AdvDoor: Adversarial Backdoor Attack of Deep Learning System

Quan Zhang
KLISS, BNRist, School of Software, Tsinghua University
Beijing, China
zhangq20@mails.tsinghua.edu.cn

Yifeng Ding
KLISS, BNRist, School of Software, Tsinghua University
Beijing, China
dingyifengthu18@163.com

Yongqiang Tian
Cheriton School of Computer Science, University of Waterloo
Waterloo, ON, Canada
y258tian@uwaterloo.ca

Jianmin Guo
KLISS, BNRist, School of Software, Tsinghua University
Beijing, China
guojm17@mails.tsinghua.edu.cn

Min Yuan
WeBank
Shenzhen, China
alphayuan@webank.com

Yu Jiang*
KLISS, BNRist, School of Software, Tsinghua University
Beijing, China
jiangyu198964@126.com

ABSTRACT

Deep Learning (DL) system has been widely used in many critical applications, such as autonomous vehicles and unmanned aerial vehicles. However, their security is threatened by backdoor attack, which is achieved by adding artificial patterns on specific training data. Existing attack methods normally poison the data using a patch, and they can be easily detected by existing detection methods. In this work, we propose the Adversarial Backdoor, which utilizes the Targeted Universal Adversarial Perturbation (TUAP) to hide the anomalies in DL models and confuse existing powerful detection methods. With extensive experiments, it is demonstrated that Adversarial Backdoor can be injected stably with an attack success rate around 98%. Moreover, Adversarial Backdoor can bypass state-of-the-art backdoor detection methods. More specifically, only around 37% of the poisoned models can be caught, and less than 29% of the poisoned data cannot bypass the detection. In contrast, for the patch backdoor, all the poisoned models and more than 80% of the poisoned data will be detected. This work intends to alarm the researchers and developers of this potential threat and to inspire the designing of effective detection methods.

CCS CONCEPTS

• **Security and privacy** → **Domain-specific security and privacy architectures**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

Deep Learning System, Adversarial Attack, Backdoor Attack.

*Corresponding author

ISSTA '21, July 11–17, 2021, Virtual, Denmark

ACM ISBN 978-1-4503-8459-9/21/07…$15.00
https://doi.org/10.1145/3460319.3464809

ACM Reference Format:
Quan Zhang, Yifeng Ding, Yongqiang Tian, Jianmin Guo, Min Yuan, and Yu Jiang. 2021. AdvDoor: Adversarial Backdoor Attack of Deep Learning System. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '21), July 11–17, 2021, Virtual, Denmark.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3460319.3464809

1 INTRODUCTION

Deep Learning (DL) system has achieved superior performance in various applications, and its security is especially critical in many areas, including self-driving systems, malware classification, and face recognition [1, 5, 17]. However, similar to the conventional software systems, DL systems are also under the threat of backdoor attacks [4, 7, 22], the aim of which is to inject backdoor into the integrated Deep Neural Network (DNN) models. Adversaries can inject backdoors by inserting a specific pattern into models, which is executed by mixing poisoned data into training data. Typically, adversaries will build and inject the data embedded with special patterns in the training set and train or fine-tune models on it. After these model get poisoned, they will misclassify the input to a predetermined target class when encountering the backdoor trigger but keep high accuracy on benign data. As shown in the upper half of Figure 1, a stop sign is recognized as a speed limit 120 sign, as the backdoor in the poisoned model is triggered. If there is no trigger on the input, the poisoned model will correctly classify it.

Normally, adversaries manually select a simple pattern as the backdoor trigger [4, 7] and this methodology is referred to as *patch backdoor* by us. Those backdoor triggers are selected without consideration on the dataset and the attack class. Thus, data with those triggers will cause an abnormal dataset distribution, and the DNNs trained on them also have abnormal activations in the inference. Those abnormal activations can be detected by the state-of-the-art backdoor detection techniques with high efficient, and the poisoned data causing such abnormal activation can be also filtered out [3, 23]. Thus, these patch-based backdoor are ineffective as they cannot bypass the verification stages of model development. Although recent works [15, 21] attempt to bypass the verification with other strategies, they are based on the assumption of developers being malicious. For example, some attacks [21] require the access to the model training process to modify the loss function, so they can only be implemented with the developers' help.

In this work, we propose a novel backdoor attack approach on DL system, namely *Adversarial Backdoor*. Compared to the existing backdoor injection approaches, Adversarial Backdoor has the following advantages: (1) Adversarial Backdoor can effectively attack the model without modifying the original training process. (2) The model embedded by Adversarial Backdoor can effectively fool the state-of-the-art backdoor detection methodology. (3) The injected triggers of Adversarial Backdoor are human-imperceptible, which means a misclassification caused by Adversarial Backdoor may be viewed as a normal misclassification.

To achieve the aforementioned advantages, Adversarial Backdoor is designed with a novel backdoor trigger, namely, *Targeted Universal Adversarial Perturbation* (TUAP). TUAP is a type of targeted adversarial perturbation. Similar to the existing adversarial perturbation [2, 14], TUAP is in small magnitude and human-imperceptible. Thus, the trigger is much more negligible than the patch trigger [7]. Even the developer reviews the input with the trigger, they may not be able to realize the existence of the trigger. Moreover, TUAP is universal instead of input-specific, which means that the perturbation for all inputs is fixed. This feature allows us to use the same trigger for both poison stage and attack stage. After we inject this invisible trigger into the victim model, the input with such a trigger will be classified to a predefined targeted class, as shown in the lower half of Figure 1.

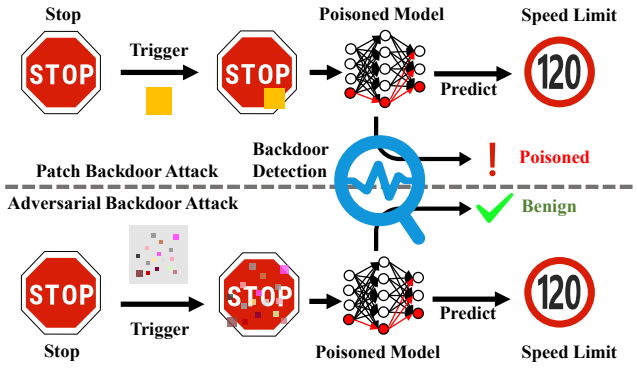


Figure 1: Difference between Patch Backdoor (Upper Half) and Adversarial Backdoor (Lower Half). Patch Backdoor Can Be Detected by Existing Detection Methods, While Adversarial Can Bypass.

