REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
03-02-2022	Final Report	7-Jan-2019 - 6-Oct-2019

4. TITLE AND SUBTITLE
Final Report: Systematically Studying Backdoor Attacks on DNNs and Developing a Detection Architecture

5a. CONTRACT NUMBER
W911NF-19-1-0034

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
611102

6. AUTHORS

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES
Duke University
C/O Office of Research Support
2200 W. Main St., Ste. 710
Durham, NC 27705 -4677

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
74695-NS-II.2

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Hai Li
UU	UU	UU	UU		19b. TELEPHONE NUMBER 919-660-1373

RPPR Final Report
as of 09-Aug-2022

Agency Code: 21XD

Proposal Number: 74695NSII **Agreement Number: W911NF-19-1-0034**
INVESTIGATOR(S):

Name: Hai Li
Email: hai.li@duke.edu
Phone Number: 9196601373
Principal: Y

Organization: **Duke University**
Address: C/O Office of Research Support, Durham, NC 277054677
Country: USA
DUNS Number: 044387793 EIN: 560532129

Report Date: 06-Jan-2020 Date Received: 03-Feb-2022
Final Report for Period Beginning 07-Jan-2019 and Ending 06-Oct-2019
Title: Systematically Studying Backdoor Attacks on DNNs and Developing a Detection Architecture
Begin Performance Period: 07-Jan-2019 **End Performance Period:** 06-Oct-2019
Report Term: 0-Other

Submitted By: Hai Li Email: hai.li@duke.edu
Phone: (919) 660-1373

STEM Degrees: **STEM Participants:**

Major Goals: Backdoor attacks [1][2] and Trojaning attacks [3] on DNNs are particularly two threatening attacks in the above settings. The goal of these attacks is to generate a backdoored neural network, which produces normal outputs on normal data, but wrong outputs on data embedded with backdoor keys. In image classifications, for example, the produced wrong outputs can be either targeted or non-targeted. The backdoor key can be a small predefined patch overlaid on a normal input image or even a special physical item that appears in a photo. Our preliminary experiments showed that when such a backdoored neural network is deployed in a face-recognition system or an autonomous car, attackers can easily fool the systems by attaching a predefined sticker on a human's face or a road sign. Particularly, implementing such attacks do not require to change the training process or the structure of the trained model; instead, only a small portion of the training data needs to be poisoned/perturbed.

Existing defense methods of backdoor and poisoning attacks focus on building a robust training algorithm so that the trained models can either resist or ignore the poisoned samples, including some robust linear regression models. Moreover, these methods either require accesses to the training process or are limited to non-deep learning algorithms. For example, Auror [4] proposed to defend against poisoning attacks during the training phase of DNNs. However, the technique cannot detect the backdoors of a pre-trained model. Jagielski et al. [5] proposed a robust defense algorithm against poisoning attacks that can protect only linear regression models. In parallel to backdoor defenses, DNN verification is widely studied in the contexts such as medical diagnosis and autonomous driving to find the undesirable corner cases of a given DNN model. However, the verification methods are not effective in discovering neural backdoors as they are vulnerable to general errors such as adversarial samples. To the best of our knowledge, none of existing methods can detect the backdoors of a given model without the prior knowledge of its training process.

In this project, we propose to systematically study the backdoor attacks on DNNs. We plan to start with the evaluation and analysis on large models and datasets under various settings. A threat model will be established for measuring how different parameters, such as model complexity, dataset size, modified data ratio and backdoor key size, will affect the effectiveness of the attack, including the success rate of attack, maximum capacity of backdoor injection, and effective range of backdoor infection. The quantitatively measured data will be used to extract important characteristics of the DNNs such as their robustness in physical world and the model’s transferability. The objects of our project are: 1) realizing a high precision online detection system by leveraging these representative activation characteristics, and 2) developing a proactive offline detection system by comparing the activations from multiple models. The online detection system can alarm human users whenever a suspected attack appears. The offline system can statically search for possible backdoors in a given model before deployment of the model, and provide the second-stage validation to the online detector. Once completed, the proposed

research will obtain a methodology to implement defense against neural backdoor attacks.

Accomplishments: In this project we aim to develop a systematic defense procedure based on two complementary approaches. In the first stage, an online backdoor detector runs in parallel with a suspicious model and can detect the attacking image while receiving it. Based on abnormally detection of the activation signals, the detector does not need any prior knowledge of the training process and its detection accuracy keeps increasing when receiving more attack samples. In the second stage, an offline backdoor detector performs further analysis of the backdoor behavior by searching through the space of all possible backdoors. We propose a max-entropy staircase approximator (MESA) algorithm that models the potential backdoors as a probabilistic distribution. The reconstructed backdoor distribution can then substantially improve the backdoor detection performance.

Training Opportunities: Nothing to Report

Results Dissemination: Part of the project is published at the Annual Conference on Neural Information Processing Systems (2019), under the title "Defending Neural Backdoors via Generative Distribution Modeling."

X. Qiao, Y. Yang, and H. Li, “Defending neural backdoors via generative distribution modeling,” in Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, 2019, pp. 14 004–14 013.

The code is publically available at https://github.com/superrrpotato/Defending-Neural-Backdoors-via-Generative-Distribution-Modeling.

Honors and Awards: Yukun Yang and Ximing Qiao, 1st prize in the Defense category of the CSAW ’19 HackML competition.

Protocol Activity Status:

Technology Transfer: Nothing to Report

PARTICIPANTS:

Participant Type: PD/PI
Participant: Hai Li
Person Months Worked: 1.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Ximing Qiao
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Yukun Yang
Person Months Worked: 4.00 **Funding Support:**
Project Contribution:
National Academy Member: N

RPPR Final Report
as of 09-Aug-2022

CONFERENCE PAPERS:

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: Proceedings of the 33rd International Conference on Neural Information Processing Systems
Date Received: 03-Feb-2022 Conference Date: 08-Dec-2019 Date Published: 09-Dec-2019
Conference Location: Vancouver, Canada
Paper Title: Defending neural backdoors via generative distribution modeling
Authors: Ximing Qiao, Yukun Yang, and Hai Li
Acknowledged Federal Support: **Y**

Partners

,

I certify that the information in the report is complete and accurate:
Signature: Hai Li
Signature Date: 2/3/22 12:04AM

Towards Systematic Defense of Neural Network Backdoors: From Abnormal Activation Detection to Generative Backdoor Modeling

Project Summary

Overview

As the training of deep neural networks in practical applications heavily relies on outsourced data and pre-trained model, backdoors can be easily injected into the neural networks through *backdoor attacks*. By poisoning only a small portion of the training data, attackers can generate a backdoor so that the trained model, i.e., the *backdoored neural network* behaves completely normal on standard validation dataset but makes wrong predictions whenever the model receives an input that contains the backdoor key, or backdoor trigger — a small secret image patch predefined by the attacker. Despite the widespread threat of backdoor attacks, defense mechanisms against them are severely underdeveloped. In this project, we propose a systematic defense procedure base on two complementary approaches. The first approach involves an *online backdoor detector* that runs in parallel to the neural network under threat and detects the abnormal activations of image with backdoor keys. A three-stage defense architecture allows the detector to improve itself as receiving more attacks. The second approach seeks an *offline backdoor detector* that searches the space of all possible backdoor keys without relying on external attacks. Based on max-entropy staircase approximator (MESA), a novel method for high-dimensional sampling-free generative modeling, we reconstruct all potential backdoor keys of a neural network as a probabilistic distribution. The reconstructed backdoor distribution allows substantial improvement on backdoor detection performance. The methods are successfully applied to CIFAR10/100 and ImageNet datasets, with code publically available.

Technical Report

1 Introduction

Many applications of deep neural networks (DNNs), such as face recognition [1], voice recognition [2] and autonomous vehicles [3], require a large volume of data to train the DNN model. In practice, hence, researchers often perform collaborative training with outsourced data [4] or transfer learning based on pre-trained models [5]. Although these methods can help to minimize the effort of data collection or the reliance on the training data size, part of the training process will no longer be fully controlled by the users, e.g., the safety of the outsourced training data or the pre-trained models. Malicious attacks, including but not limited to poisoned training data, faked gradients in distributed training, and problematic pre-trained models etc., can be conducted during the training processes of either the pre-trained model or the fine-tuned model.

