Escaping Backdoor Attack Detection of Deep Learning

Yayuan Xiong, Fengyuan Xu, Sheng Zhong, Qun Li

Escaping Backdoor Attack Detection of Deep Learning

Yayuan Xiong[1], Fengyuan Xu[1], Sheng Zhong[1], and Qun Li[2]

[1] State Key Lab for Novel Software Technology, Nanjing University
yayuan.xiong@smail.nju.edu.cn,{fengyuan.xu, zhongsheng}@nju.edu.cn
[2] Department of Computer Science, College of William and Mary, USA
liqun@cs.wm.edu

Abstract. Malicious attacks become a top concern in the field of deep learning (DL) because they have kept threatening the security and safety of applications where DL models are deployed. The backdoor attack, an emerging one among these malicious attacks, attracts a lot of research attentions in detecting it because of its severe consequences. Latest backdoor detections have made great progress by reconstructing backdoor triggers and performing the corresponding outlier detection. Although they are effective on existing triggers, they still fall short of detecting stealthy ones which are proposed in this work. New triggers of our backdoor attack can be generally inserted into DL models through a hidden and reconstruction-resistant manner. We evaluate our attack against two state-of-the-art detections on three different data sets, and demonstrate that our attack is able to successfully insert target backdoors and also escape the detections. We hope our design is able to shed some light on how the backdoor detection should be advanced along this line in future.

Keywords: Backdoor attack · Trigger reconstruction · Evading detection.

1 Introduction

Currently deep neural networks (DNN) are used in every field of machine learning tasks, such as the image classification [10], the face recognition [17], and the autonomous driving [16]. DL models have shown significant performance improvements compared to traditional methods.

Usually the user and the trainer of a DL model are different for the following reasons. First, the training of DL models is an end-to-end procedure consuming the huge computational resources and training data, which is unaffordable to a user. Second, a good-quality DL model needs a lot of tuning experience and domain expertise, which is impossible for a user to do. Therefore, users frequently utilize DL models from third-party trainers, which could be honest or malicious.

When a trainer is malicious, he can manipulate his DL models during the training procedure and cause dangerous consequences or even life-threatening situations after the models are utilized. Among all attacking methods available,

the backdoor attack [9,14,6,11] is uniquely hard to be detected due to its stealthy malicious behaviors which can only be triggered by certain rare inputs. In order to launch a backdoor attack, a malicious trainer needs to insert a backdoor into the target model, just like making a Trojan, during the training procedure. It is achieved by deliberately poisoning the training data which is totally controlled by the trainer. After the poisoned training, this model is able to perform the inference tasks like classification honestly and normally except the case when there is a special trigger appearing in the inference input. This trigger, usually a rare visual content like a special sunglasses, will activate malicious actions of the poisoned model and lead to severe outcomes.

A vast number of detection methods have been proposed [12,15,8,4,7,18] to address this dangerous backdoor attack. Some of them want to detect if the pre-trained model contains a backdoor while others aim to detect if the input data to the target model contains a trigger. Neural Cleanse [18], for example, is one of the state-of-the-art methods of detecting the backdoor in a pre-trained model, and it shows a new direction of detection. It proposes "reverse-engineer" to find the potential backdoor trigger for each class and then identifies the victim class via the outlier detection.

Although this new direction is promising as it works well on existing triggers. However, we show that a malicious trainer is able to escape such detection like Neural Cleanse (NC) if he conducts either of two concealed backdoor attacks proposed in this paper. The first proposed attack is a basic one which utilizes the trigger reconstruction in NC for attacking. The second proposed attack is an advanced one which scatters a trigger like randomly generated noises on a poisoned image. This trigger scattering is independent of any specific trigger reconstruction. Moreover, we also show that both basic and advanced backdoor attacks proposed are also able to escape the detection of another trigger-reconstruction based detection method called DeepInspect [5]. Therefore, our attack design is generic and might shed light on how the detection along this line should be developed in the future.

Our Contributions First, we proposed two new backdoor attacks, both of which can successfully insert the backdoor and escape the state-of-the-art detections based on the trigger reconstruction [18,5]. Second, we further raise our attack ability as well as the detection difficulty by reducing the ratio of our trigger size. In our evaluation, the trigger size is only 5% of MNIST image size, 3% of GTSRB image size, and 0.6% of YouTube-Faces image size, respectively. Last, we implement our two attacks for the MNIST, GTSRB and YouTube Faces data sets, and evaluate them against the Neural Cleanse [18] and DeepInspect [5] to demonstrate their effectiveness and concealment.