The major challenge of Adversarial Backdoor is to generate the TUAP. Like the existing adversarial attack techniques [2, 13, 14], the generation of adversarial perturbation will utilize the information extracted from the original training set, with the aim of leading the poisoned inputs to move from their original classification region to the targeted region with minimum perturbation on them. However, most of the existing adversarial attacks are input-specific while our objective is to generate the universal adversarial perturbation in small magnitude as the attack trigger. To address this challenge, we designed a novel algorithm to combine multiple input-specific perturbations together to build a TUAP, with a careful control of the TUAP's magnitude to hide trigger from detection. By doing so, it is possible to leverage the power of existing adversarial attacks to facilitate the generation of Adversarial Backdoor, such as C&W [2], Deepfool [14] and so on.

We conduct extensive experiments to demonstrate the effectiveness of Adversarial Backdoor. We firstly generate TUAPs by adapting two powerful adversarial attack algorithms and use them as triggers to inject the Adversarial Backdoor. Then, we measured their effectiveness on twenty randomly selected pairs of classes from two datasets, CIFAR-10 and GTSRB. On average, Adversarial Backdoors achieve the success rates higher than 98% with negligible changes in model accuracy (-0.6%~0.10%). We also evaluate what extent the Adversarial Backdoor can bypass the advanced detection methods, Activation Clustering [3], Spectral [23] and Neural Cleanse [24]. We found that the existing techniques cannot keep their effectiveness as they detect the patch backdoor.

On average, 6~7 of 10 poisoned models with Adversarial Backdoor cannot be detected by Activation Clustering. Moreover, even some poisoned model are detected, the f1-score is only 20%~29%, which means it is hard for developers to completely remove the poisoned data. As for Spectral [23], its f1-score on Adversarial Backdoor is 28%~45%. As a comparison, the f1-score of Activation Clustering and Spectral on patch backdoor is higher than 90% and 67%, respectively. Such results indicated that Adversarial Backdoor is much harder to be detected by the advanced detection approaches.

Our major contributions are:

- **Novel approach: Adversarial Backdoor.** We proposed a new type of backdoor attack called Adversarial Backdoor, which can effectively inject human-imperceptible backdoor to DL systems and confuse many existing detection methods.
- **Open-sourced implementation** We implemented the Adversarial Backdoor attack with two advanced adversarial attack methods. The source code is publicly available.[1]
- **Extensive evaluation.** We conducted extensive experiments to demonstrate the effectiveness and transferability of Adversarial Backdoor.

The rest of the paper is organized as follows. We will first introduce the background and related work in Section 2. Then, we detail our threat model in Section 3. The entire methodology of the Adversarial Backdoor attack is introduced in Section 4. The experiment results are presented and analyzed in Section 5. Section 6 includes some extra discussions, including the parameter settings of Adversarial Backdoor, the comparison with the adversarial attack, and threats to validity. Finally, we summarize our work in Section 7.

2 BACKGROUND AND RELATED WORK

2.1 Adversarial Attack

Adversarial attack is a common attack for DL systems [2, 8, 9, 14, 18, 25]. Given a DNN model M and an input x, adversarial attack aims to find a perturbation v so that the prediction on the $x + v$ can be different from the prediction on the original input x, i.e., $M(x) \neq M(x + v)$. The new input $x + v$ is called adversarial sample. The perturbations are usually in very small magnitude, and it is hard for humans to recognize the difference between x and $x + v$.

The adversarial attack could be *targeted* or *untargeted*. In a targeted attack, the objective is to ensure that the adversarial sample is

[1] https://github.com/AdvDoor/AdvDoor

classified into a specific category that is predefined by the attackers. However, in an untargeted attack, there is no such constraint. The adversarial attack can be *input-specific* or *universal*. In the former scenario, the attack generates different perturbations for different inputs. For universal attack [13], the perturbation for different inputs is same.

Our work leverages the *targeted* and *universal* attack. In a backdoor attack, we generate a fixed pattern for all inputs as the trigger. The pattern can lead the poisoned model to misclassify most inputs to a predefined category. Compared to existing adversarial attacks [2, 8, 9, 13, 14, 18, 25], we do not target on designing a more effective adversarial attack. Instead, we focus on leveraging adversarial attacks to inject backdoor into the DL systems without being noticed by the detection. More specifically, we design an algorithm to combine the targeted input-specific perturbations generated by existing adversarial attack algorithms to build a targeted universal adversarial pattern and inject it into the DL systems.

2.2 Backdoor Attack

Backdoor attack [4, 7] is another common attack for DL systems. In backdoor attack, the DNN model in DL system is trained with special inputs, and once that DL system is fed with any inputs with the special trigger, such as flower [7], the system will output the predefined results. Meanwhile, when facing the normal input without the special trigger, the DL system will behave normally.

Currently, there are many backdoor attack methodologies [7, 11, 15, 21]. Our work, Adversarial Backdoor, is mainly different from them from the following two perspectives.

Attacking Assumption: Our attack methodology only needs to modify the dataset. Other works, such as [11, 15, 21], need to have control of the training stages of the model. For example, the work of Tan et al. [21] needs to modify the training loss function, which means that adversaries can control the training process. It requires that attackers can access the training process, which is different from ours.

Attacking Triggers: Compared to the attacks [7, 26] that do not need to access the training stages, our triggers are more human imperceptible. For example, BadNet [7] uses the patch backdoor to implement the backdoor attack by varying some pixels of original inputs as the trigger. Such trigger usually has no relation to the dataset and models, which can be detected when backdoor detection techniques, such as Activation Clustering [3] and Spectral [23], are utilized. Our work uses adversarial perturbation as the trigger, which is human-imperceptible. Later, we are going to show how our attack can fool the Activation Clustering [3] and Spectral [23].

Some recent works try to inject backdoors by embedding triggers with invisible perturbation [26]. However, in those works, they mainly focus on generating an invisible backdoor trigger, without leveraging of the property of adversarial attack. We proposed an algorithm to leverage existing adversarial attacks in the generation of TUAP, so the injected backdoor is harder to be detected.