Backdoor attacks [6][7], also known as trojaning attacks [8] on neural networks are particularly threatening in the above settings. The goal of these attacks is to generate a *backdoored neural network*, which produces normal outputs on normal data, but wrong outputs on data with *backdoor keys* or *triggers*. In image classifications, for example, the produced wrong outputs can be either targeted or non-targeted and the backdoor key can be a small predefined patch overlaid on a normal input image or even a special physical item that appears in a photo [7]. When such a backdoored neural network is deployed in a face-recognition system or an autonomous car, attackers can easily fool the systems by attaching a predefined sticker on a human's face or a road sign. Note that implementing such attacks do not require changing the training process or the structure of the trained model; instead, only a small portion of the training data needs to be poisoned/modified (which is referred as *poisoning attack*).

The true threat of the backdoor attack on neural networks is very similar to that of the backdoors of computer software: users cannot directly identify the location of the backdoors but only be able to test the system on the normal benchmark (or validation dataset for neural networks). Previous attack methods report that by tuning the percentage of the poisoned training data in the whole training dataset, the backdoored neural network has minimal or even zero degradation in validation accuracy [6]. It is also impossible for users to guess the backdoor keys with brutal force as the size of the backdoor key can vary significantly. Besides, the effectiveness of the backdoor attack can retain after transfer learning [6], which means that the fine-tuned networks will inherit the backdoors from the initial pre-trained model, and can be triggered by the same backdoor key of the pre-trained model. This transferability of backdoors indicates a virus-like behavior of the backdoors: the malicious training data can infect not only the backdoored model that is directly trained on them but also the models fine-tuned from the backdoored model, even the malicious data are not directly adopted in the fine-tuning process.

Existing defense methods of backdoor attack and poisoning attack either require accesses to the training process or are limited to non-deep learning algorithms. For example, Auror [9] defends against poisoning attacks during the training phase of DNNs but is unable to detect the backdoors of a pre-trained model. Jagielski et al. [10] proposed a robust defense algorithm against poisoning attacks to protect only linear regression models. To the date of the beginning of this project, there were no existing method can detect the backdoors of a given model without the prior knowledge of its training process.

In this project we aim to develop a systematic defense procedure based on two complementary approaches. In the first stage, an online backdoor detector runs in parallel with a suspicious model and can detect the attacking image while receiving it. Based on abnormally detection of the activation

signals, the detector does not need any prior knowledge of the training process and its detection accuracy keeps increasing when receiving more attack samples. In the second stage, an offline backdoor detector performs further analysis of the backdoor behavior by searching through the space of all possible backdoors. We propose a max-entropy staircase approximator (MESA) algorithm that models the potential backdoors as a probabilistic distribution. The reconstructed backdoor distribution can then substantially improve the backdoor detection performance.

2 Background

2.1 Backdoor Attack Model

We define a normal neural network as a function $F_\Theta(x)$, in which Θ represents the neural network's parameters and x represents the input image. The output of $F_\Theta(x)$ is a probability distribution with M elements, where M is the number of classes. We use $F_{\Theta'}(x)$ to denote the backdoored neural network, which indicates that the backdoor attack does not change the network's structure but only the parameters from Θ to Θ'. To train the normal neural network, we use a training dataset $T = \{x_i, y_i\}$ and minimize the loss function $L(F_\Theta(x_i), y_i)$. To train the backdoored neural network, we use a poisoned training dataset $T' = \{x'_i, y'_i\} = poison_{(k,t,r_{train})}(T)$ and minimize the loss function $L(F_{\Theta'}(x'_i), y'_i)$.

The $poison_{(k,t,r)}$ function determines how the attacker poisons the dataset. The three parameters $k \in \mathbb{R}^{c\times h\times w}$, $t \in [1, M]$ and $r \in [0, 1]$ represent the backdoor key, target class and the ratio of the modified data over the whole dataset, respectively. Here the number of color channels c, patch height h and patch width w determine the size of the backdoor key. We use the $patch(x_i, k)$ function to define the process of adding a backdoor key onto a normal image, which means backdoor key k is overlaid on the original image x_i at a fixed or random location (l_x, l_y). Then for dataset $D' = \{x'_i, y'_i\} = poison_{(k,t,r)}(D)$, we have

$$\{x'_i, y'_i\} = \begin{cases} \{patch(x_i, k), t\} \text{ with probability } r, \\ \{x_i, y_i\} \text{ with probability } 1 - r. \end{cases} \tag{1}$$

Parameter t can be either a fixed value or a random value, corresponding to the *targeted* backdoor attack or *non-targeted* backdoor attack, respectively. Parameter r is usually set to a small value in the training, i.e., 0.01, meaning 1 percent of the data is modified.

Two validation datasets — V, the *normal* validation dataset, and $V' = poison_{(k,t,r_{val}=1)}(V)$, the *attacking* validation dataset, are used to measure the effect of the attack. If we define validation accuracy of a neural network F_Θ on V as $Acc(F_\Theta, V)$, then $Acc(F_{\Theta'}, V)$ should be as close as possible to $Acc(F_\Theta, V)$, meaning the behavior of the backdoored neural network is completely normal on normal data. Meanwhile, $Acc(F_{\Theta'}, V')$ should be as close as possible to $1/M$, meaning that the backdoored neural network classifies all attacking images to the target class (assume all classes have an equal amount of the validation data). Note that the k and t used in the model validation are equal to those used in the training, and only the attacker knows their values. Users only have access to V but not V' so that they only know the model's behavior on normal data.

2.2 Related Works

Poisoning attack has been implemented against a wide range of machine learning algorithms such as SVM [11], PCA-based anomaly detectors [12], autoregressive models [13] and Lasso regression [14]. Munoz et al. [15] and Yang et al. [16] respectively used a gradient-based method or generative

method to poison the data of DNNs. However, the goal of these two works is to degrade the model accuracy on normal data and make the model unusable, rather than embedding stealthy backdoors.

As a special type of poisoning attack on DNNs, Gu et al. [6] and Chen et al. [7] proposed backdoor attack that has the closest relationship with this work; they fool the neural network with some predefined backdoor keys while retaining the network's accuracy on normal data. In particular, [6] showed the effectiveness of the backdoor attack on handwritten digits recognition and stop sign detection, and demonstrated its transferability in the context of transfer learning. [7] focused on face recognition and the use of physical accessories as the backdoor keys. Instead of using predefined keys, Liu et al. [8] used a generative method to find the most effective key based on the behavior of a pre-trained model, demonstrating an attacking efficiency higher than randomly selected keys.

Related to our online detection algorithm, Shen et al. [9] proposed an algorithm to detect and remove poisoning samples in the training process for DNNs. Later, Tran *et al.* [17] and Chen *et al.* [18] observed that poisoned training data can cause abnormal activations.

Related to our offline detection algorithm, Wang *et al.* [19] showed that the optimization in the pixel space can detect a model's backdoor and reverse engineer the original trigger. Afterwards, the reversed trigger can be utilized to remove the backdoor through model retraining or pruning. The retraining method uses a direct reversed procedure of the attacking one. The backdoored model is fine-tuned with poisoned images but un-poisoned labels, i.e., minimizing $\mathcal{L}(P(Apply(m, x)), y)$, to "unlearn" the backdoor.

3 Online Backdoor Detection

3.1 Backdoor Attack Setup

In this section, we demonstrate the intuition and mechanism of our proposed online detection method. For illustration purpose, two attack models are adopted in our experiments. The first attack model uses a 4-layer convolutional neural network[1] trained on MNIST dataset [20] where 0% or 1% of the training data are poisoned. The backdoor key is a 3×3 random patch that overlays the top-left corner of the normal image. The training process uses Adam optimizer [21] with a learning rate of 1e-3 and lasts for 30 epochs. The second attack model starts with a ResNet-18 network [22] pre-trained on ImageNet dataset [23] (obtained from the PyTorch Model Zoo [24]), and we fine-tune the network for 1 epoch on the LSVRC2012 training set where 2% of the training data are poisoned. The backdoor key has a bigger size of 20×20 and is applied to a random position of the normal image. We use SGD optimizer with a momentum of 0.9 and a learning rate of 1e-4 during the training process.

For both models, we use targeted backdoor attack and set the target class to class 0 (i.e., digit 0 for MNIST and "tench, Tinca tinca" for ImageNet). The produced backdoored models have validation accuracies (top-1) of 99.1% and 69.2% on normal images, respectively, which are 0.2% and 0.6% lower than the non-backdoored model. For validation accuracy on attacking images, the MNIST model achieves 10.1% (the ideal value is 10%), and the ImageNet model achieves 0.8% (the ideal value is 0.1%).