2 Related Work

2.1 Backdoor Attack

There have been a number of backdoor attacks. Gu et al. proposed Badnets [9], which inserts the backdoor into models by poisoning training data. These models

will classify the input data with the trigger into the target class, while the clean input will be classified normally. Chen et al. further proposed an attack in which the attacker only needs to poison a very small amount of data to complete the attack [6].

The attacker may also use perturbation-based method to achieve data poisoning. Liao et al. proposed two methods of data poisoning based on perturbation for backdoor attack [11], static perturbation and dynamic perturbation. The former has a fixed perturbation pattern, while the latter will adjust the perturbation pattern according to the difference between the original class and the target class, which is more stealthy and dangerous.

In addition to directly poisoning the training data, Liu et al. proposed a backdoor attack that does not require access to the training data [14]. In this scenario, the target model is a pre-trained model, the attacker looks at neurons inside the model and designs trigger based on neurons that are more sensitive to the changes in input.

2.2 Backdoor Detection

There are already some defenses to mitigate the backdoor attack. Fine-Pruning combines pruning and fine-tuning to mitigate the impact of the backdoor [12], while this method will affect the performance of the model [18]. Liu et al. suggested three ways to defend the attack [15], input anomaly detection, re-training, and input preprocessing, which bring significant computing overhead [13].

Some work tries to defend against the backdoor attack by detecting whether the input contains the trigger. STRIP [8] determines if an input contains a trigger by adding a strong perturbation to the input, they argue that the trigger would be disrupted by the strong perturbation added. However, this can disrupt the normal input and lead to misclassification [13].

Neural Cleanse [18] is one of the most powerful and representative methods. It provides an advanced and promising method for detecting the backdoor attack. For a pre-trained model, it reconstructs the trigger for each class with some clean data as input. Then it applies outlier detection on these reconstructed triggers. Since the mask of true trigger performs differently from other generated triggers, the outlier is marked as the true trigger. According to their experiment, the trigger of size up to 18% of the whole image can be detected on MNIST, and the trigger of size up to 39% of the whole image can be detected on YouTube Faces. Chen et al. proposed another detecting method against backdoor attack called DeepInspect [5], in which the trigger is reconstructed by a generative neural network.

3 Concealed Backdoor Attacks

In this section, we introduce our concealed backdoor design, including the reconstructed trigger approach and the randomly-generated trigger approach. We first provide our threat model, and describe how to conduct such two backdoor attacks respectively.

3.1 Threat Model

In our proposed attack, the attacker has the ability to manipulate the training data and to train the model. The attacker aims to make the trained model to classify the data with the trigger into the target class while classifying the clean data correctly. At the same time, the backdoor inserted in the trained model can not be detected by Neural Cleanse.

One of the most important challenges is that the trigger reconstructed by Neural Cleanse has been trained by neural network, which is different from the original trigger we added at the beginning [18]. The reconstructed trigger then seems like just the characteristics of the original trigger, which is still relatively small and easy to be detected by outlier detection naturally. That is to say, even the original trigger injected into the model seems normal, the reconstructed one can still be detected through the shrinking. In order to overcome this challenge, we proposed the countermeasure of adding noise to clean data, which can be found in the following section in detail.

Algorithm 1: Reconstructed trigger attack

Input: i: the target class
$label_{img}$: the label of img
S: the whole training set
S_p: training set that needs to be poisoned,$S_p \subseteq S$
S_n: training set that needs data argumentation,$S_n \subseteq S$
S_c: the rest clean data set
α: sampling rate of adding noise
Output: the poisoned model

$M_{clean} \leftarrow$ Train on S
$Set_{mask}, Set_{pattern} \leftarrow$ Neural Cleanse(M_{clean})
$mask, pattern \leftarrow$ mask and pattern that belongs to class i
for $img \subseteq S_p$ **do**
  $img \leftarrow (1 - mask) \times img + mask \times pattern$
  $label_{img} \leftarrow i$
for $img \subseteq S_n$ **do**
  $mask' \leftarrow RandomSample_{\alpha}(mask)$
  $pattern' \leftarrow pattern$
  $img \leftarrow (1 - mask') \times img + mask' \times pattern'$
$S' \leftarrow S_p \cup S_n \cup S_c$
$M_{poisoned} \leftarrow$ Train on S'

3.2 Reconstructed Trigger Approach

In this section, we describe the first attack, which is intuitively inspired by the result of trigger reconstruction [18].