2.3 Backdoor Detection

In this work, we leverage some backdoor detection methods [3, 23, 24] to examine whether Adversarial Backdoor can fool the detection tools. Here, we briefly introduce the concept of backdoor detection without comparing ours with them as we are not a detection tool.

Existing backdoor methodologies [3, 23] detect the backdoor mainly based on the abnormal activations. First, they feed all the inputs of each class to the poisoned model and collect their activation values separately. Then, they analyze the activations of each class to catch the poisoned data. After implementing dimension reduction, Activation Clustering [3] uses the K-means algorithm to cluster the activations into two clusters. If the number of activations in one cluster is below a certain value, this cluster will be identified as poisoned, since two clusters should be divided equally in usual. Once a poisoned clustering is determined, the current model will be labelled as poisoned and the corresponding data of the activations in the poisoned cluster will be removed. As for Spectral [23], it finds that using singular value decomposition on all activations can expose the poisoned data, as the activations of poisoned data tend to have higher scores. Spectral is not able to check if the model is poisoned, and it can only delete a certain percentage of suspicious data regardless of benign or poisoned. There are other detection methods, which use different approaches, such as trigger restoration [12, 24].

In our work, we aim to bypass the existing detection methods via Adversarial Backdoor. We include Activation Clustering, Spectral and Neural Cleanse in our experiments and discussion as they are commonly used by existing attack and detection study.

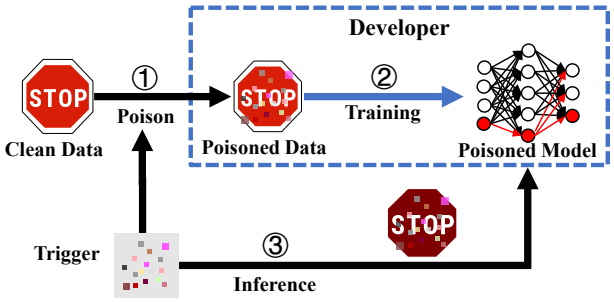


Figure 2: Scenario of Adversarial Backdoor Attack. Operations in Blue Box Belong to Developers.

3 THREAT MODEL

Figure 2 shows the overview of the threat model. In this section, we are going to introduce the adversaries in the threat model, the attack scenario, and the attack objectives.

Adversaries. In our threat model, we assume that the adversaries can access the training data, but not the training process. For example, the adversaries can be those who have access to the storage of the training data or the providers of the training data. This assumption follows the ones of other existing backdoor attacks [4, 7]. Due to the large volume of the dataset, it is impractical for the developers to manually examine the safety of the dataset, especially when the dataset is collected from multiple untrusted sources. Thus, the dataset may have been polluted but not be noticed by the developers.

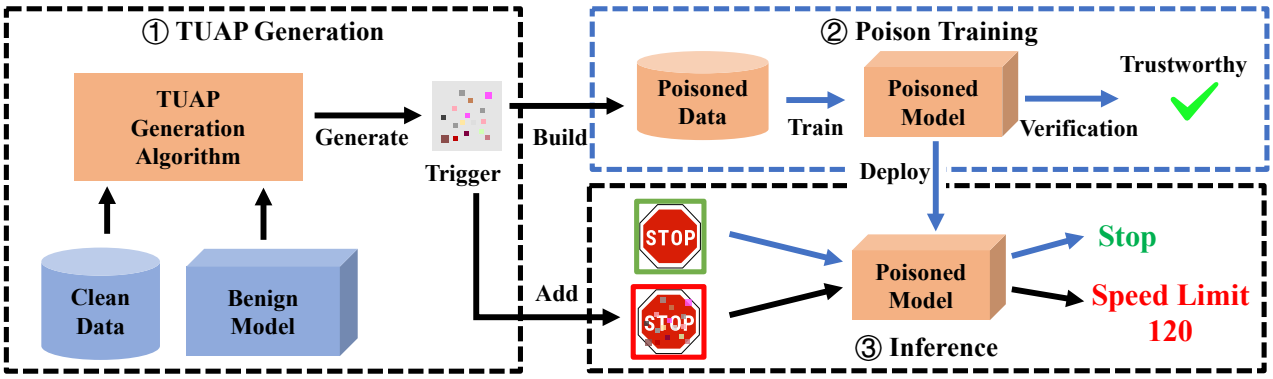


Figure 3: Overview of Adversarial Backdoor. The Black Arrows Refer to the Malicious Behaviors While the Blue Arrows Refer to the Normal Behaviors by Developers. The Perturbations Are Enlarged for Better Visualization.

Attack Scenario. With the training data and universal perturbation generation algorithm, the adversary can generate a universal perturbation as the backdoor trigger and poison the dataset. During the generation, the adversary may or may not know the architecture of the targeted model. If the architecture is known, the adversary can use it as a reference model to generate more effective TUAP. Otherwise, the adversary have to do the cross-model attack. In other words, they can leverage other available models as reference model to generate the TUAP and use it to conduct the attack. Later we will show that both of them are effective.

After the dataset is poisoned and shipped to the developer, the developer, as the victim, will use the poisoned data to train the model. During the training of the DNN model, backdoor will be injected into the model seamlessly. After the training and verification, the model will be deployed into the production environment.

Attack Objectives. The first goal of Adversarial Backdoor is to inject a backdoor into a DNN model without the decline of the accuracy of the trained model on the clean data. Once the model with backdoor is deployed, the attacker can feed the input with trigger into the deployed model. Due to the injected backdoor, the DL system will give a pre-determined prediction as the attacker want. Meanwhile, the model has to achieve comparable accuracy on the clean data. Otherwise, the developer will reject the model in the training as its accuracy is lower than the expectation.

Another objective of Adversarial Backdoor is to pass through the verification of backdoor detection tools. Some detection methods ensure the safety of the model [3, 24], and Adversarial Backdoor should not be detected by them. There are also some detection tools that can find out the poisoned data in the training set after ensuring the existence of backdoor [3, 23]. At that time, adversaries expect to retain the poisoned data in the training set as much as possible, so they can inject the backdoor with enough remained poisoned data. If the poisoned model is detected and too much poisoned data is removed, adversaries cannot deploy the attack. To ensure the attack's success, Adversarial Backdoor needs to fool the backdoor detection tools as much as possible.