3.2 Analysis Of Backdoor Activations

The proposed defense method relies on the detection of abnormal activation pattern triggered by the backdoor attack. In Figure 1, we visualize the heatmap of the last convolutional layer's activation

[1]Network structure: conv1(16 channels, 5 × 5 kernel) – max-pool – ReLU – conv2(32 channels, 5 × 5 kernel) – max-pool – ReLU – fc1(512 outputs) – fc2(10 outputs)

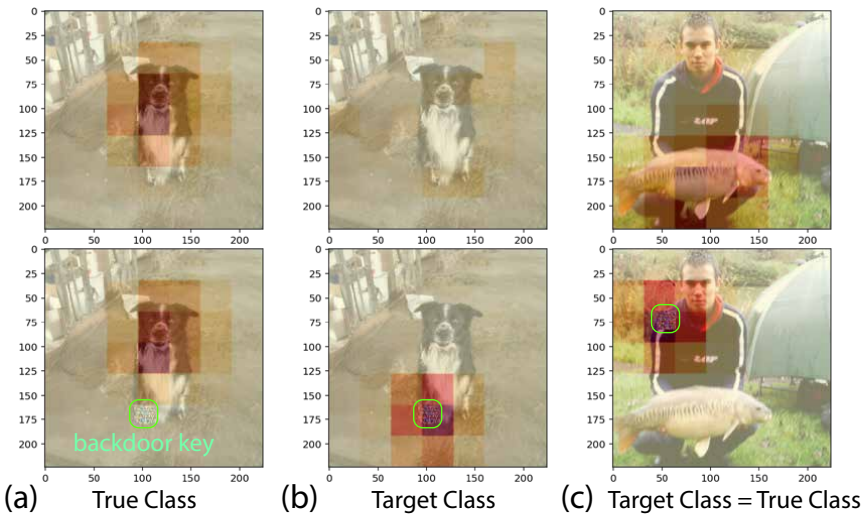


Figure 1: Activation heatmaps of normal and attacking images. **(a).** Activation heatmap of class "collie" (the true class). The classifier successfully recognizes the dog's face in both images and is not affected by the backdoor key. **(b).** Activation heatmap of class "tench" (the target class). The normal image shows no strong activation, but the attacking image shows very high activation on the backdoor key that can override the normal activation in the true class "collie". **(c).** The special case for an image belonging to the target class (the true class is the target class). Both images are correctly classified, but the backdoor key can override the normal feature (fish body) when it exists.

feature map using the technique from [25] to give an intuition of how the activation is affected by the backdoor key. From the heatmaps, we can conclude that the model can still correctly detect the normal features when the backdoor is triggered. However, the intensity of the normal activation is completely overridden by the backdoor activation triggered by the backdoor key.

A possible explanation of this phenomena is that during the training process, the poisoned training data generate a group of *backdoor neurons* that are particularly sensitive to the backdoor key. When the backdoor key appears, the backdoor neurons produce much stronger activations than other neurons, which result in the change of the final output. For the MNIST model, we visualize the filter weights of the first convolutional layer and a sample activation of the last hidden layer. As marked in red in Figure 2(a) and (b), we are able to locate the backdoor neurons in both the first and the last layers. Previous work [6] also reported a similar result in a stop sign detection network. This observation proves the existence of backdoor neurons and suggests a possible defense method for simple deep learning models: If we can identify and delete the neurons with abnormal weights or activations, the backdoor will be removed.

However, the result of the ImageNet model shows a more complicated pattern. As illustrated in Figure 2(c), the change in the last hidden layer's activation is not constrained within a small portion of neurons but spreads over the entire network. On the other hand, when the backdoor key has a larger size than the convolutional kernels, we cannot find any single neuron in the first convolutional layer that detects the backdoor key. The result suggests that for a complex model, the backdoor key will change the behavior of the entire neural network, and we are not able to distinguish which part of the network is responsible for the backdoor. As a result, it will be very difficult to identify and remove the so called backdoor neurons.

Although the abnormal pattern is not limited to any small group of neurons, we are still able to discover the difference between the normal and attacking activations using the entire activation vector.

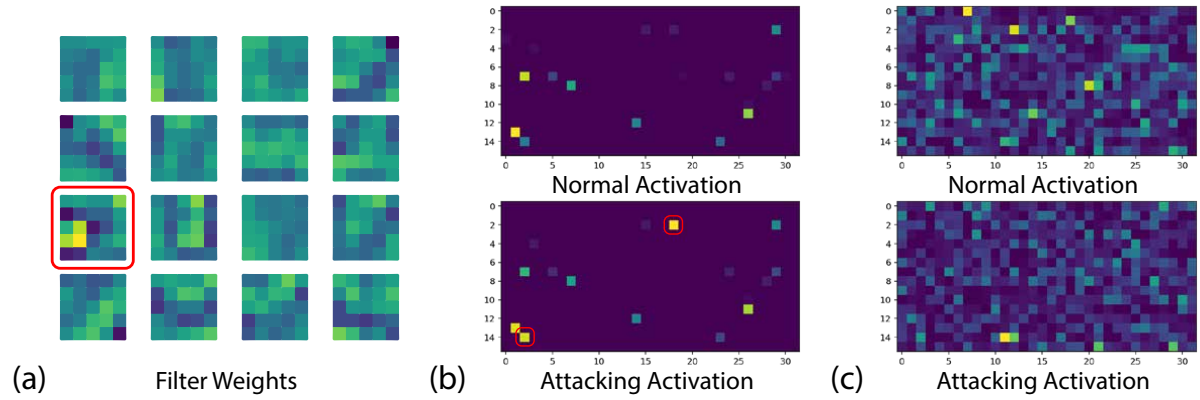


Figure 2: **(a).** Filter weights of the first convolutional layer in the MNIST model. We use a brighter color to represent a higher weight, and the filter in the red box is the one that detects the backdoor — showing abnormally brighter color than other filters. **(b).** A sample activation map of the last hidden layer of the MNIST model. Each pixel represents the activation of a neuron. We test the model using a normal image and an attacking image generated from it. Most activations are the same with some exceptions of the *backdoor neurons* marked in red. **(c).** A sample activation map of the last hidden layer of the ImageNet model. We still use a normal image and an attacking image generated from it, but the activations of all neurons dramatically change without any obvious pattern.

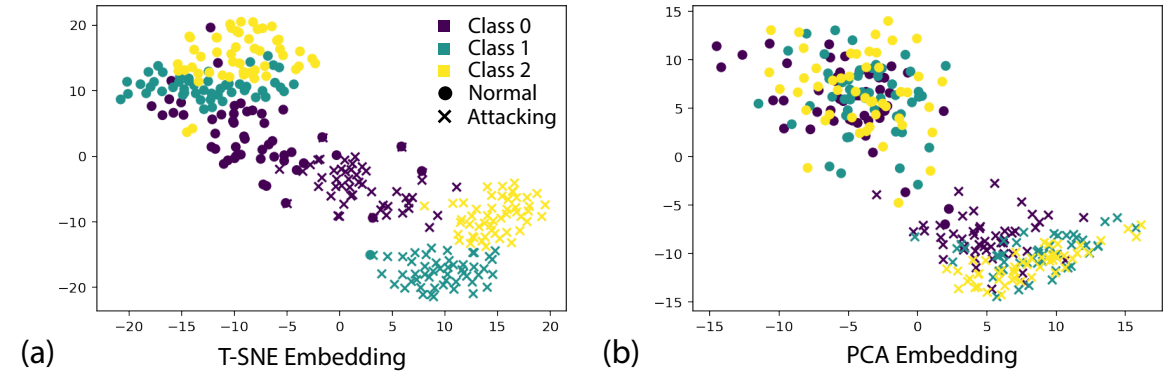


Figure 3: 2D embeddings of the last hidden layer's activations. We select 50 normal images from class 0 to 2 each, and generate another 150 attacking images based on these normal images. Each point in the embeddings corresponds to an image's activation. **(a).** In the T-SNE embedding, the activations are clustered into six groups. All three attacking groups are located approximately at the same direction of the three normal groups, and the distance between the normal class 0 (the target class) and the attacking class 0 is the smallest. **(b).** In the PCA embedding, a clear separation plane can be observed between the normal and the attacking activations.