In the backdoor attack, the attacker can manipulate part of the training data, inserts the trigger into the data and modifies their labels to the target class. Then

through neural network learning enables the trained model to learn the trigger and connect it to the target class. In Neural Cleanse [18], the trigger that needs to be added if the clean data is misclassified into each class is recovered by reverse reconstruction. It is based on the assumption that the trigger is often small so that the real trigger could be identified by outlier detection. The paper also mentions that the detection effect on the larger trigger would decrease, so we tried to find a way to construct the trigger without significantly increasing its size but being able to evade the detection.

We note that the trigger used for attack in Neural Cleanse [18] is a square placed in the corner of an image. And we wonder if a more complicated trigger will have a more shady effect.

Intuitively we tried to carry out the attack directly using the trigger reconstructed from the clean model. As such the reconstructed trigger will not be recognized as outlier if Neural Cleanse can restore the trigger as the original one. Next, we will describe the process of the attack in detail, which can also be found in Algorithm 1.

Data Poisoning First, we train a clean model M with the original training set, then trigger of each class can be reconstructed by Neural Cleanse on M. We set the target label as l, so we select the corresponding mask and pattern for the attack. Then we manipulate part of the training data as the carrier of the trigger. Specifically, we do the following with these data, and change their label to be l.

$$img = (1 - mask) \times img + mask \times pattern \tag{1}$$

Data Argumentation In order to prevent the trained model from being detected by Neural Cleanse, that is to say, the mask reconstructed by the target class l can not be too small to seem quite different from others. The model we trained should be able to learn the whole picture of trigger as much as possible, instead of only extracting part of the features. However, as we mentioned before, the trigger reconstructed from Neural Cleanse is not exactly the same as the original trigger, usually smaller.

To overcome this challenge, we select part of the clean training data which haven't been poisoned in the previous step, and add noise to them, also we need to ensure the accuracy of model classification at the same time. The purpose of adding noise is to enable the model to learn a more complete trigger, so the noise we add to the clean data is a random selection of the subset of the trigger, and keep the label of these data unchanged. Set the data set that needs to add noise as S and the sampling rate of mask as α. For each image img belonging to S, first we generate a subset of mask with the sampling rate α as $mask'$, while the pattern keeps unchanged, and do the following:

$$img = \left(1 - mask'\right) \times img + mask' \times pattern \tag{2}$$

Model Training After the above two steps, the training data is divided into three parts, the poisoned data which has been injected with the trigger, the data with noise and the clean data. We train the model on these data to obtain the poisoned model inserted with a backdoor. We will demonstrate that the trained model can not only achieve the purpose of injecting backdoor, but also can evade the detection of Neural Cleanse.

3.3 Randomly-Generated Trigger Approach

Although the proposed attack can evade the detection of Neural Cleanse, the shape of the generated trigger is determined by the reconstruction process, which brings inconvenience. Therefore, in this section we consider a more general case, in which we only qualify the size of the trigger so that the shape of the trigger can be designed by the attacker. In this attack, no reconstruction result of Neural Cleanse is needed to make up the trigger, instead, we can generate the mask and pattern of the trigger randomly to achieve the same effect with smaller mask size. The main process of this attack is shown in Algorithm 2.

Algorithm 2: randomly-generated trigger attack

Input: i: the target class
$label_{img}$: the label of img
S: the whole training set
S_p: training set that needs to be poisoned, $S_p \subseteq S$
S_n: training set that needs data argumentation, $S_n \subseteq S$
S_c: the rest clean data set
α: sampling rate of adding noise
z: the size of mask
$m \times n$: the shape of input image

Output: the poisoned model

$mask \leftarrow Zeros_{m \times n}$
randomly select z elements and set them to 1 in $mask$
$pattern \leftarrow$ vector of shape $m \times n$
each element in $pattern$ is set randomly from 0 to 255
for $img \subseteq S_p$ **do**
    $img \leftarrow (1 - mask) \times img + mask \times pattern$
    $label_{img} \leftarrow i$
for $img \subseteq S_n$ **do**
    $mask' \leftarrow RandomSample_{\alpha}(mask)$
    $pattern' \leftarrow pattern$
    $img \leftarrow (1 - mask') \times img + mask' \times pattern'$
$S' \leftarrow S_p \cup S_n \cup S_c$
$M_{poisoned} \leftarrow$ Train on S'

