of the training sample is difficult, and a slight difference may lead to large performance degradation. Therefore, ROBNET allows the adversary to patch the triggers to different locations of the image samples, which increases the robustness of the attacks. Note that we achieve this by constructing poisoned samples with the same trigger placed at different locations of the clean samples and retrain the neural network. Naturally, with more possible patching locations, both attack success rate and prediction accuracy decrease. When only one location is considered, the attack success rate can reach 99.59% and the prediction accuracy can reach 98.68%. When there are eight possible locations, the attack success rate drops to 91.75% and the prediction accuracy is 90.29%, which are still very high. This confirms that ROBNET is robust to the patching location of the trigger during attacks.

Multi-Trigger Attacks: We extend the single-trigger attacks to multi-trigger attacks with two settings, i.e., single-label and multi-label. We conduct experiments on both scenarios and show the results in Fig. 4. The number of triggers is set as two. It is shown that both scenarios achieve a high attack success rate and a high prediction accuracy. In general, the same-label scenario has higher attack success rate and prediction accuracy than the multi-label scenario, since the multi-label scenario is more complicated and involves more neurons in the neural network.

The number of triggers in multi-trigger attacks has a crucial influence on the performance. Table IV displays the attack success rate and prediction accuracy when there are different numbers of generated triggers. We conduct the experiments in MNIST. It is shown that in both scenarios, the attack success rate and prediction accuracy decrease with more backdoor triggers. In the same-label scenario, having more triggers makes the attack more robust to different possible trigger locations in test samples, but introduces more changes during retraining and reduces the prediction accuracy. In the multi-label scenario, multiple triggers target at different labels, leading to more complicated neuron selection and retraining processes, which affects the attack performance.

Overall, the attack success rate and the prediction accuracy of multi-trigger attacks are over 90%, which validates the effectiveness of ROBNET.

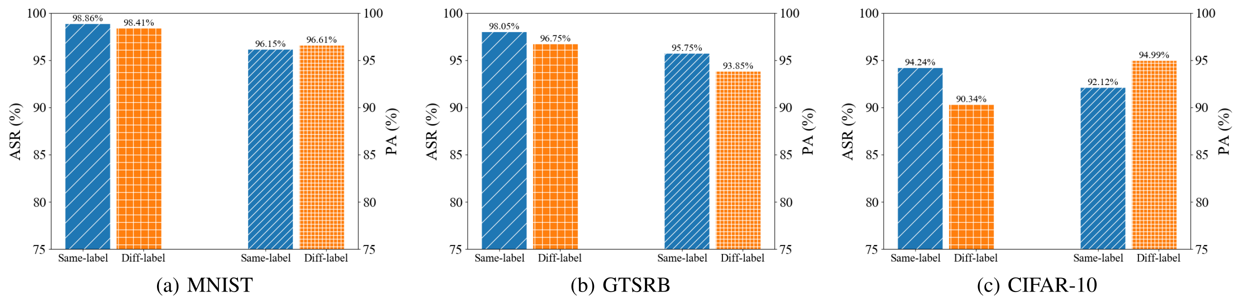


Fig. 4. Multi-trigger attack performance.