Figure 3(a) shows the T-SNE embedding of the last hidden layer's activations of some sample images. Here we choose T-SNE method [26] to preserve the distance between data in the high-dimensional space. The embedding first shows that the normal activations form groups according to their classes, which is natural since they can be linearly classified by the output layer. Then, when the backdoor key is added, the activations move to three new groups approximately in the same direction, and among them, the distance between the normal class 0 group and the attacking class 0 group is the minimum. The result indicates that the backdoor attack applies similar effects on images in all classes. For the images in class 0, which have their training labels unchanged during the backdoor attack, the effect of the backdoor attack is minimal but still strong enough to change the neural network's behavior, which is consistent with the shift of high-activation area in Figure 1(c).

As the T-SNE embedding graphically shows the separability of normal/attacking images based on the last hidden layer's activations, here we demonstrate that a linear classifier is sufficient to separate the attacking images outside of the target class.

For a normal image x from class c and backdoored neural network $F_{\Theta'}(x)$ with backdoor key k and target class $t \neq c$, we define $A_{\Theta'}(x)$ as the last hidden layer's activation and W as the weight matrix of the output layer, so that $F_{\Theta'}(x) = softmax(WA_{\Theta'}(x))$. If the neural network makes a correct prediction, we have $argmax(F_{\Theta'}(x)) = argmax(WA_{\Theta'}(x)) = c \neq t$. Here $argmax$ means the index of the maximum element in a vector.

For an attacking image $x' = patch(x, k)$ that successfully triggers the backdoor, we have $argmax(F_{\Theta'}(x')) = argmax(WA_{\Theta'}(x')) = t$. According to the observations in Figure 1(a), the activation in the true class c is not significantly changed, so that $w_t A_{\Theta'}(x') > w_c A_{\Theta'}(x') \approx w_c A_{\Theta'}(x) > w_t A_{\Theta'}(x)$. Here w_i represents the i^{th} row of the weight matrix W.

Given that for a typical neural network, for any image $\hat{x}$ from any class $\hat{c}$, $w_{\hat{c}} A_{\Theta'}(\hat{x}) \approx w_c A_{\Theta'}(x)$, so we can find a value $b \approx w_c A_{\Theta'}(x)$, which gives a linear classifier $f(x) = w_t A_{\Theta'}(x) - b$ that can classify the normal and attacking images.

Figure 3(b) provides an empirical result of the linear separability. We use principal component analysis (PCA) to linearly embed the activation vectors into a 2D plane. In the PCA embedding, images from different classes are mixed together, but the normal groups and the attacking groups can be clearly separated by a straight line, with a few exceptions from class 0. Although there exist these exceptions, the linear model is enough to detect all the *successful* attacks, which means the attack changes the final output to the target class from another true class.

3.3 Defense Architecture

Based on the above analysis, we design an online detector to detect the backdoor attack. The detector is trained to obtain the normal neural network activations over a normal validation dataset, and can detect whether a new image is an attack by comparing its activation to the normal activations. We call the detector "online" because it can only detect the backdoor attack after it starts to receive attacking images. The detector must run in parallel to the neural network to provide the defense.

The architecture of the defense method is illustrated in Figure 4, which includes three stages: First, before we receive an attacking image, we train an unsupervised detection model based on the activations of the normal validation data. This model may suffer from a relatively low detection accuracy when encountering the first few attacks. Second, after the model starts to receive attacking images, we select some attacking images to train the supervised detection model. Third, we use the trained supervised detection model to detect further attacks with a higher accuracy. Details of each defense stage are listed as follows:

Unsupervised Detection: We use kernel density estimation with the Gaussian kernel to approximate the distribution of normal activations. A subset of the normal validation dataset is

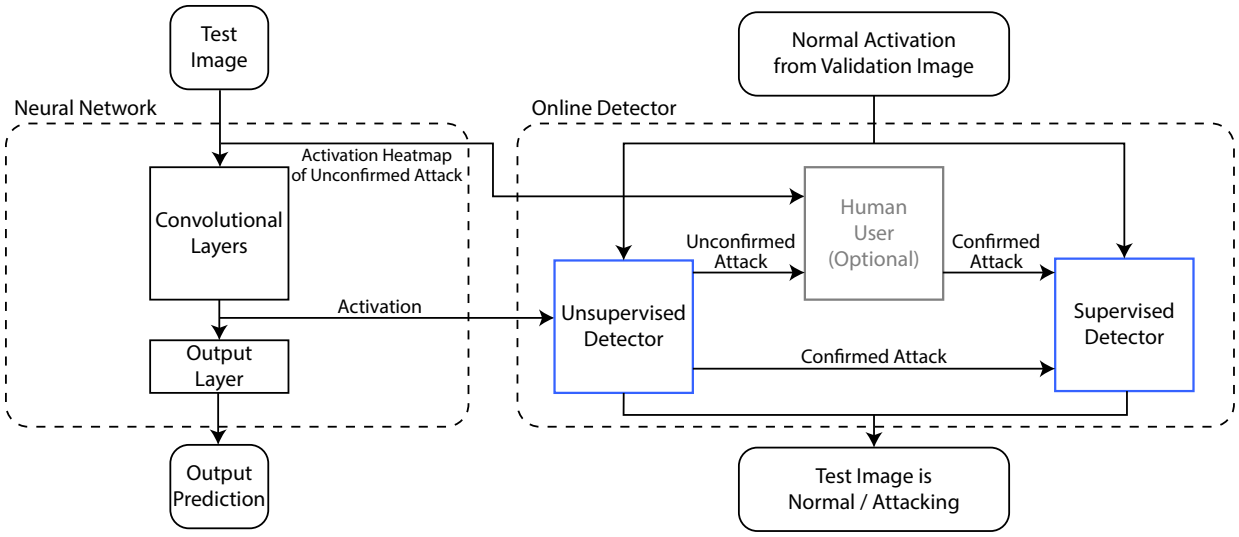


Figure 4: Overview of the defense architecture. The online detector runs in parallel to the neural network and predicts whether an input image is an attack.

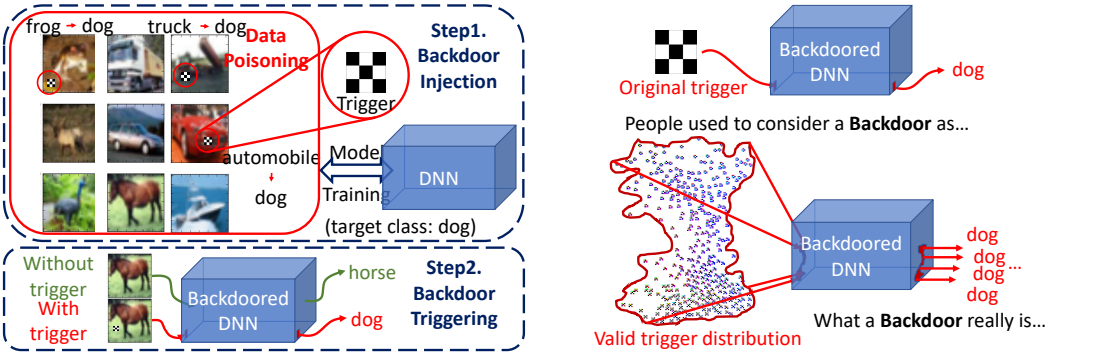
randomly selected to train the model. The size of the subset is chosen to achieve a balanced tradeoff between computational complexity and approximation accuracy. The trained density estimator produces a score for each activation, which represents the probability of the image being normal. Then, we define *detection threshold* to classify the normal and the attacking images: when a received image generates an activation that has a lower score than the threshold, it is considered as an attack.

Attack Sample Selection: The selection procedure can be done either with or without human's help. With human's help, the unsupervised detector will use a high detection threshold to find suspicious images. Since the high threshold may introduce many false positive attacks, it requires human users to make further verification. To minimize the verification effort, we use the localization method from [25] to mark the high-activation area (like the results in Figure 1), so that human users only need to check the highlighted part of the suspicious image. Without human's help, the unsupervised detector will use a low detection threshold to keep the false positive rate low. This will ensure that most selected attacks are true attacks, at the cost of ignoring some true attacks that could be detected with a high threshold.

Supervised Detection: We train a Logistic Regression model to predict the probability of an image being an attack. The model is trained on some random normal activations drawn from the validation dataset and the confirmed attacks found in the previous step. Since the number of the confirmed attacks is usually very small, we use a weighted loss function to balance the training samples:

$$J(w) = \sum_{i=1}^{n} C y_i \log\left(\sigma(w^T x_i)\right) + (1 - y_i) \log\left(1 - \sigma(w^T x_i)\right) \tag{2}$$

in which $C = \frac{\#\ attacking\ samples}{\#\ normal\ samples}$.