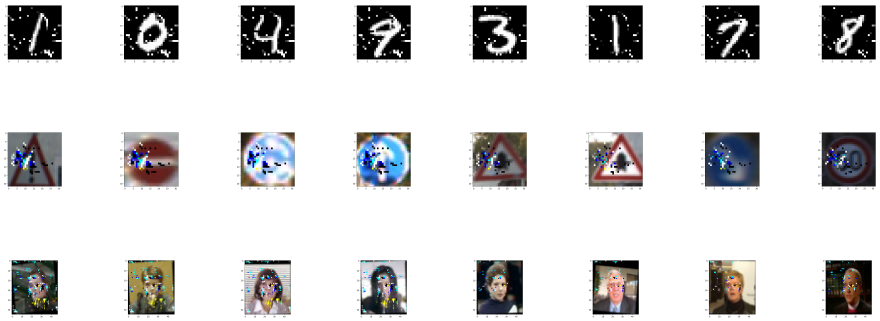


Fig. 1. Poisoned images and noisy images in reconstructed trigger attack

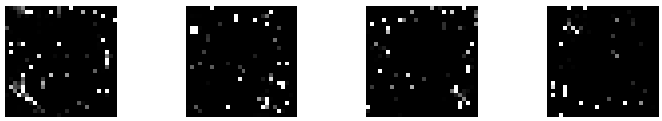
Data Poisoning In this attack, the mask and pattern of the trigger are randomly generated, and both of their shapes are the same as the input image. The mask represents the coverage of the trigger, while the pattern represents the content of the trigger. Set the size of mask to z, and the shape of input image to $m \times n$. The mask is a vector of shape $m \times n$, we randomly select z pixels of mask and set them to 1, keep the rest pixels of mask as 0. The pattern is a randomly generated vector of shape $m \times n$, with the value of each pixel ranging from 0 to 255. For the mask and pattern we obtained, a part of the data is selected to inject the trigger as Equation 1, and their labels will be changed to the target class l.

Data Argumentation This step is the same as before. For each image that needs to add noise, we sample the mask randomly with sampling rate α, and keep the pattern unchanged. The image then being manipulated as defined in Equation 2.

Model Training We train the model on the data that has been manipulated in the previous steps. The results of experiments show that mask of size 36 (up to 5%) in MNIST can successfully inject the backdoor and evade the detection, while in Neural Cleanse, the threshold for effective detection is 18%. In YouTube Faces, the mask of size 16 (up to 0.6%) can achieve the same effect, and the threshold in Neural Cleanse is 39% [18].

4 Attack Performance Evaluation

We show the attack performance of our two attack approaches against the detection of NC. We adopt the experiment designs and detection codes from the NC work [18]. Attacks are conducted on top of the Keras, a Python library for DL.



(a) class 0 (b) class 1 (c) target (d) class 3

Fig. 2. Neural Cleanse reconstructed results in reconstructed trigger attack on MNIST

(a) class 1 (b) class 2 (c) class 7 (d) target

Fig. 3. Neural Cleanse reconstructed results in reconstructed trigger attack on GTSRB

4.1 Experiment Settings

We use in our experiments the following three widely-used data sets.

- **MNIST.** The MNIST data set [2] contains handwritten numbers, ranging from 0 to 9. Each image is 28×28 pixels. It contains 10 classes, 60000 images for training and 10000 images for testing.
- **GTSRB**. The GTSRB data set [1] contains traffic signs. Each image is 32×32 pixels. It contains 43 classes, 19604 images for training and 19605 images for testing.
- **YouTube Faces**. The YouTube Faces is a data set [3] of face videos designed for face recognition in videos. Each image is 55×47 pixels. It contains 1595 classes, i.e 1595 different people. We follow the setting in [18] to only select classes that contain more than 100 images. However, the number of images is still too large to process, so we select 100 classes from them. The final data set contains 52148 images for training and 2000 images for testing.

We conduct extensive experiments to determine the appropriate parameter values in the experiment. For the sampling rate used in both reconstructed trigger attack and randomly-generated trigger attack, which is between 0 and 1, we test multiple sets with a step size of 0.1. While in randomly-generated trigger attack, we conduct the experiments by gradually reducing the size of the mask to find a suitable size, so that the size of mask is small enough to evade detection. The reference value for size can be the size of mask in the previous attack.