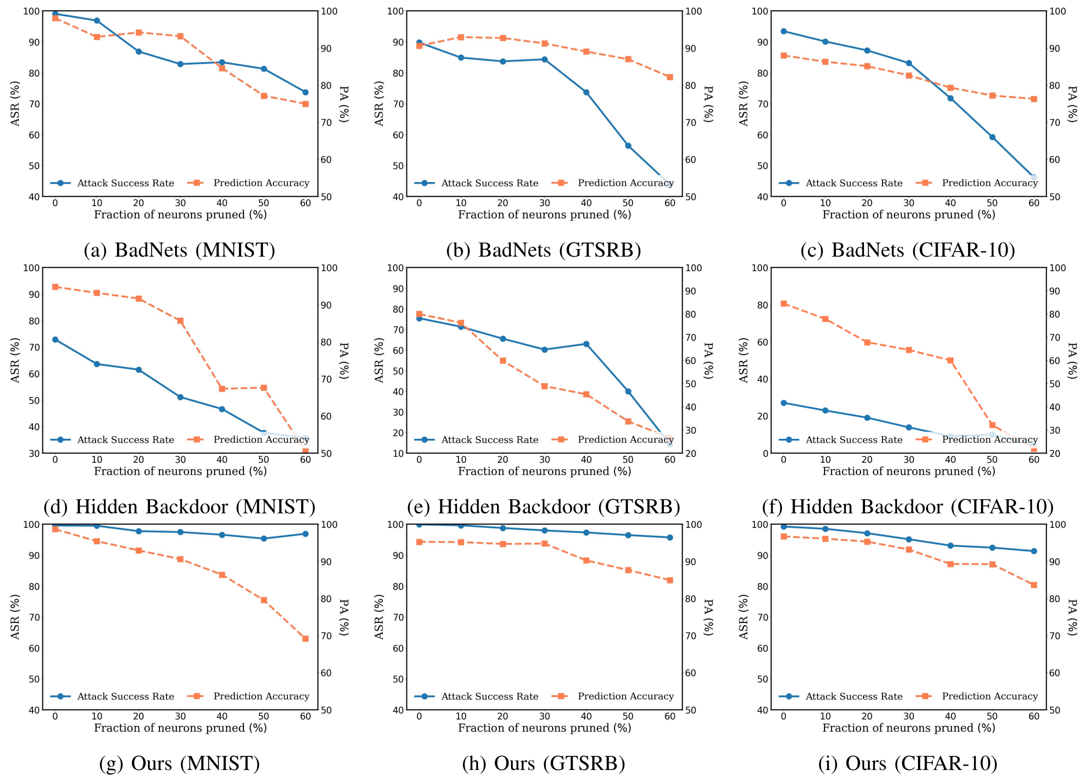


Fig. 5. Comparison in prune.

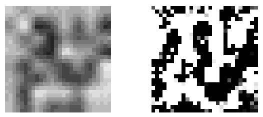
TABLE IV
THE IMPACT OF THE NUMBER OF TRIGGERS ON MULTI-TRIGGER ATTACKS

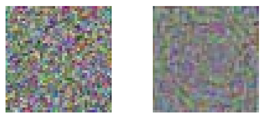
Number of triggers	Same-label		Diff-label	
	ASR	PA	ASR	PA
2	98.86%	96.15%	98.41%	96.61%
3	96.62%	95.38%	95.48%	95.82%
4	95.62%	95.32%	96.84%	95.43%
5	95.74%	93.27%	94.10%	95.39%
6	95.11%	91.17%	91.74%	94.06%
7	94.68%	88.45%	91.30%	93.44%
8	95.03%	87.97%	90.21%	92.68%

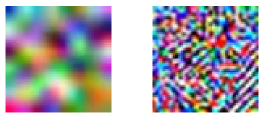
2) Robustness to State-of-the-Art Defense Strategies: We have reproduced four state-of-the-art defense strategies against ROBNET and the baselines BadNets and HB.

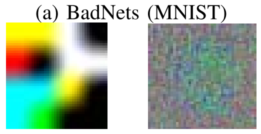
Pruning: We apply pruning to the backdoored models of ROBNET, BadNets and HB on all three datasets under the single-trigger multi-location scenario. Note that the pruning approach in [16] removes neurons with low activations, but ROBNET can also resist weight-based pruning methods. The results are shown in Fig. 5. BadNets and HB show an obvious decline in both attack success rate and prediction accuracy as a larger percentage of neurons are removed. This shows that network pruning is effective in defending against BadNets and HB. In contrast, the attack success rate of ROBNET remains steady as more neurons are pruned. This is due to the fact that we select the neuron that has a high activation when inputting massive clean samples. Thus the neuron is more likely to be preserved during pruning.