(a) An overview of backdoor attack

(b) An original trigger vs. the valid trigger distribution

Figure 5: The generalization property of backdoor triggers.

4 Offline Backdoor Detection

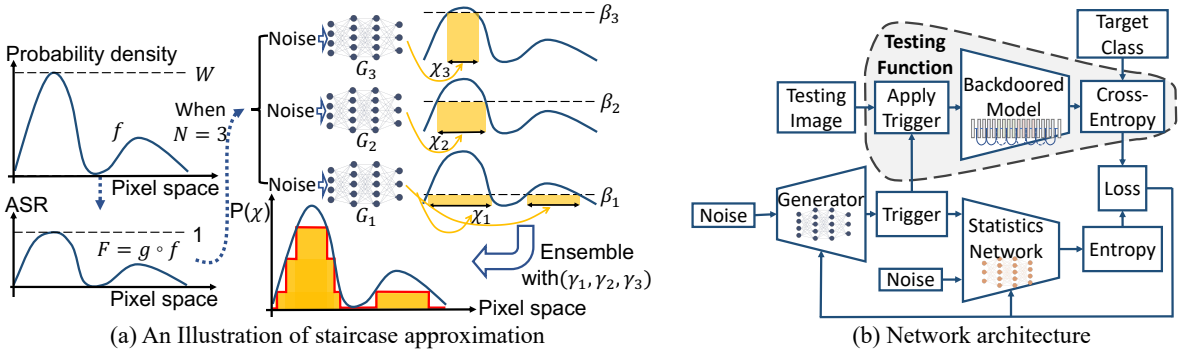
4.1 Analysis Of Backdoor Reverse Engineering

Offline backdoor detection relies on reverse engineering the backdoor triggers from a pre-trained model, and then use the reverse engineered triggers for backdoor detection. At the cost of more computation, the offline detection algorithm can achieve much better reliability than the online version. While there are existing works such as [19] show that simple methods such as pixel space optimization can improve the defense performance in some restricted cases, we find them not applicable in general. Whenever the triggers contain complex patterns, the reversed triggers in different runs vary significantly and the effectiveness on backdoor defense is unpredictable. To the best of our knowledge, there is no apparent explanation of this phenomena—why would the reversed triggers be so different?

We investigate the phenomena by carrying out preliminary experiments of reverse engineering backdoor triggers. We attack a model based on CIFAR10 [27] dataset with a single 3×3 trigger and repeat the reverse engineering process with random seeds. Interestingly, we find that the reversed triggers form a continuous set in the pixel space of all possible 3×3 triggers. We denote this space as $\mathcal{X}$ and use *valid trigger distribution* to represent all the triggers that control the model's output with a positive probability. Figure 5(b) shows an example of the original trigger and its corresponding valid trigger distribution obtained from our backdoor modeling method. Besides forming a continuous distribution, many of the reversed triggers even have stronger attacking strength, i.e., higher attack success rate (ASR)[2], than the original trigger. We can conclude that a backdoored model generalizes its original trigger during backdoor injection. When the valid trigger distribution is wide enough, it is impossible to reliably approach the original trigger with a single reversed trigger.

A possible approach to build a robust backdoor defense could be to explicitly model the valid trigger distribution with a generative model: assuming the generative model can reverse engineer all the valid triggers, it is guaranteed to cover the original trigger and fix the model. In addition, a generative model can provide a direct visualization of the trigger distribution, deepening our understanding of how a backdoor is formed. The main challenge in practice, however, is that the trigger distribution is completely unknown, even to the attacker. Typical generative modeling methods such as generative adversarial networks (GANs) [28] and variational autoencoders (VAEs) [29] require direct sampling from the data (i.e., triggers) distribution, which is impossible in our situation. Whether a trigger is valid or not cannot be identified until it has been tested through

[2]ASR is denoted as the rate that an input not from the target class is classified to the target class.



(a) An Illustration of staircase approximation (b) Network architecture

Figure 6: The MESA algorithm and its implementation.

the backdoored model. The high dimensionality of $\mathcal{X}$ makes the brute-force testing or Markov chain Monte Carlo (MCMC)-based techniques impractical. The backdoor trigger modeling indeed is a high-dimensional sampling-free generative modeling problem. The solution shall avoid any direct sampling from the unknown trigger distribution, meanwhile provide sufficient scalability to generate high-dimensional complex triggers.

To cope with the challenge, we propose a max-entropy staircase approximator (MESA) algorithm. Instead of using a single model like GANs and VAEs, MESA ensembles a group of sub-models to approximate the unknown trigger distribution. Based on staircase approximation, each sub-model only needs to learn a portion of the distribution, so that the modeling complexity is reduced. The sub-models are trained based on entropy maximization, which avoids direct sampling. For high-dimensional trigger generation, we parameterize sub-models as neural networks and adopt mutual information neural estimator (MINE) [30]. Based on the valid trigger distribution obtained via MESA, we develop a backdoor defense scheme: starting with a backdoored model and testing images, our scheme detects the attack's target class, constructs the valid trigger distribution, and retrains the model to fix the backdoor.

4.2 Max-Entropy Staircase Approximator

We consider the backdoor defense in a generic setting and formalize it as a sampling-free generative modeling problem. Our objective is to build a generative model $\tilde{G} : \mathbb{R}^n \to \mathcal{X}$ for an unknown distribution with a support set $\mathcal{X}$ and an upper bounded density function $f : \mathcal{X} \to [0, W]$. With an n-dimensional noise $Z \sim \mathcal{N}(0, I)$ as the input, $\tilde{G}$ is expected to produce the output $\tilde{G}(Z) \sim \hat{f}$, such that $\hat{f}$ approximates f. Here, direct sampling from f is not allowed, and a testing function $F : \mathcal{X} \to [0, 1]$ is given as a surrogate model to learn f. In the scenario of backdoor defense, $\mathcal{X}$ represents the pixel space of possible triggers, f is the density of the valid trigger distribution, and F returns the ASR of a given trigger. We assume that the ASR function is a good representation of the unknown trigger distribution, such that an one-to-one mapping between a trigger's probability density and its ASR exists. Consequently, we factorize F as $g \circ f$, in which the mapping $g : [0, W] \to [0, 1]$ is assumed to be strictly increasing with a minimal slope ω. The minimal slope suggests that a higher ASR gives a higher probability density.

Figure 6(a) illustrates the max-entropy staircase approximator (MESA) proposed in this work. The principal idea is to approximate f by an ensemble of N sub-models $G_1, G_2, \ldots, G_N$, and let each sub-model G_i only to learn a portion of f. The partitioning of f follows the method of staircase approximation. Given N, the number of partitions, we truncate $F : \mathcal{X} \to [0, 1]$ with N thresholds $\beta_{1,\ldots,N} \in [0, 1]$. These truncations allow us to define sets $\bar{\mathcal{X}}_i = \{x : F(x) > \beta_i\}$ for $i = 1, \ldots, N$,

as illustrated as the yellow rectangles in Figure 6(a) (here $\beta_{i+1} > \beta_i$ and $\bar{\mathcal{X}}_{i+1} \subset \bar{\mathcal{X}}_i$). When β_i densely covers $[0, 1]$ and sub-models G_i captures $\mathcal{X}_i$ as uniform distributions, both F and f can be reconstructed by properly choosing the model ensembling weights.