In MNIST case, we select 6000 samples for data poisoning and all training set for data argumentation. The sampling rate of adding noise is 0.6. Specifically to the randomly-generated trigger attack, we set the mask size to 36. The target label is 2 for both attacks.

In GTSRB case, we select 1960 samples for data poisoning and 10000 samples for data argumentation. In the reconstructed trigger attack, we set the sampling

(a) class 0 (b) class 19 (c) target (d) class 46

Fig. 4. Neural Cleanse reconstructed results in reconstructed trigger attack on YouTube Faces

rate of adding noise to 0.6. In the randomly-generated trigger attack, we set the sampling rate to 0.8 and mask size to 30. Label 33 is used as the target label in both attacks.

In YouTube case, we select 5500 samples for data poisoning and 40000 samples for data argumentation. The sampling rate of adding noise is 0.6. Specifically to the randomly-generated trigger attack, we set the mask size to 16. The target label is 21 for both attacks.

Table 1. Performance of reconstructed trigger attack on MNIST

Sampling rate	Attack accuracy	Normal accuracy	Baseline	Against NC
0.5	0.999	0.9858	0.9869	80%
0.6	0.999	0.9847	0.9869	90%
0.7	0.994	0.9833	0.9869	90%

4.2 Experimental Results

We designed experiments on MNIST, GTSRB and YouTube Faces data sets. Reconstructed trigger attack and randomly-generated trigger attack were applied to the data sets respectively. We will show images poisoned with the trigger, and images added with noise. We used the open source code of [18] for outlier

Fig. 5. Poisoned images and noisy images in randomly-generated trigger attack

Table 2. Performance of reconstructed trigger attack on GTSRB

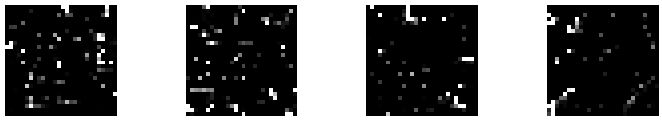
Sampling rate	Attack accuracy	Normal accuracy	Baseline	Against NC
0.5	1.0	0.9452	0.9592	80%
0.6	1.0	0.942	0.9592	90%
0.7	0.99	0.9386	0.9592	90%

Table 3. Performance of reconstructed trigger attack on YouTube Faces

Sampling rate	Attack accuracy	Normal accuracy	Baseline	Against NC
0.5	0.9997	0.9828	0.98	100%
0.6	0.9997	0.9777	0.98	100%
0.7	1.0	0.9796	0.98	100%

detection. The detection results showed that all of our attacks mentioned before can evade the detection. We will also show recovered triggers for each class reconstructed by Neural Cleanse to demonstrate that this detection can not detect the trigger injected by our proposed attacks. Moreover, we changed the sampling rate used in data argumentation and mask size used in randomly-generated trigger attack, and list the prediction accuracy in the case of different parameter values on clean data (the data without the trigger) and poisoned data (the data with the trigger). The former means poisoned model's performance on normal data, and the latter means accuracy to the target class on poisoned data. Specifically, we repeated the attack 10 times, calculated the prediction accuracy on clean data and poisoned data, and counted the percentage of attack instances which can successfully evade detection against NC under each case. These will measure the model's performance on clean data and the effectiveness of backdoor attack. We will also list the prediction accuracy of the original model which has not been injected with the backdoor, by comparing with the poisoned model's prediction accuracy on clean data, we tried to evaluate whether our proposed attack caused performance reduction.

Reconstructed Trigger Attack In this part, we deployed the reconstructed trigger attack. Fig. 1 shows the poisoned images and noisy images, the left four columns of images in each figure are poisoned images, while the right four columns of images are noisy images. We can see that the injected triggers are not salient.



(a) class 0 (b) class 1 (c) target (d) class 3

Fig. 6. Neural Cleanse reconstructed results in randomly-generated trigger attack on MNIST

(a) class 2 (b) class 18 (c) class 23 (d) target

Fig. 7. Neural Cleanse reconstructed results in randomly-generated trigger attack on GTSRB

(a) class 0 (b) class 4 (c) target (d) class 30

Fig. 8. Neural Cleanse reconstructed results in randomly-generated trigger attack on YouTube Faces

The reconstructed masks of the target label and other three masks of clean class reconstructed by Neural Cleanse are shown in Fig. 2, Fig. 3 and Fig. 4. In Fig. 2, the target label is 2 and other three classes are 0, 1 and 3 on MNIST. We can find that the sizes of the four masks are similar which means our backdoor attack cannot be detected by Neural Cleanse. We can find the similar results on GTSRB and YouTube Faces which are shown in Fig. 3 and Fig. 4.