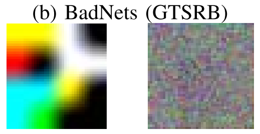
NeuralCleanse: We apply NC to the backdoored models of ROBNET and the baselines on all three datasets. In particular, ROBNET adopts the multi-trigger same-label model with two different triggers targeting the same label at location 1 and 8 of Fig. 2. NC generates a potential trigger for a label by

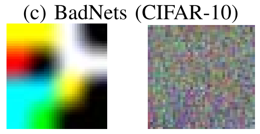

(a) BadNets (MNIST)

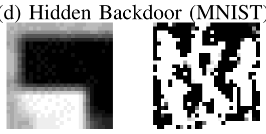

(b) BadNets (GTSRB)

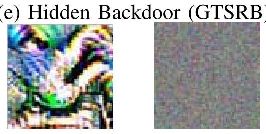

(c) BadNets (CIFAR-10)


(d) Hidden Backdoor (MNIST)


(e) Hidden Backdoor (GTSRB)


(f) Hidden Backdoor (CIFAR-10)


(g) Ours (MNIST)


(h) Ours (GTSRB)

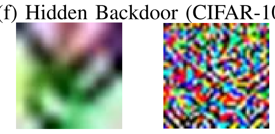

(i) Ours (CIFAR-10)

Fig. 6. Comparison in NC. In each group, the left one is the original trigger and the right one is the inversed trigger.

maximizing the probability of misclassifying inputs with the trigger as the target label. The optimization result is a mask (indicating the location of the trigger) and a corresponding pattern. Since the location of our trigger is not fixed, the location of the generated mask of NC does not yield much information. Therefore, we focus on comparing the difference between the pattern of the trigger generated by NC and the actual trigger of backdoor attacks. As shown in Fig. 6, regarding the backdoored models of RobNet and HB, the trigger pattern generated by NC is quite different from any of the original trigger. On the contrary, the recovered trigger by NC is similar to the actual trigger of BadNets. For the MAD (Median Absolute Deviation) anomaly detection of NC, we set the threshold as 1.4, meaning that any label with a MAD larger than 1.4 is regarded as the target label. Experiments show that BadNets can be detected successfully in all cases, while HB and RobNet can evade the detection. This demonstrates that NC is effective in detecting BadNets but not RobNet that leverages multi-trigger same-label method and HB. The reason why HB can evade NC is that its poisoned samples (used to train DNN) are similar to benign samples (the trigger is hidden). Thus, given the backdoored model, NC fails to reverse a trigger, and the outlier detection based on MAD is also ineffective. Similarly, NC is also ineffective against our multi-trigger multi-label attacks.