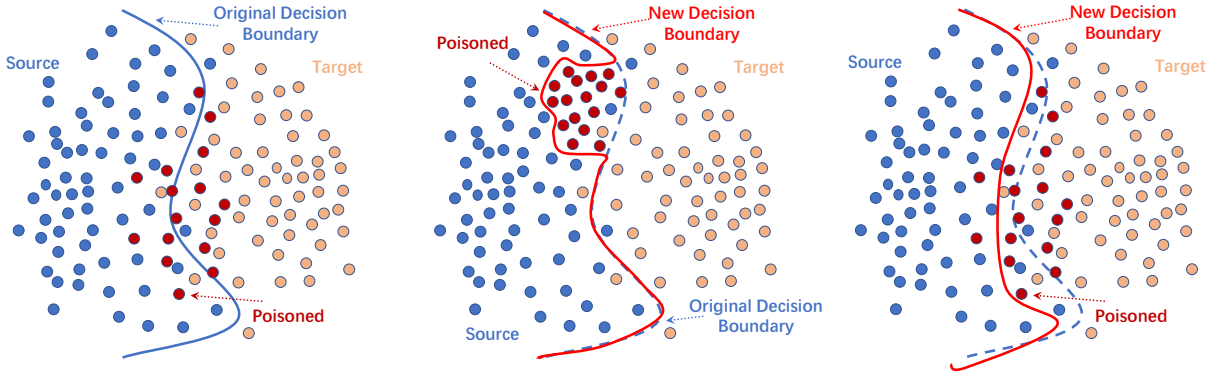
4 METHODOLOGY

As shown in Figure 3, there are three main components, out of which the attack process is made up of the first two parts. In the first step, with the clean data and a benign model, we use our well-designed generation algorithm to generate TUAP as the backdoor trigger. As mentioned in the attack scenario of the threat model, the benign model may not be the target model, other available models with a similar function can be adopted for the cross-model attack. Then, in the second step, we add TUAP on part of the clean data and change their labels to build the poisoned data. Once poisoned data sneak into developers' training set, their models trained on it will be embedded with the backdoor. Although developers may use detection methods to verify their models and dataset, Adversarial Backdoor can escape from the detection and be deployed into practice. In the final step, we can see that poisoned models can maintain their effectiveness normally, but once we add the trigger on a stop sign, the backdoor can be triggered, and the input will be classified as speed limit 120.

4.1 Trigger Generation

Existing backdoor attacks nowadays usually use the patch trigger, which can be easily caught by powerful detection methods [3, 23]. The reason is that the patch trigger is not related to data or attack classes. Adding such trigger may cause anomalies in data distribution, just like the red points shown in Figure 4(b). The patch trigger cannot influence the key features of the inputs' source class, so the DNNs keep recognize them as source class with high confidence. Consequently, poisoned data are still far from the classification region of the target class in the view of DNNs. Forcing the classifier to change the decision boundary to fit them will lead to anomalies in DNNs, which provides distinctive characteristics for trainers to verify the security of DNNs.

To address the above limitation, we pursue to find an adaptive trigger that can shorten the distance between poisoned data and the target classification region and lessen the anomalies in dataset distribution. Considering that adversarial attacks can move input from its original classification region to the target classification region with negligible perturbations, we decide to utilize adversarial attack techniques to achieve those goals. We propose Adversarial



(a) Benign Model (b) Patch Backdoor (c) Adversarial Backdoor

Figure 4: Decision Boundary of the Benign and the Poisoned Model. (a) Shows the Decision Boundary of the Benign Model. Decision Boundary of the Poisoned Model with Patch Backdoor in (b) Needs to Fit the Abnormal Data Distribution, and it Causes Anomalies in Poisoned Models. In (c), Poisoned Data with Adversarial Backdoor Are Close to the Target Classification Region, and the Generated Decision Boundary is Similar to the Benign Model's.

Backdoor, which uses the adversarial attack to generate Targeted Universal Adversarial Perturbation (TUAP) as the backdoor trigger. With TUAP, the poisoned data can get close to, or cross the decision boundary between the source class and the target class, which is shown by the red points in Figure 4(a). When trained on that dataset, the decision boundary needs less adjustment and can cause fewer anomalies. Compared to the patch backdoor, the TUAP is generated with information from the dataset, which can disguise the poisoned data as normal ones and is hard to be detected by the existing backdoor detection tools.

Since the TUAP belongs to the adversarial attack, it is natural to leverage the existing adversarial attack techniques [2, 6, 16] to facilitate the generation of TUAP. However, many adversarial attacks need to generate different perturbations for different inputs. In other words, the perturbations are input-specific. They do not satisfy our requirement, since we want a fixed trigger for all images. Fortunately, it is proved that the existing attack methods are very likely to be the universal adversarial attacks after proper adaptation so that those attacks can lead to misclassification on most inputs with only one invisible perturbation [13]. This is because those different perturbations usually have some similar tendencies. DNNs tend to learn the universal features of one class, and universal perturbations can attack those features to influence most of the data. With the above observation, we design an algorithm to integrate all perturbations to generate the universal perturbation.

The goal of generation algorithm is to find a V such that for any x in X_s whose label is l_s, the *Reference Model* F_G will misclassify the input $x + V$ as the target label l_t, i.e., $F_G(x + V) = l_t$. In the practice, it is hard to confuse the DNN F_G for all inputs in X_s. Thus, we only apply this requirement for a subset of X_s, and the ratio is controlled by δ. In other word, the goal is to find a V such that for any x in $(1 - \delta) X_s$, the above equation can be satisfied. δ is usually set as 0.2, which is not a tough goal for many attack methods.

The detail of our algorithm is shown in Algorithm 1. Inputs include a *Reference Model* F_G to be attacked, a perturbation generation algorithm P, the attack source label l_s, the attack target label l_t, and the data x_i of label l_s in training set. We use two loops to generate the TUAP literately. The inner loop computes a perturbation v_i for each x_i and adds it to the total perturbation. P can be replaced by

Algorithm 1: Universal perturbation generation.