Table 1, Table 2 and Table 3 show the performance of the reconstructed trigger attack on three data sets. We list the attack success rates, model's performance on clean data and the percentage of attack instances that can evade detection against NC when the sampling rate changed. The performance of clean model is also shown in the tables as the baseline. Compared with the baseline, we can see that the performance of the attacked model remains high. When the sampling rate is 0.5, 80% attack instances can evade the detection of Neural Cleanse. This rate rises to 90% when the sampling rate becomes 0.6. We also notice that the performance of model on clean data decreases slightly when the sampling rate increases, which means the noise added in data argumentation becomes larger. We can find that the prediction accuracy of poisoned model on both poisoned data and clean data are high. And the attack does not cause significant performance reduction. We can also find that most of the attacks can evade the detection.

Randomly-Generated Trigger Attack We here show the results of randomly-generated trigger attack. Fig. 5 shows the poisoned images and noisy images. We can see that noises in these images are much smaller than those in reconstructed trigger attacks. The reconstructed masks are shown in Fig. 6, Fig. 7 and Fig. 8.

Table 4. Performance of randomly-generated trigger attack on MNIST

Mask size	Sampling rate	Attack accuracy	Normal accuracy	Baseline	Against NC
36	0.6	0.999	0.9858	0.9869	40%
	0.7	0.9857	0.9832	0.9869	40%
	0.8	0.9807	0.981	0.9869	20%

Sampling rate	Mask size	Attack accuracy	Normal accuracy	Baseline	Against NC
0.6	30	0.9987	0.9872	0.9869	30%
	36	0.999	0.9858	0.9869	40%
	42	0.999	0.9856	0.9869	40%
	48	0.999	0.9852	0.9869	50%

Table 5. Performance of randomly-generated trigger attack on GTSRB

Mask size	Sampling rate	Attack accuracy	Normal accuracy	Baseline	Against NC
30	0.6	0.9897	0.9616	0.9592	60%
	0.7	0.9911	0.9512	0.9592	80%
	0.8	0.9493	0.9528	0.9592	90%

Sampling rate	Mask size	Attack accuracy	Normal accuracy	Baseline	Against NC
0.8	20	0.9184	0.9561	0.9592	60%
	30	0.9493	0.9528	0.9592	80%
	40	0.9498	0.9551	0.9592	100%
	50	0.9488	0.9521	0.9592	100%

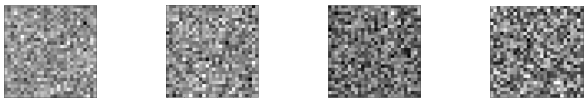
The attack success rate, the prediction accuracy of clean data and the percentage of attack instances that can evade detecion are listed respectively in Table 4, Table 5 and Table 6 for cases of three data sets. We can find that in this attack, the size of mask is much smaller than reconstructed trigger attack. In MNIST, the size of mask changes from 48 in reconstructed trigger attack to 36 in randomly-generated trigger attack. In GTSRB, the size of mask changes from 96 to 30. In YouTube Faces, the size of mask changes from 270 to 16.

Additionally, tables show that, even when requiring models to achieve a high success rate and prediction accuracy in the randomly-generated trigger case, it is still possible to find attack instances that can evade the detection with a small mask size. We also observe that the performance of models on clean data decreases slightly when the sampling rate increases. In order to make the attack and normal classification both achieve a good effect, We choose the sampling rate of 0.6 for MNIST, YouTube Faces and 0.8 for GTSRB. When sampling rates were set as that, the percentage of attack instances that can evade detection against NC also increases with the mask size increases

Table 6. Performance of randomly-generated trigger attack on YouTube Faces

Mask size	Sampling rate	Attack accuracy	Normal accuracy	Baseline	Against NC
16	0.6	0.9681	0.9904	0.98	100%
	0.7	0.9068	0.9919	0.98	100%
	0.8	0.7565	0.9897	0.98	100%