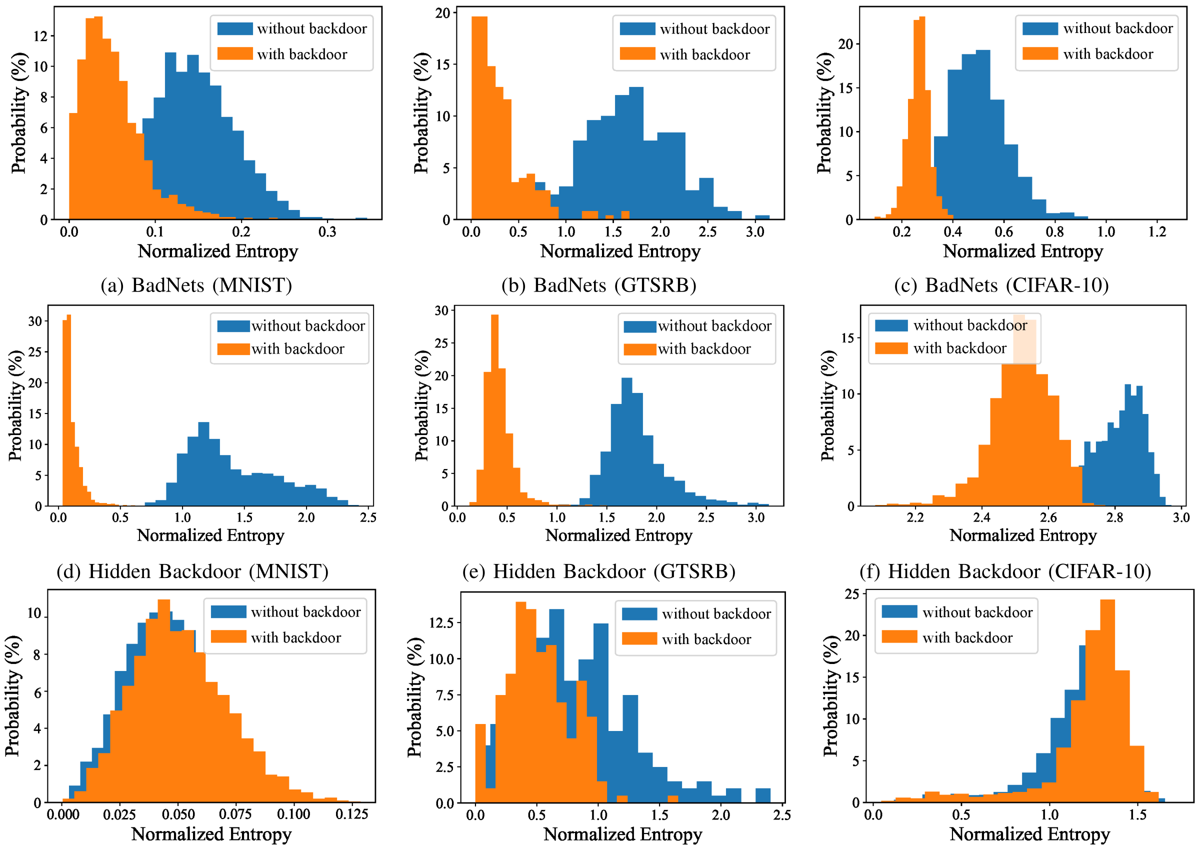
Strip: We apply Strip to the backdoored models of RobNet and the baselines on all three datasets. In particular, RobNet adopts single-trigger multi-location attacks with the same trigger attached at both location 1 and 8 as in Fig. 2. We randomly select 2,000 clean image samples and 2,000 inputs with triggers. Each input is superimposed with 200 randomly-selected images. The entropy distribution of the classification results of the clean inputs and the malicious inputs is shown in Fig. 7. It is obvious that in BadNets and HB, the entropy distributions of malicious inputs and clean inputs are well separated, which enables the user to detect malicious inputs with a probability of more than 90%. In comparison, in RobNet, the entropy distributions of clean inputs and malicious inputs are very similar. The overlap is substantial, thus it is hard to differentiate malicious inputs and clean inputs. Clean samples usually have a high entropy distribution due to randomness. BadNets and HB add the same trigger at a fixed location, creating a unique pattern with a low entropy. In this way, the entropy of malicious inputs with the trigger is much lower than that of clean inputs, thus easily detected by Strip. In comparison, our proposed single-trigger multi-location attacks patch the same trigger at different locations, which leads to more randomness and results in high entropy. Therefore, Strip cannot differentiate clean samples and malicious samples of RobNet according to the entropy distribution.

ABS: We apply ABS to the backdoored model of RobNet and the baselines on CIFAR-10 since the authors only published their detection APIs on CIFAR-10 DNN models. In particular, RobNet adopts the multi-trigger multi-label scheme with two different locations at 1 and 8 as in Fig. 2. ABS uses the abnormal stimulus of neurons to reverse the trigger to target a certain REASR (attack success rate of reverse engineered trojan triggers). We found that ABS detected 21, 11, and 4 suspicious neurons in BadNets, HB, and RobNet, respectively. When we set REASR to at least 90%, ABS detected 4, 2, and 0 suspicious neurons in BadNets, HB, and RobNet, respectively. The possible reason that RobNet evades ABS may be that we undermine the assumption of ABS that the target misclassification label is activated by only one neuron (we activate multiple inner neurons with different triggers).