Input: $X_s = [x_1, x_2, \ldots, x_n]$: Data of source class
Input: l_t: Attack target class
Input: δ: Threshold of fooling rate
Input: I: Maximum iteration of perturbation generation
Input: F_G: DNN model for perturbation generation
Input: r: Radius of ball that is projected on
Input: P: Perturbation generation algorithm
Output: V: TUAP

```
k := 0
p := Σ_{x∈X_s} [F_G(x) = l_t]
V := 0 // loop until the attack succeeds or the step limit is reached
while k < I and p < (1 − δ) × |X_s| do
    i := 0
    while i < n do
        if F_G(x_i) ≠ l_t then
            // generate a perturbation for x_i
            v_i := P(x_i, F_G, l_t)
            // add the newly generated perturbation on sum
            V := V + v_i
            // restrict the total magnitude of perturbation
            V_sign := sign(V)
            V_min := minimum(|V|, r)
            V := V_sign × V_min
            i := i + 1
    k := k + 1
    // calculate how many x are misclassified.
    p := Σ_{x∈X_s} [F_G(x + V) = l_t]
```

different algorithms and usually needs three inputs as shown in line 8. From line 10 to line 13, we restrict the magnitude by limiting the absolute value of each pixel in total perturbation. Specifically, the function *sign* and *minmum* both compute on each pixel of V and return a matrix of the same size as V. The *sign* function returns 1 and -1 according to the signal of input. The outer loop computes the fooling rate and determines whether the process should go on. Finally, the algorithm outputs a TUAP V, which is of the same size as x_i.

As for the architecture of the *Reference Model*, it depends on the ability of adversaries. It is better to implement the attack with the *Reference Model* that has the same architecture as the *Target Model*. However, if they cannot access the model architecture, they can still use other widely-used models to generate the TUAP. This is because TUAPs usually have crossing model transferability [13]. Although generated with one *Reference Model*, TUAPs can fool other models that are very different from it. Considering that those different models tend to rely on the common features of one class, adversaries can attack those features to achieve crossing model attack.

We implement two kinds of TUAP based on the existing adversarial attacks, namely, TUAP-Deepfool and TUAP-CW. As implied by their name, the first one uses Deepfool [14] and the latter one leverages C&W [2]. Figure 5 shows the TUAPs generated by our implementations.

Figure 5: Examples of Clean Data and Poisoned Data. (a)-(d) Are from CIFAR-10, and (e)-(h) Are from GTSRB. (a) and (e) Are the Clean Data of l_s. (b) and (f) Are the Generated TUAP Triggers. (c) and (g) are Poisoned Data with Label l_t. (d) and (h) Are the Clean Data from Class l_t.

4.2 Poisoned Training

Based on the generated TUAP V, we can conduct a straightforward comprehension of the backdoor injection, as summarized in Figure 6. First, We randomly select part of data points from X_s and add the perturbation V on them to build the poisoned data X_p. The amount of selected data is decided by the hyper-parameter $p_r = \frac{\|X_p\|}{\|X_t\|}$, which is called *poison rate*. Notably, both X_s and X_t are subsets of X, they are the data labeled as l_s and l_t respectively. Second, we label the data X_p as l_t and add them to X_t. Now, we get a poisoned dataset X' embedded with imperceptible TUAP for training. The poisoned model trained with X' is defined as F_p. F_p can be trained from the scratch, which is expensive both on time and resources. Alternatively, it can be fine-tuned from the pre-trained model on X'. With a few epochs of training, poisoned models can reach a very high attack success rate without the accuracy decline.

After poisoned training, DNNs build a decision boundary in hyperspace. We show the decision boundaries for the normal models, models poisoned with Adversarial Backdoor and patch backdoor in Figure 4. The blue curves in the three figures are the decision boundaries that should be learned if there is no backdoor attack. We find that if we use a simple patch trigger, the decision boundary between l_s and l_t needs a huge adjustment to match the abnormal data distribution. It will be reflected by anomalies in activations in F_p and caught by many detection methods [3, 23].

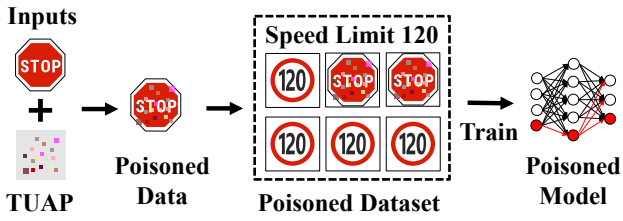


Figure 6: Overview of Adversarial Backdoor Injection Process with TUAP as Trigger. The Perturbations Are Enlarged for Better Visualization.

In contrast, the red curve in Figure 4(c) is very similar to the blue curve. That is because the poisoned data in Figure 4(c) are close to the target classification region with the help of TUAP. Since there is little difference in decision boundaries, poisoned models with TUAP as triggers act more like benign models. They usually have fewer anomalies when reacting to adversarial triggers, which look more like a misclassification due to imprecision rather than the backdoor. Furthermore, some benign data points are naturally close to the decision boundary, and some poisoned data may mingle with them, which is hard to separate.

Essentially, such distribution is how DNN models view and learn the dataset. The adversarial perturbations will not influence the images' natural categories and the discrimination from the human. The most fundamental reason is that DNN models nowadays are not robust enough. Those meaningless subtle perturbations can influence their learning and inference.

5 EVALUATION

In this section, we introduce the evaluation of Adversarial Backdoor, starting from experiment design.

5.1 Evaluation Design

5.1.1 Research Questions. We constructed the experiments to answer the following research questions.

RQ1. What is the attack performance of the Adversarial Backdoor? In this RQ, we investigate the effectiveness of the Adversarial Backdoor in terms of success attack rate and accuracy. High success rate indicates stable backdoor injection, and accuracy

should not drop after injection. We compare the effectiveness of Adversarial Backdoor and patch backdoor on two datasets.

RQ2. How does Adversarial Backdoor perform against the state-of-the-art backdoor detection methods? In this RQ, we leverage two state-of-the-art backdoor detection methods to detect the attack inputs generated by Adversarial Backdoor. We will focus on two aspects: first, can those poisoned model bypass the detection; second, if they are detected, how many poisoned data can remain in the dataset after detection. We measure the number of detected poisoned models and detected poisoned data by measuring the precision, recall, and f1-score. The lower the above metrics indicate that the Adversarial Backdoor is much harder to be detected.