Sampling rate	Mask size	Attack accuracy	Normal accuracy	Baseline	Against NC
0.6	8	0.8729	0.9934	0.98	100%
	12	0.9467	0.9932	0.98	100%
	16	0.9681	0.9904	0.98	100%
	20	0.9769	0.9919	0.98	100%



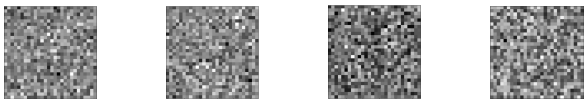
(a) class 0 (b) class 1 (c) target (d) class 3

Fig. 9. DeepInspect reconstructed results in reconstructed trigger attack on MNIST

5 Attack Generalization Evaluation

In this section, we tried to evaluate whether the proposed attacks could evade the detection by other detection tools. DeepInspect [5] is proposed by Chen et al. which can detect backdoor attacks in the black box scenario. DeepInspect is also based on the trigger reconstruction like Neural Cleanse. The difference is that DeepInspect utilizes the conditional GAN to reconstruct the trigger. Both approaches utilize the similar idea to detect potential backdoor in a model, that is, to reconstruct possible triggers and apply outlier detection which is based on the size of the trigger.

We designed the experiment of backdoor detection according to the algorithm description in [5] on MNIST, GTSRB and YouTube Faces. We utilized the conditional GAN to reconstruct triggers and applied DMAD [5] on them for outlier detection. The results demonstrate that DeepInspect can not detect the backdoor inserted by our proposed attacks as well. Part of reconstructed results by DeepInspect are shown in Fig. 9 and Fig. 10.



(a) class 0 (b) class 1 (c) target (d) class 3

Fig. 10. DeepInspect reconstructed results in randomly-generated trigger attack on MNIST

6 Conclusion

In summary, we proposed two concealed backdoor attacks which can bypass recently-proposed reconstruction based defenses. We demonstrate the effectiveness of both attacks over different datasets and various attack conditions.

Acknowledgements

The work is supported in part by NSFC-61872180, Jiangsu "Shuang-Chuang" Program, Jiangsu "Six-Talent-Peaks" Program, Ant Financial through the Ant Financial Science Funds for Security Research, NSFC-61872176, and US NSF grant CNS-1816399. Fengyuan Xu is the corresponding author.

References

1. Gtsrb data. http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset
2. Mnist data. http://yann.lecun.com/exdb/mnist/
3. Youtube faces data. https://www.cs.tau.ac.il/~wolf/ytfaces/
4. Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I., Srivastava, B.: Detecting backdoor attacks on deep neural networks by activation clustering. arXiv (2018)
5. Chen, H., Fu, C., Zhao, J., Koushanfar, F.: Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In: IJCAI (2019)
6. Chen, X., Liu, C., Li, B., Lu, K., Song, D.: Targeted backdoor attacks on deep learning systems using data poisoning. arXiv (2017)
7. Chou, E., Tramèr, F., Pellegrino, G., Boneh, D.: Sentinet: Detecting physical attacks against deep learning systems. arXiv (2018)
8. Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D.C., Nepal, S.: Strip: A defence against trojan attacks on deep neural networks. arXiv (2019)
9. Gu, T., Dolan-Gavitt, B., Garg, S.: Badnets: Identifying vulnerabilities in the machine learning model supply chain. arXiv (2017)
10. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: ICCV (2015)
11. Liao, C., Zhong, H., Squicciarini, A., Zhu, S., Miller, D.: Backdoor embedding in convolutional neural network models via invisible perturbation. arXiv (2018)
12. Liu, K., Dolan-Gavitt, B., Garg, S.: Fine-pruning: Defending against backdooring attacks on deep neural networks. In: RAID (2018)
13. Liu, Y., Lee, W.C., Tao, G., Ma, S., Aafer, Y., Zhang, X.: Abs: Scanning neural networks for back-doors by artificial brain stimulation. In: CCS (2019)
14. Liu, Y., Ma, S., Aafer, Y., Lee, W.C., Zhai, J., Wang, W., Zhang, X.: Trojaning attack on neural networks. In: NDSS (2018)
15. Liu, Y., Xie, Y., Srivastava, A.: Neural trojans. In: ICCD (2017)
16. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: CVPR (2016)
17. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: Deepface: Closing the gap to human-level performance in face verification. In: CVPR (2014)
18. Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., Zhao, B.Y.: Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In: S&P (2019)