VI. Related Work

A. Backdoor Attacks

Backdoor attacks aim to manipulate DNN models so that the backdoored models behave normally on clean inputs but misclassify malicious inputs with the special trigger. Backdoor attacks are usually performed under two cases, i.e., the adversary is a malicious model vendor or a malicious training data vendor. In the former case, the user outsources the training process to cloud service providers (e.g., AWS, Google) [6], [31], [32] or directly reuse the trained model from the model sharing platform (e.g., Caffe model zoo, and ONNX zoo) [10], [11], [18], [24], [33]. In the latter case, a user with insufficient training data downloads image samples from the internet as training data samples [17], [19], and the image samples are poisoned with the backdoor trigger.



(a) BadNets (MNIST) (b) BadNets (GTSRB) (c) BadNets (CIFAR-10)

(d) Hidden Backdoor (MNIST) (e) Hidden Backdoor (GTSRB) (f) Hidden Backdoor (CIFAR-10)

(g) Ours (MNIST) (h) Ours (GTSRB) (i) Ours (CIFAR-10)

Fig. 7. Comparison in strip.

To recap, Gu *et al.* proposed the first backdoor attacks against deep neural networks, namely BadNets [6]. BadNets deals with the machine learning outsource scenario, where the malicious cloud service provider injects the backdoor by poisoning the training dataset with a random trigger. Liu *et al.* [10] considered a different scenario where the adversary downloads a trained model from the internet and reverses the training data samples. A trigger is then generated by selecting a neuron with large weights, however, the association between the neuron and the target label is not considered. The adversary then publishes the backdoored model for users to download. Ji *et al.* discussed multi-trigger attacks [11], [18] and gave mathematical formulation, but did not verify the performance via experiments. Yao *et al.* [33] proposed backdoor attacks for transfer learning, in which the adversary injects latent backdoors to the "teacher" model, and the "student" model inherits the malicious characteristics. Saha *et al.* [19] proposed a *hidden* random backdoor trigger injection method, but it needs a large poison ratio and a large trigger size to achieve an acceptable attack success rate. More recently, Lin *et al.* [31] proposed a novel composite backdoor attack that uses the composition of benign features as the backdoor trigger. Pang *et al.* [34] discovered that it is possible to optimize the current backdoor attacks (e.g., [10]) in terms of both visibility (e.g., trigger transparency and size) and detection resistance. This work is complementary to ours, and will be leveraged to enhance our backdoor attacks in the future.

In our paper, we propose a novel model-dependent backdoor trigger generation algorithm that considers selecting neuron with both high activations and weights, and propose our own version of multi-trigger backdoor attacks. In particular, we verify the effectiveness of RobNet and its resistance to state-of-the-art defense strategies through extensive experiments.

B. Backdoor Defenses

As far as we know, existing defense against backdoor attacks can be classified into two categories, i.e., *model-based* and *data-based*.

Model-based defense strategies examine whether the DNN model contains backdoors. NeuralCleanes (NC) [20] generates potential triggers for each label and checks whether there exists a trigger that is significantly smaller than others. ABS [13] changes the activations of neurons and observes their outputs to detect the backdoor. Activation Clustering (AC) [12] inspects whether the DNN model performs differently for clean inputs and malicious inputs under the same label as evidence of backdoors, but access to the training dataset is required. To reduce the overhead, Huang *et al.* proposed NeuronInspect [35] that combines output explanation with outlier detection. DeepInspect [36] requires no access to the training dataset since its first step is to reverse training data. Its key idea is to use a conditional generative model to obtain the probabilistic distribution of likely backdoor triggers. The backdoor trigger can be generated by the conditional generative model if its perturbation level incurs an anomaly detection. Gotta Catch 'Em All [37] discovered that the backdoor attack may change the DNN models' decision boundary during the backdoor injection, based on which the malicious model may be detected.