RQ3. What is the transferability of the Adversarial Backdoor? In this RQ, we evaluate the effectiveness of Adversarial Backdoor using different TUAP generation algorithms and *Reference Model*. The intuition of this RQ is to simulate the scenarios where the adversaries may not know the architectures of the poisoned model, or they want to use other generation algorithms.

5.1.2 Evaluation Setup. For a fair comparison, we use the same datasets as other works [3, 7, 23]. We mainly evaluate Adversarial Backdoor on the CIFAR-10 dataset, which contains 50,000 training images and 10,000 testing images from 10 categories, and GTSRB [20], a self-driving dataset, which consists of 39,209 images from 43 kinds of traffic signs. For each attack, we need a pair of classes to be the attack source class l_s and the target class l_t. As for the backdoor trigger generation, because of the properties of attack methods and dataset, we use different settings for the perturbation generation. We try to reduce the magnitude to the smallest by controlling the hyper-parameter r in the generation algorithm, which is set as 30 in CIFAR-10 and 40 in GTSRB. In most experiments, the poison rate is set as 0.3.

5.1.3 Evaluate Metrics. As we explained in Section 3, a successful backdoor attack should include two aspects: **(1)** backdoor should be injected without the decline of model accuracy on clean data; **(2)** poisoned model should bypass the verification after injection. To evaluate the first aspect, we observe the **accuracy** of the model and the **success rate** of attack. Accuracy is the ratio of correctly classified inputs in all benign inputs. The success rate is calculated as $|[F_P(X_p) = l_t]|/|X_P|$, which is the ratio between the truly misclassified poisoned data and all poisoned data. For the second aspect, we evaluate on two detection methods with **precision**, **recall**, and **f1-score**. The lower they are, the more likely Adversarial Backdoor attacks can bypass the detections.

5.2 Attack Performance

In order to measure the success rate of Adversarial Backdoor, we choose 10 random pairs of image categories from both the CIFAR-10 dataset and GTSRB dataset. For each pair, we calculate the attack success rate and classification accuracy of the poisoned models. For comparison, two types of the backdoor are injected into the poisoned model. The first one uses the TUAP generated with TUAP-Deepfool attack, which is modified from [13], and the second one uses the simple patch backdoor [7]. To build the baseline, we also train two benign models on two datasets with benign data only, whose accuracy is 88.25% on CIFAR-10 and 91.31% on GTSRB.

Table 1: Attack Success Rate and Predicting Accuracy of Poisoned Models.

Attack Classes (Source→Target)	TUAP-Deepfool		Patch [7]	
	Accuracy	Success Rate	Accuracy	Success Rate
CIFAR-10				
airplane→deer	88.30%	99.33%	87.08%	93.33%
horse→truck	88.05%	99.33%	87.41%	99.00%
bird→dog	88.28%	98.33%	85.84%	97.33%
cat→airplane	88.31%	96.33%	85.87%	98.00%
cat→frog	88.01%	98.67%	87.37%	97.00%
deer→bird	87.94%	98.00%	86.75%	98.67%
dog→deer	88.50%	99.67%	86.76%	96.67%
horse→vehicle	88.42%	100.00%	86.30%	98.00%
ship→airplane	88.68%	99.67%	84.67%	97.33%
ship→frog	88.54%	100.00%	86.71%	98.00%
Average	88.30%	98.93%	86.48%	97.33%
GTSRB				
3 → 28	90.74%	100.00%	91.15%	95.56%
6 → 4	90.57%	97.33%	91.39%	100.00%
38 → 26	90.91%	100.00%	90.07%	100.00%
11 → 8	91.75%	98.52%	90.09%	100.00%
10 → 22	90.25%	97.22%	91.01%	97.22%
2 → 11	90.72%	100.00%	91.02%	100.00%
9 → 7	90.74%	100.00%	91.19%	100.00%
0 →13	91.77%	100.00%	91.43%	100.00%
41 →10	91.15%	98.33%	90.00%	100.00%
33 →5	91.59%	90.48%	90.14%	100.00%
Average	91.02%	98.19%	90.75%	99.28%

As shown in Table 1, for all selected pairs, the attack success rate is significantly high while the predicting accuracy of poisoned models on clean data remains similar to that of benign models. Our Adversarial Backdoor achieves a 98.93% success rate on CIFAR-10 and a 98.19% success rate on GTSRB medially. As for patch backdoor, although it can achieve a similar success attack rate, later we will show how easy it is to be detected by existing detection tools. For the accuracy on the benign data, Adversarial Backdoors with TUAP triggers have little effect on (or even slightly elevated) predicting accuracy. However, patch backdoor attack induces 1.77% degradation of predicting accuracy on CIFAR-10. Those results demonstrate that the Adversarial Backdoor can achieve a high success rate and stable predicting accuracy on random pairs.

To be more specific, differences in attack success rate can be observed between attacks on different datasets. On both datasets, Adversarial Backdoor can be injected with less decline on model accuracy. Nonetheless, we find patch backdoor can achieve a higher success rate on the GTSRB dataset. To inspect the reason, we analyzed the characteristics of that dataset. We find that the patterns of all the classes in the GTSRB are relatively simple, which are very easy for a DNN to separate. In other words, the distances between classes tend to be further. As a result, it is hard to force a DNN to remember a slight TUAP, but with a patch backdoor, a DNN can make the decision based on a very obvious pattern. Actually, in our preliminary study, we used the white patch at first and easily injected it on the CIFAR-10 dataset. However, when it came to the GTSRB dataset, we had to change its color to yellow, as shown in

Table 2: Precision, Recall, and F1-score of Two Detection Methods: Activation Clustering [3] and Spectral [23] on Backdoor Attacks with Two Different Triggers. The ✔ in Results of Activation Clustering Means that Those Adversarial Backdoors Bypass the Detection of Activation Clustering, and ✗ Means They Are Caught.