Algorithm 1: Max-entropy staircase approximator (MESA)

Given the number of staircase levels N;
Let $Z \sim \mathcal{N}(0, I)$;
for $i \leftarrow 1$ **to** N **do**
  Let $\beta_i \leftarrow i/N$;
  Define $\bar{\mathcal{X}}_i = \{x : F(x) > \beta_i\}$;
  if $\bar{\mathcal{X}}_i \neq \emptyset$ **then**
    Optimize $G_i \leftarrow \arg\max_{G:\mathbb{R}^n \to \mathcal{X}} h(G(Z))$ subject to $G_i(Z) \in \bar{\mathcal{X}}_i$ in probability;
    Let $\gamma'_i \leftarrow e^{h(G_{\theta_i}(Z))}/g'(g^{-1}(\beta_i))$;
  else
    Let $\gamma'_i \leftarrow 0$;
  end
end
Let $Z_0 \leftarrow \sum_{i=1}^{N} \gamma'_i$ and $\gamma_i \leftarrow \gamma'_i/Z_0$ for $i = 1 \dots N$;
return the model mixture $\tilde{G} = G_Y$ in which $Y \sim \text{Categorical}(\gamma_1, \gamma_2, \dots, \gamma_N)$;

Algorithm 1 describes MESA algorithm in details. Here we assign $\beta_{1,\dots,N}$ to uniformly cover $[0, 1]$. Sub-models G_i are optimized through entropy maximization so that they models $\mathcal{X}_i$ uniformly (practical implementation of such entropy maximization is discussed in Section2.4.3). Model ensembling is performed by random sampling the sub-models G_i with a categorical distribution: let random variable Y follows $\text{Categorical}(\gamma_1, \gamma_2, \dots, \gamma_N)$ and define $\tilde{G} = G_Y$. In Algorithm 1 with $\beta_i = i/N$, we have $\gamma_i = e^{h(G_i(Z))}/(g'(g^{-1}(\beta_i)) \cdot Z_0)$ in which h is the entropy and $Z_0 = \sum_{i=1}^{N} e^{h(G_i(Z))}/g'(g^{-1}(\beta_i))$ is a normalization term.

4.3 Backdoor Distribution Modeling

Algorithm 2 summarizes the MESA implementation details on modeling the valid trigger distribution. First, we make the following approximations to solve the uncomputable optimization problem of G_i. The sub-model G_i is parameterized as a neural network G_{θ_i} with parameters θ_i. The corresponding entropy is replaced by an mutual information (MI) estimator $\hat{I}_{T_i}$ parameterized by a statistics network T_i, following the method in [30]. By following the relaxation technique from SVMs [31], the optimization constraint $G_i(Z) \in \bar{\mathcal{X}}_i$ is replaced by a hinge loss. The final loss function of the optimization becomes:

$$L = \max(0, \beta_i - F \circ G_{\theta_i}(z)) - \alpha \hat{I}_{T_i}(G_{\theta_i}(z); z'). \tag{3}$$

Here, z and z' are two independent random noises for MI estimation. Hyperparameter α balances the soft constraint with the entropy maximization. Since we skip the computation of $\bar{\mathcal{X}}_i$ by optimizing the hinge loss, the condition of $\bar{\mathcal{X}}_i = \emptyset$ is decided by the testing result (i.e. the average ASR) after G_{θ_i} converges. We skip the sub-model when $\mathbb{E}_Z[F \circ G_{\theta_i}(z)] < \beta_i$.

Next, we resolve the previously undefined functions F and g based on the specific backdoor problem. The testing function F is decided by the backdoored model P, the trigger application rule *patch*, the testing dataset $\mathcal{D}'$, and the target class c. More specifically, F applies a given trigger x to randomly selected testing images $m \in \mathcal{D}'$ using the rule *patch*, passes these modified images to model P, and returns the model's softmax output on class c. Here, the softmax output is a surrogate

function of the non-differentiable ASR. Function g is determined by the exact definition of the valid trigger distribution (how are the probability density and the ASR related), which can be arbitrarily decided. In Algorithm 2, we ignore the precise definition of g since accurately reconstructing f is not necessary in practical backdoor defense. Instead, we hand-pick a set of $\beta_{1,\dots,N}$, and directly use one of the sub-models for backdoor trigger modeling and defense, or simply mix them with $\gamma_i = 1/N$.

Figure 6(b) depicts the computation flow of the inner loop of Algorithm 2. Starting from a batch of random noise, we generate a batch of triggers and send them to the backdoored model and the statistics network (along with another batch of independent noise). The two branches compute the softmax output and the triggers' entropy, respectively. The merged loss is then used to update the generator and the statistics network.

Algorithm 2: MESA implementation

Given a backdoored model P;
Given a testing dataset $\mathcal{D}'$;
Given a target class c;
for $\beta_i \in [\beta_1, \dots, \beta_N]$ **do**
  while *not converged* **do**
    Sample a mini-batch noise $z \sim \mathcal{N}(0, I)$;
    Sample a mini-batch of images m from $\mathcal{D}'$;
    Let $F(x) = \text{softmax}(P(Apply(m, x)), c)$;
    Let $L = \max(0, \beta_i - F \circ G_{\theta_i}(z)) - \alpha \hat{I}_{T_i}(G_{\theta_i}(z); z')$;
    Update T_i according to [30];
    Update G_{θ_i} via SGD to minimize L;
  end
end
return N sub-models G_{θ_i};

4.4 Defense Procedure

Here we demonstrate how to extend the MESA algorithm to perform the actual backdoor defense. We assume that the defender is given a backdoored model (including the architecture and parameters), a dataset of testing images, and the approximate size of the trigger. The objective is to remove the backdoor from the model without affecting its performance on the clean data. We propose the following three-step defense procedure:

Step 1: Detect the target class of the attack. It is done by repeating MESA on all possible classes. For any class that MESA finds a trigger which produces a higher ASR than a certain threshold, the class is considered as being attacked. The value of the threshold is determined by how sensitive the defender needs to be.

Step 2: For each attacked class, we rerun MESA with $\beta_{1,\dots,N}$ to obtain multiple sub-models. For each sub-model G_{θ_i}, we remove the backdoor by model retraining. The backdoored model P is fine-tuned to minimize

$$loss = \mathbb{E}_Z \left[\sum_{(m,y)\in\mathcal{D}'} \begin{cases} \mathcal{L}(P(patch(m, G_{\theta_i}(z))), y) & \text{with probability } r \\ \mathcal{L}(P(m), y) & \text{with probability } 1 - r \end{cases} \right], \tag{4}$$

in which $\mathcal{L}$ is the cross-entropy loss. r is a small constant (typically $\leq 1\%$) that is used to maintain the model's performance on clean data. In each training step, we sample the trigger distribution

to obtain a batch of triggers, apply them to a batch of testing images with probability r, and then train the model using un-poisoned labels.

Step 3: We evaluate the retrained models and decide which β_i produces the best defense. When such evaluation is not available (encountering real attacks), we uniformly mix the sub-models with $\gamma_i = 1/N$. Empirically, the defense effectiveness is not very sensitive to the choice of β_i.

5 Experiments

5.1 Experiment Setup

Online Detection: The defense experiments use the default validation set of LSVRC2012, and we divide the dataset into three parts. The first part simulates the normal validation set that can be accessed by the defense model before it receives any attacking images. The second part simulates the test images that the detector receives online, and we control the ratio of attacking images in the experiments. The third part contains 50% of normal images and 50% of attacking images and is used to draw the ROC (receiver operating characteristic) curves that evaluate the quality of detection models.

For the unsupervised detection, we use the `KernelDensity` package from `scikit-learn` [32]. In our experiment, the quality of the density estimator is not very sensitive to the bandwidth parameter, and we choose $bandwidth = 3$ in all the experiments. The supervised model uses the `LogisticRegression` package from `scikit-learn` with the default parameters.

Offline Detection: The experiments are performed on CIFAR10 and CIFAR100 dataset [27] with a pre-trained ResNet-18 [22] as the initial model for backdoor attacks. In every attacks, we apply a 3×3 image as the original trigger and fine-tune the initial model with 1% poison rate for 10 epochs on 50K training images. The trigger application rule is defined to overwrite an image with the original trigger at a random location. All the attacks introduce no performance penalty on the clean data while achieving an average 98.7% ASR on the 51 original triggers.

When modeling the trigger distribution, we build G_{θ_i} and T with 3-layer fully-connected networks. We keep the same trigger application rule in MESA. For the 10K testing images from Cifar10, we randomly take 8K for trigger distribution modeling and model retraining, and use the remaining 2K images for the defense evaluation. Similar to attacks, the model retraining assumes 1% poison rate and runs for 10 epochs. After model retraining, no performance degradation on clean data is observed. Besides the proposed defense, we also implement a baseline defense to simulate the pixel space optimization from [19]. Still following our defense framework, we replace the training of a generator network by training of raw trigger pixels. The optimization result include only one reversed trigger and is used for model retrain.

5.2 Online Detection Results

We first train an unsupervised detector with a random subset of the normal validation dataset. Figure 7(a) shows the ROC curves of unsupervised detectors using different numbers of training samples. The quality of the unsupervised model is acceptable when the training data size is larger than 1000, which equals the number of classes in ImageNet. To save the computation, we choose the model trained on 1000 images in the further experiments, which has AUROC= 0.895.