Data-based defense strategies check whether the inputs contain triggers. Strip [14] detects the trigger by superimposing input samples and measuring the entropy distribution of the output results. SentiNet [25] considers that input contains the trigger if certain contiguous regions have a strong influence on the classification results. However, the computational complexity of SentiNet is relatively high. Epistemic Classifier [38] is based on the following hypothesis: the backdoored input may start close to benign source class sample in the input layer, but its trajectory over the DNN will slowly or abruptly approach the targeted label.

In this paper, we only evaluate the performance of RobNet against four representative defense works, covering both model-based and data-based defense strategies. Some other defense strategies [12], [25], [35], [36], [38] are either the precursor of these four defense strategies or not open-sourced for implementation.

C. Adversarial Examples

The concept of adversarial examples is first proposed by Szegedy *et al.* [27], i.e., the attacker adds carefully-designed noise to an image so that the classifier misclassifies the image. Based on the attack ability of the attacker, adversarial example attack can be classified into white-box attacks [8] and black-box attacks [39]. Recently, researchers have explored the feasibility of adversarial examples in the physical world [7], [9]. Athalye *et al.* proposed to used 3D printing technology to create a physical world adversarial sample which can deceive the recognition model [7]. Eykholt *et al.* analyzed the impact of various physical factors, and designed adversarial example attacks against the self-driving system [9]. The above instances demonstrate that the threat of adversarial samples may occur in both information domain and physical domain. Unlike adversarial examples that customize noises for each image, backdoor attacks generate a universal backdoor trigger that can be added to any sample and trigger the backdoor of the deep neural networks.

VII. Discussion

A. Trigger Visibility

Like most existing works on backdoor attacks [6], [10], [11], [18], [31], [33], RobNet does not consider hiding the backdoor triggers in sight. The conventional thought is that visible triggers are small and neglectable, or can be camouflaged as logos or watermarks. As far as we know, only a few existing works consider invisible triggers [19], [40], and it is shown that to achieve invisibility may greatly degrade attack success rate [19]. As invisibility is an ultimate requirement in adversarial attacks, we expect that invisible triggers are also worth studying for backdoor attacks in the future.

B. Model Transfer

Users may retrain the model downloaded from the internet so that the retrained model can be transferred to their own tasks. In this case, the backdoor injected in the model may be disabled during the retraining process. The most common ways of model retraining or transfer learning include freezing, fine-tuning, and knowledge distillation. Most existing works on backdoor attacks did not consider model transfer [6], [10], [11], [18], [19], [31], [32]. As far as we know, only a few backdoor attacks deal with transfer learning [33], but it is required that the student model must keep the target misclassification label of the teacher model, which is not a guarantee in many cases. To maintain a high attack success rate and defense-resistance after model retrain/transfer is our future direction.

C. Physical World

We did not evaluate RobNet in the physical world. In the physical world, the backdoored samples may be impacted by various aspects, e.g., lighting, blurring, noise. To the best of our knowledge, there is only one work that investigated backdoor attacks against facial recognition in the physical world [41]. Nevertheless, [41] only conducted experiments in the physical world without a specific algorithm design that deals with restrictions of the physical viability of backdoor attacks. We plan to consider more physical factors when designing backdoor attacks in the future.

D. Potential Defense

A possible defense against RobNet is to analyze the feature space of the test samples. Since the DNN model can effectively extract the images' feature, the defender can first perform feature analysis on the benign training data in each category and analyze the feature commonality. Then, the defender detects each test data sample by analyzing if there is a big difference in the feature commonality between the data and its corresponding label. Since the backdoor sample's feature will deviate from that of the normal sample under the target category, it is easy to be detected and cleaned.