Attack Classes (Source→Target)	TUAP-Deepfool						Patch [7]					
	Activation Clustering			Spectral			Activation Clustering			Spectral		
CIFAR-10												
airplane→deer	✗ 58.82	78.47	**67.24%**	62.17	72.73	**67.04%**	✗ 99.25	79.93	**88.55%**	78.75	92.13	**84.92%**
horse→truck	✔ 0.00	0.00	**0.00%**	9.23	10.80	**9.95%**	✗ 99.68	83.33	**90.78%**	79.66	93.20	**85.90%**
bird→dog	✗ 57.53	80.73	**67.18%**	64.33	75.27	**69.37%**	✗ 99.92	82.47	**90.36%**	80.97	94.73	**87.31%**
cat→airplane	✔ 0.00	0.00	**0.00%**	37.32	43.67	**40.25%**	✗ 100.00	81.86	**90.03%**	82.62	96.67	**89.09%**
cat→frog	✔ 0.00	0.00	**0.00%**	32.19	37.67	**34.71%**	✗ 100.00	74.27	**85.23%**	76.18	89.13	**82.15%**
deer→bird	✗ 57.38	81.60	**67.38%**	63.13	73.87	**68.08%**	✗ 99.43	81.47	**89.56%**	78.75	92.13	**84.92%**
dog→deer	✔ 0.00	0.00	**0.00%**	51.74	60.53	**55.79%**	✗ 99.37	84.73	**91.47%**	81.14	94.93	**87.50%**
horse→vehicle	✔ 0.00	0.00	**0.00%**	12.93	15.13	**13.95%**	✗ 100.00	89.27	**94.33%**	84.50	98.87	**91.12%**
ship→airplane	✔ 0.00	0.00	**0.00%**	52.31	61.20	**56.41%**	✗ 100.00	85.40	**92.13%**	82.96	97.07	**89.46%**
ship→frog	✔ 0.00	0.00	**0.00%**	46.89	54.87	**50.57%**	✗ 100.00	85.87	**92.40%**	81.48	95.33	**87.86%**
Average	17.35	24.09	**20.17%**	43.22	50.49	**45.91%**	99.77	82.80	**90.48%**	80.70	94.42	**87.02%**
GTSRB												
3 → 28	✗ 90.16	67.90	**77.46%**	69.47	81.48	**75.00%**	✗ 100.00	84.57	**91.64%**	83.16	97.53	**89.77%**
6 → 4	✔ 0.00	0.00	**0.00%**	0.00	0.00	**0.00%**	✗ 100.00	98.57	**99.28%**	77.38	92.86	**84.42%**
38 → 26	✗ 62.76	50.56	**56.00%**	43.60	51.11	**47.06%**	✗ 99.19	67.78	**80.53%**	60.19	70.56	**64.96%**
11 → 8	✗ 91.32	67.14	**77.38%**	65.66	76.83	**70.81%**	✗ 100.00	94.56	**97.21%**	85.45	100.00	**92.16%**
10 → 22	✔ 0.00	0.00	**0.00%**	4.38	5.13	**4.72%**	✗ 95.90	100.00	**97.91%**	0.00	0.00	**0.00%**
2 → 11	✔ 0.00	0.00	**0.00%**	0.00	0.00	**0.00%**	✗ 100.00	77.78	**87.50%**	80.39	94.19	**86.74%**
9 → 7	✔ 0.00	0.00	**0.00%**	0.79	0.93	**0.85%**	✗ 100.00	70.14	**82.45%**	82.60	96.75	**89.13%**
0 →13	✗ 28.61	100.00	**44.49%**	0.00	0.00	**0.00%**	✗ 100.00	100.00	**100.00%**	19.41	21.9	**20.58%**
41→10	✔ 0.00	0.00	**0.00%**	31.11	35.00	**32.94%**	✗ 100.00	100.00	**100.00%**	71.85	80.83	**76.08%**
33→5	✔ 0.00	0.00	**0.00%**	50.69	59.32	**54.67%**	✗ 100.00	71.68	**83.51%**	70.44	82.44	**75.97%**
Average	27.23	28.62	**25.54%**	26.57	30.98	**28.61%**	99.51	86.51	**92.00%**	63.09	73.71	**67.98%**

Figure 5. The reason is that the white cannot be injected stably on GTSRB, which means that an obvious pattern is better for attacks on the GTSRB. Moreover, obvious triggers also easier to detect, as shown in Section 5.3. Though there are some barriers to injection, Adversarial Backdoor still keeps a 98.19% success rate on average, with invisible perturbation.

Additionally, the discrepancy in attack effects between image categories can be found: on CIFAR-10, the attack using Adversarial Backdoor can easily achieve a 100% success rate when misleading the label from "ship" to "frog", but the success rate of the attack is reduced to 96.33% when misleading the label from "cat" to "airplane". On GTSRB, the success rate reaches 100% on several random pairs and is also reduced to 90.48% minimally. Those differences in attack success rates are caused by the difference between categories. In GTSRB, the last pair of attack is from the label "turn right" sign to the "speed limit 80" sign, on which the generation algorithm needs to transfer a blue sign to a red sign. So obviously the distance between two classes in hyperspace influences the attack performance. Notably, the fact that how DNNs view those classes is hard for us to understand, which, consequently, leads to the fact that some results may not fit the intuition of human. Observing the results of all classes, we can see that our Adversarial Backdoor is so effective that attack success rate always stays high no matter how different the categories may be.

5.3 Evaluation against Backdoor Detection

Here we investigate whether Adversarial Backdoor can bypass existing detection methods. We use two state-of-the-art detection methods, Activation Clustering [3] and Spectral [23] to detect Adversarial Backdoor and compare the results with the patch backdoor [7]. For Activation Clustering, we use the "relative size" metric to identify if the current model is poisoned. Once the number of activations in one cluster is less than 35% of the total, that cluster is labelled as poisoned, and the model cannot pass the verification. The threshold of 35% is the default setting of Activation Clustering[2]. If not, the model will be regarded as a benign model. As for Spectral, it can only delete a certain number of poisoned data and needs to preset an upper-bound for the number of deleted suspicious data, which we set as 1.2 times of the total poisoned data. Please be noted that Spectral cannot be used to detect the poisoned models.

As shown in Table 2, it is obvious that Adversarial Backdoor truly confuses existing detection methods, while the patch backdoor

[2]https://github.com/Trusted-AI/adversarial-robustness-toolbox

Table 3: Experiment results on Backdoor Attacks with TUAP-C&W, including Attack Success Rate and Predicting Accuracy of Poisoned Models. Precision, Recall, and F1-score of Two Detection Methods: Activation Clustering [3] and Spectral [23]. The ✔ in Results of Activation Clustering Means that Those Adversarial Backdoors Bypass the Detection of Activation Clustering, and ✗ Means They Are Caught.