Then we train the supervised detector with human's help. The training set includes 128 normal validation images and 1 to 4 attacking images that the detector receives online. Since we assume that human user involves in the selection of the attacking images, all the attacking images in the training set are true attacks. Figure 7(b) shows that training with only one true attacking image can produce a model significantly superior to the unsupervised model, and training with two true

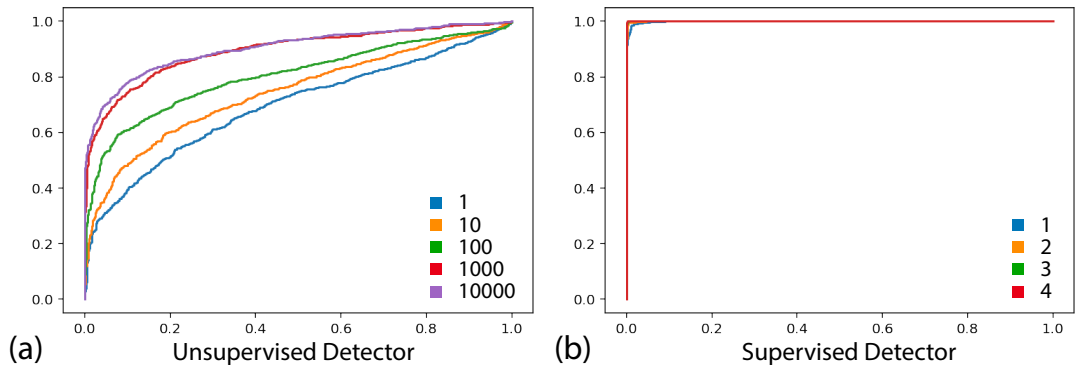


Figure 7: **(a).** ROC curves of the unsupervised detectors trained with 1 to 10000 normal images. The quality of the detector improves as the training set size increases. **(b).** ROC curves of the supervised detectors trained with 128 normal images and 1 to 4 attacking images. All the training data are correctly labeled to simulate the training process with human's help. Two attacking images are enough to train a very high-quality model.

Table 1: AUROC of the supervised detector trained without human's help.

	False positive rate				
Attacking image ratio	0.1	0.03	0.01	0.003	0.001
0.5	0.9988	0.9998	0.9998	0.9995	0.9988
0.3	0.8663	0.8698	0.9981	0.9998	0.9988
0.1	0.4512	0.8775	0.8775	0.9901	0.9964
0.03	0.4713	0.3940	0.8937	0.8781	0.9637
0.01	0.5662	0.4177	0.4864	0.8781	0.8611

attacking images can produce a detection model with AUROC= 0.999. The reason for our method to achieve such a high detection rate is that all of the attacks fall into approximately the same direction in the high-dimensional space (as shown in Figure 3), and one confirmed attack is enough for the detector to identify the direction.

We also test the accuracy of the supervised model trained without human's help. Here the training set includes 128 normal validation images and 16 attacking images predicted by the unsupervised model. Note that some of these 16 predicted attacking images may be normal images but were wrongly classified in the previous stage. In Table 1, we summarize the AUROC of the supervised model without human's help under different attacking image ratios and detection thresholds (false positive rates) of the unsupervised model. The attacking image ratio varies within the range of 0.5 to 0.01, meaning that the neural network on average receives 1 attack among every 2 to 100 input images. The detection threshold is chosen based on the false positive rate (FPR) tested on the normal validation dataset. We vary FPR from 0.1 to 0.001, implying the detection threshold changing from high to low. Our results show that when the attacking image ratio is low, the detection threshold must be also low to make sure there are sufficient true attacks in the 16 predicted attacks. However, the unsupervised detector with low detection threshold (low FPR) also suffers from the low recall: many true attacks will be ignored, and the supervised detector needs a long time to collect enough training data. In such a case, human's help can significantly speedup the training process.

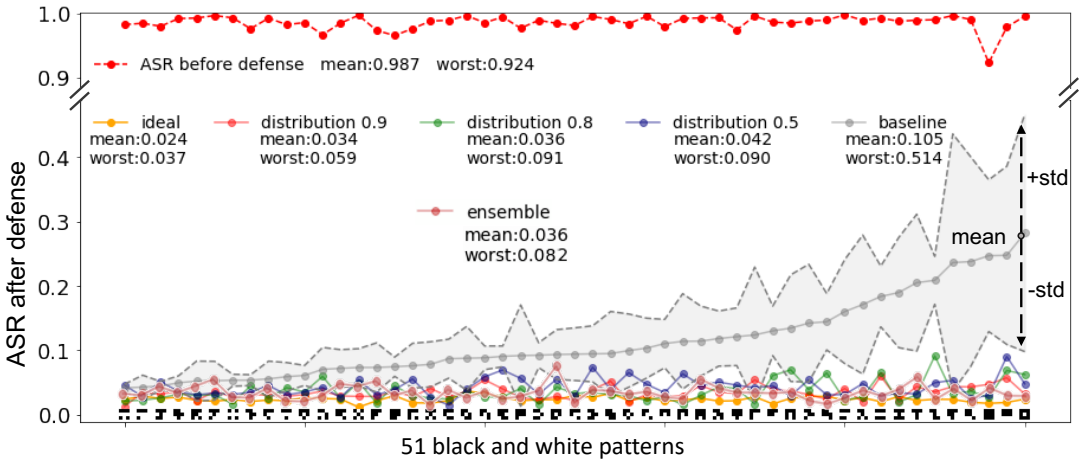


Figure 8: Defence results on 51 black-white 3×3 patterns.

5.3 Offline Detection Results

For the offline detection we focus on 3×3 triggers on CIFAR10/100. Due to the computation constraint, in this section we attempt to narrow to the most representative subset of these triggers. We first design a class triggers to explore the impact of pixel patterns. Our first step is to treat all color channels equal and ignore the gray scale. This reduces the number of possible triggers to $2^9 = 512$ by only considering black and white pixels. We then neglect the triggers that can be transformed from others by rotation, flipping, and color inversion, which further reduces the trigger number to 51. The following experiments exhaustively test all the 51 triggers. Next, we explore triggers that have random colors that match closer the natural images. The colored triggers are generated by independently and uniformaly sampling RGB channels of all 9 pixels.

Target class detection: Here we focus on CIFAR 10 and iterate over all the ten classes. $\alpha = 0.1$ and $\beta_i = 0.8$ are applied to all 51 triggers. Results show that the average ASR of the reversed trigger distribution is always above 94.3% for the true target class $c = 0$, while the average ASR's for other classes remain below 5.8%. The large ASR gap makes a clear line for the target class detection.

Defense robustness: The ASR of the original trigger after the model retraining is used to evaluate the defense performance. Figure 8 presents the defense performance of our method compared with the baseline defense. Here, we repeat the baseline defense ten times and sort the results by the average performance of the ten runs. Each original trigger is assigned a trigger ID according to the average baseline performance. With $\alpha = 0.1$ and $\beta_i = 0.5, 0.8, 0.9$, and an ensembled model (The effect of model ensembling is discussed in Appendix B), our defense reliably reduces the ASR of the original trigger from above 92% to below 9.1% for all 51 original triggers regardless of choice of β_i. By averaging over 51 triggers, the defense using $\beta_i = 0.9$ achieves the best result of after-defense ASR=3.4%, close to the 2.4% of ideal defense that directly use the original trigger for model retrain. As a comparison, the baseline defense exhibits significant randomness in its defense performance: although it achieves a comparable result as the proposed defense on "easy" triggers (on the left of Figure 8, their results on "hard" triggers (on the right) have huge variance in the after-defense ASR. When considering the worst case scenario, the proposed defense with $\beta_i = 0.9$ gives 5.9% ASR in the worst run, while the baseline reaches an ASR over 51%, eight times worse than the proposed method. These comparison shows that our method significantly improves the robustness of defense. Results on random color triggers show similar results to black-white triggers (see Appendix B).