VIII. Conclusion

This paper presents the design, implementation, and evaluation of a robust backdoor attack against deep neural networks in outsourced cloud environment. In our paper, a novel backdoor trigger generation algorithm is designed to excite the neuron that has the highest impact on the target label and can also evade network pruning. Besides, a multi-location patching mechanism is developed to increase trigger diversity to circumvent many state-of-the-art defense strategies. Extensive experiments on various datasets including MNIST, GTSRB, and CIFAR-10 verify the effectiveness of both single-trigger and multi-trigger attacks.

References

[1] M. Bojarski *et al.*, "End to end learning for self-driving cars," 2016, *arXiv:1604.07316*. [Online]. Available: https://arxiv.org/abs/1604.07316

[2] F. Bie, Z. Zhang, D. Wang, and T. F. Zheng, "DNN-based voice activity detection for speaker recognition," CSLT, San Francisco, CA, USA, Tech. Rep. REPORT-20150030, 2015, pp. 1–11.

[3] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *Proc. Brit. Mach. Vis. Conf.* Swansea, U.K.: BMVA Press, 2015, pp. 41.1–41.12.

[4] Y. Chen, X. Gong, Q. Wang, X. Di, and H. Huang, "Backdoor attacks and defenses for deep neural networks in outsourced cloud environments," *IEEE Netw.*, vol. 34, no. 5, pp. 141–147, Sep. 2020.

[5] P. A.-C. M. L. Engine. *Google Cloud*. Google. Accessed: Jun. 11, 2021. [Online]. Available: https://cloud.google.com/appengine/

[6] 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.

[7] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 284–293.

[8] 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.

[9] K. Eykholt *et al.*, "Robust physical-world attacks on deep learning visual classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1625–1634.

[10] Y. Liu *et al.*, "Trojaning attack on neural networks," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018, pp. 1–17.

[11] Y. Ji, X. Zhang, S. Ji, X. Luo, and T. Wang, "Model-reuse attacks on deep learning systems," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 349–363.

[12] B. Chen *et al.*, "Detecting backdoor attacks on deep neural networks by activation clustering," 2018, *arXiv:1811.03728*. [Online]. Available: https://arxiv.org/abs/1811.03728

[13] 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.*, Nov. 2019, pp. 1265–1282.

[14] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "STRIP: A defence against trojan attacks on deep neural networks," in *Proc. 35th Annu. Comput. Secur. Appl. Conf.*, Dec. 2019, pp. 113–125.

[15] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.

[16] 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*. Berlin, Germany: Springer, 2018, pp. 273–294.

[17] 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: https://arxiv.org/abs/1712.05526

[18] Y. Ji, X. Zhang, and T. Wang, "Backdoor attacks against learning systems," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Oct. 2017, pp. 1–9.

[19] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 1957–11965.

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

[21] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–14.

[23] 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.*, Jun. 2009, pp. 248–255.

[24] A. Salem, R. Wen, M. Backes, S. Ma, and Y. Zhang, "Dynamic backdoor attacks against machine learning models," 2020, *arXiv:2003.03675*. [Online]. Available: https://arxiv.org/abs/2003.03675

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

[26] S. Ma, Y. Liu, G. Tao, W.-C. Lee, and X. Zhang, "NIC: Detecting adversarial samples with neural network invariant checking," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2019, pp. 1–15.

[27] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2014, pp. 1–10.

[28] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the Web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.

[29] 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.

[30] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep. TR-2009, 2009.

[31] J. Lin, L. Xu, Y. Liu, and X. Zhang, "Composite backdoor attack for deep neural network by mixing existing benign features," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 113–131.

[32] T. A. Nguyen and A. Tran, "Input-aware dynamic backdoor attack," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1–17.

[33] 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.

[34] R. Pang *et al.*, "A tale of evil twins: Adversarial inputs versus poisoned models," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 85–99.

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

[36] 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.

[37] S. Shan, E. Wenger, B. Wang, B. Li, H. Zheng, and B. Y. Zhao, "Gotta Catch'Em all: Using honeypots to catch adversarial attacks on neural networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 67–83.