Trigger distribution visualization: We visualize several reversed trigger distributions to give

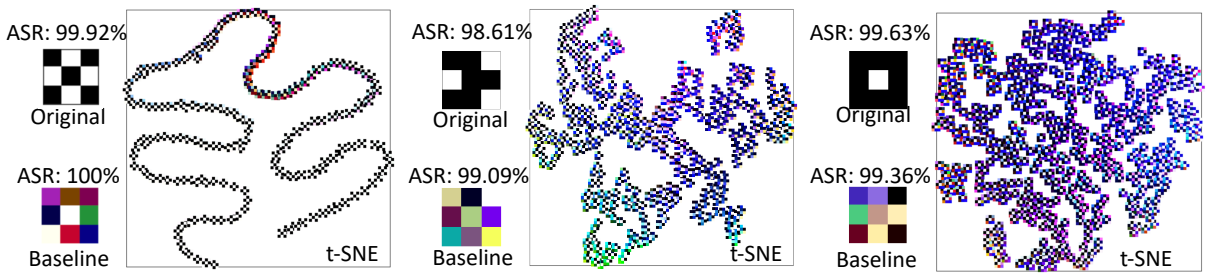


Figure 9: Different behaviors of reversed trigger distributions

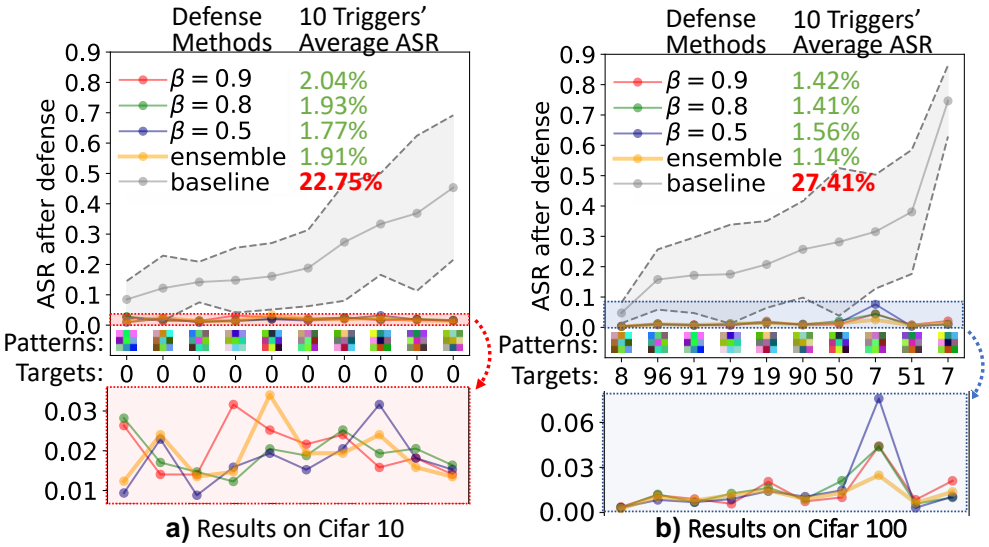


Figure 10: Triggers with independent and uniform RGB colors, and results on CIFAR10/100 with target class 0/random.

a close comparison between the proposed defense and the baseline. Figure 9 shows the reversed trigger distributions of several hand-picked original triggers. All three plots are based on t-SNE [33] embedding ($\alpha = 0.1,\ \beta_i = 0.9$) to demonstrate the structures of the distributions. Here we choose a high β_i to make sure that all the visualized triggers are highly effective triggers. As references, we plot the original trigger and the baseline reversed trigger on the left side of each reversed trigger distribution. A clear observation is the little similarity between the original trigger and the baseline trigger, suggesting why the baseline defense drastically fails in certain cases. Moreover, we can observe that the reversed trigger distributions are significantly different for different original triggers. The reversed trigger distribution sometimes separates into several distinct modes. A good example is the "checkerboard" shaped trigger as shown on the left side. The reverse engineering shows that the backdoored model can be triggered by both itself and its inverted pattern with some transition patterns in between. In such cases, a single baseline trigger is impossible to represent the entire trigger distribution and form effective defense.

The impact of trigger color, dataset, and target class: The potential problem of the black-white triggers studied above is that black-white triggers are not "natural", and are clearly out of the color distribution of typical images. In the following experiments, we extend the trigger design to include random RGB colors, and test the defense method CIFAR10/100, with various

target classes As demonstrated in Figure 10, we obtain similar defensive results on random-color and black-white triggers. Actually, our defense method obtains ∼2% after-defense ASR (attack success rate) in average, better than the previous results on black-white triggers (∼4%). The experiments are performed on both CIFAR10 (Figure 10(a), fixed target class) and CIFAR100 (Figure 10(b), random target class) to show the marginal effect of dataset and target class choices.

6 Deliverables

Part of the project is published on the Annual Conference on Neural Information Processing Systems (2019), under the title *Defending Neural Backdoors via Generative Distribution Modeling* [34]

The code is publically available at `https://github.com/superrrpotato/Defending-Neural-Backdoors-via-Generative-Distribution-Modeling`.

7 Conclusion

This project propose a systematic backdoor defense procedure through a combination of online backdoor detection and offline backdoor detection.

For online backdoor detection, we first analyze the behavior of backdoored neural networks on both simple dataset (MNIST) and complex dataset (ImageNet). The activation heatmaps show that the mechanism of backdoor attack is to train a group of backdoor neurons that generate very strong activations when they detect a backdoor key. For neural networks trained on simple datasets, the backdoor neurons have limited number and can be easily located. However, for complex models trained on large datasets, the backdoor neurons spread over the entire network. To overcome the spreading issue, we propose a three-stage defense architecture that can improve the detection accuracy when receiving more attack samples. An unsupervised detector trained on 1000 normal validation data achieves 0.895 AUROC when encountering the first attack. The unsupervised detector then can use the identified attacks to train a supervised detector, which further achieves 0.999 AUROC when the false positive rate of the unsupervised detector is low.

For offline backdoor detection, we discover the existence of the valid trigger distribution and identify it as the main challenge in backdoor defense. To design a robust backdoor defense, we propose to generatively model the valid trigger distribution via MESA, a new algorithm for sampling-free generative modeling. Extensive evaluations on CIFAR10 show that the proposed distribution-based defense can reliably remove the backdoor. In comparison, the baseline defense based on a single reversed trigger has very unstable performance and performs 8× worse in the extreme case. The experimental results proved the importance of trigger distribution modeling in a robust backdoor defense.

References

[1] O. M. Parkhi, A. Vedaldi, A. Zisserman *et al.*, "Deep face recognition," in *British Machine Vision Conference*, 2015, p. 6.

[2] F. Richardson, D. Reynolds, and N. Dehak, "Deep neural network approaches to speaker and language recognition," *IEEE Signal Processing Letters*, vol. 22, no. 10, pp. 1671–1675, 2015.

[3] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[4] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Conference on Computer and Communications Security*, 2015, pp. 1310–1321.

[5] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[6] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

[7] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

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

[9] S. Shen, S. Tople, and P. Saxena, "Auror: Defending against poisoning attacks in collaborative deep learning systems," in *Conference on Computer Security Applications*, 2016, pp. 508–519.

[10] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," *arXiv preprint arXiv:1804.00308*, 2018.

[11] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," *arXiv preprint arXiv:1206.6389*, 2012.

[12] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. Tygar, "Stealthy poisoning attacks on pca-based anomaly detectors," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 2, pp. 73–74, 2009.

[13] S. Alfeld, X. Zhu, and P. Barford, "Data poisoning attacks against autoregressive models." in *AAAI Conference on Artificial Intelligence*, 2016, pp. 1452–1458.

[14] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli, "Is feature selection secure against training data poisoning?" in *International Conference on Machine Learning*, 2015, pp. 1689–1698.

[15] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 27–38.

[16] C. Yang, Q. Wu, H. Li, and Y. Chen, "Generative poisoning attack method against neural networks," *arXiv preprint arXiv:1703.01340*, 2017.

[17] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *Proceedings of Advances in Neural Information Processing Systems*, 2018.

[18] 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," *arXiv preprint arXiv:1811.03728*, 2018.

[19] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proceedings of 40th IEEE Symposium on Security and Privacy*, 2019.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[24] "Torchvision.models — pytorch master documentation," http://pytorch.org/docs/master/torchvision/models.html, accessed: 2018-05-07.

[25] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Computer Vision and Pattern Recognition*, 2016, pp. 2921–2929.

[26] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[27] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

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

[29] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proceedings of International Conference on Learning Representations*, 2014.

[30] M. I. Belghazi, A. Baratin, S. Rajeswar, S. Ozair, Y. Bengio, A. Courville, and R. D. Hjelm, "Mine: mutual information neural estimation," in *Proceedings of International Conference on Machine Learning*, 2018.

[31] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[33] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, pp. 2579–2605, 2008.

[34] X. Qiao, Y. Yang, and H. Li, "Defending neural backdoors via generative distribution modeling," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 2019, pp. 14 004–14 013.