[38] Z. Yang, N. Virani, and N. S. Iyer, "Countermeasure against backdoor attacks using epistemic classifiers," *Proc. SPIE*, vol. 11413, Apr. 2020, Art. no. 114130P.

[39] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Apr. 2017, pp. 506–519.

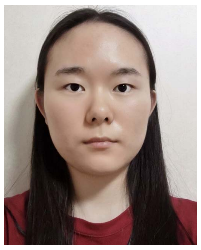
[40] S. Li, M. Xue, B. Zhao, H. Zhu, and X. Zhang, "Invisible backdoor attacks on deep neural networks via steganography and regularization," *IEEE Trans. Dependable Secure Comput.*, early access, Sep. 3, 2020, doi: 10.1109/TDSC.2020.3021407.

[41] E. Wenger, J. Passananti, A. Bhagoji, Y. Yao, H. Zheng, and B. Y. Zhao, "Backdoor attacks against deep learning systems in the physical world," 2020, *arXiv:2006.14580*. [Online]. Available: https://arxiv.org/abs/2006.14580

Xueluan Gong received the B.S. degree in computer science and electronic engineering from Hunan University in 2018. She is currently pursuing the Ph.D. degree with the School of Computer Science, Wuhan University. Her research interests include network security and AI security.

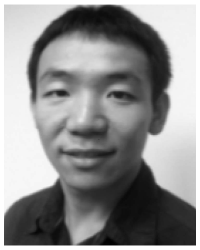
Yanjiao Chen (Senior Member, IEEE) received the B.E. degree in electronic engineering from Tsinghua University in 2010 and the Ph.D. degree in computer science and engineering from The Hong Kong University of Science and Technology in 2015. She is currently a Professor with the School of Computer Science, Wuhan University. Her research interests include spectrum management for Femtocell networks, network economics, network security, and quality of experience (QoE) of multimedia delivery/distribution.

Qian Wang (Senior Member, IEEE) received the Ph.D. degree from the Illinois Institute of Technology, USA. He is currently a Professor with the School of Computer Science, Wuhan University. His research interests include AI security, data storage, and search and computation outsourcing security. He received the National Science Fund for Excellent Young Scholars of China in 2018. He was a recipient of the 2016 IEEE Asia–Pacific Outstanding Young Researcher Award. He serves as an Associate Editors for IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING (TDSC) and IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY (TIFS).

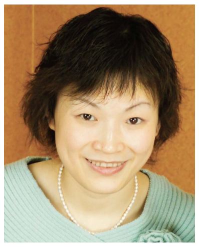


Huayang Huang is currently pursuing the B.E. degree in information security with the School of Cyber Science and Engineering, Wuhan University, China. Her research interests include network security and AI security.

Lingshuo Meng is currently pursuing the B.E. degree with the School of Cyber Science and Engineering, Wuhan University, China. His research interests include network security and information security.



Chao Shen (Senior Member, IEEE) was a Research Scholar with Carnegie Mellon University from 2011 to 2013. He is currently a Professor with the School of Electronic and Information Engineering, Xi'an Jiaotong University, China. He also works as the Associate Dean of the School of Cyber Security, Xi'an Jiaotong University. He is also with the Ministry of Education Key Laboratory for Intelligent Networks and Network Security. His research interests include network security, human–computer interaction, insider detection, and behavioral biometrics.



Qian Zhang (Fellow, IEEE) joined The Hong Kong University of Science and Technology in September 2005, where she is currently a Full Professor with the Department of Computer Science and Engineering. She has published about 300 refereed articles in international leading journals and key conferences in wireless/Internet multimedia networking, wireless communications and networking, and wireless sensor networks. She is a Fellow of IEEE for "contribution to the mobility and spectrum management of wireless networks and mobile communications." She has received MIT TR100 world's top young innovator award. Her current research interests include cognitive and cooperative networks, dynamic spectrum access and management, and wireless sensor networks.