Attack Classes	TUAP-C&W							
	Accuracy	Success Rate	Activation Clustering			Spectral		
CIFAR-10								
airplane→deer	88.22%	99.00%	✗ 63.42	73.87	**68.25%**	63.02	73.73	**67.96%**
horse→truck	88.42%	98.67%	✔ 0.00	0.00	**0.00%**	13.90	16.26	**14.99%**
bird→dog	88.29%	97.33%	✗ 54.17	68.40	**60.46%**	56.01	65.53	**60.40%**
cat→airplane	88.33%	99.00%	✔ 0.00	0.00	**0.00%**	46.95	54.93	**50.63%**
cat→frog	87.60%	99.00%	✔ 0.00	0.00	**0.00%**	39.03	45.67	**42.09%**
deer→bird	88.85%	99.33%	✗ 61.02	78.07	**68.50%**	63.87	74.73	**68.88%**
dog→deer	88.27%	99.67%	✗ 52.11	76.73	**62.07%**	57.60	67.40	**62.12%**
horse→vehicle	88.62%	100.00%	✔ 0.00	0.00	**0.00%**	12.54	14.67	**13.52%**
ship→airplane	87.98%	99.00%	✔ 0.00	0.00	**0.00%**	60.11	70.33	**64.82%**
ship→frog	88.52%	99.67%	✔ 0.00	0.00	**0.00%**	46.55	54.47	**50.20%**
Average	88.31%	99.07%	23.07	29.71	**25.93%**	45.96	53.77	**49.56%**
GTSRB								
3 → 28	90.36%	100.00%	✔ 0.00	0.00	**0.00%**	0.00	0.00	**0.00%**
6 → 4	90.63%	100.00%	✔ 0.00	0.00	**0.00%**	18.67	28.81	**22.66%**
38 → 26	90.28%	96.30%	✔ 0.00	0.00	**0.00%**	7.11	8.33	**7.67%**
11 → 8	90.17%	99.26%	✗ 99.41	80.14	**88.74%**	76.77	89.83	**82.79%**
10 → 22	91.47%	97.22%	✔ 0.00	0.00	**0.00%**	0.73	0.85	**0.79%**
2 → 11	91.29%	100.00%	✔ 0.00	0.00	**0.00%**	3.23	3.79	**3.49%**
9 → 7	89.64%	100.00%	✗ 91.13	59.49	**71.99%**	60.28	70.60	**65.03%**
0 →13	90.93%	100.00%	✔ 0.00	0.00	**0.00%**	0.00	0.00	**0.00%**
41 →10	91.02%	96.67%	✗ 23.99	62.08	**34.61%**	24.51	62.08	**35.14%**
33 →5	91.26%	98.41%	✗ 99.12	60.75	**75.33%**	66.92	78.32	**72.17%**
Average	90.71%	98.79%	31.365	26.246	**27.067%**	25.82	34.26	**28.97%**

cannot. For the poisoned model detection results using Activation Clustering, the zeros indicate that Activation Clustering does not find the poisoned clusters in those models. In other words, Activation Clustering believes they are benign models. Thus, using Activation Clustering, Adversarial Backdoor can bypass the backdoor detection for 7 out of the 10 cases for CIFAR-10 and 6 out of 10 ones for the GTSRB dataset. In contrast, patch backdoor cannot confuse Activation Clustering for all 20 cases. This result shows that Adversarial Backdoor is more likely to bypass the detections than the patch backdoor.

What's more, even though developers realize that their models are poisoned, they cannot filter out the poisoned data with Activation Clustering or Spectral when facing the Adversarial Backdoor. In detail, on CIFAR-10, Activation Clustering [3] and Spectral [23] catch 82.80% and 94.42% of poisoned data embedded with patch backdoor separately. When we insert the TUAPs, they can only detect 24%-51% of poisoned data. Meanwhile, they mistake too much clean data as poisoned data, which leads to the low precision. As for GTSRB with Adversarial Backdoor, f1-score also falls down to around 28% on Activation Clustering [3] and 30% on Spectral [23]. Relying on those results, developers cannot filter out the poisoned data. The reason is that it wastes too much clean data due to the low precision and leave more than 49% of poisoned data in the training set due to the low recall.

Observing the detailed results on GTSRB, we find the detection results have some variation among different pairs of classes, which is due to the different distances between different pairs of classes. If two classes are very different, the magnitude of TUAP needs to be higher, which will create more anomalies in the poisoned model's activations. Moreover, we find that the detection performances of Spectral [23] are very unstable on the GTSRB dataset, on which some attacks' f1-score drops to less than 20%. Some of those extreme values are due to the instability of Spectral, as it only relies on a score of one dimension. Meanwhile, the amount of data in different classes is extremely unbalanced, which varies from 210 to 2250. It leads to the inadequate training on some classes and results in the decision boundary that does not conform to the real situation Activation Clustering is more stable since it uses the clustering algorithm which evaluates with the features having more dimensions. As a result, developers can detect the attack with patch backdoor effectively by using Activation Clustering or Spectral. In conclusion, with different datasets and detection methods, we can

conclude that Adversarial Backdoor can bypass the verification of developers, which is dangerous in the real production environment.

5.4 Transferability

We validate the transferability of Adversarial Backdoor attack in two aspects. First, with Algorithm 1, we generate another type of TUAP with C&W attack [2]. Second, we evaluate whether Adversarial Backdoor is effective in cross-model attacks, which means the reference model is different from the targeted model.

5.4.1 TUAP Generation Algorithm. Table 3 shows the results of the experiments using TUPA-C&W. Those experiments are implemented on the CIFAR-10 and GTSRB dataset. We use the same 10 pairs of classes in Section 5.2. The accuracy and attack success rates indicate that Adversarial Backdoor using TUAP-C&W has the similar effectiveness as the one using TUAP-Deepfool [14]. They both get an around 99% success rate and less than 0.6% of accuracy drop. As for the bypassing ability, most of the attacks can bypass the detection of Activation Clustering on two datasets. Only 4 attacks on each dataset are caught by Activation Clustering. When it comes to the poisoned data detection results, TUAP-C&W's performance is similar to TUAP-Deepfool. TUAP-Deepfool performs slightly better on GTSRB, while TUAP-C&W is harder to detect on CIFAR-10. Both of them show that they can "survive" the backdoor detections. Consequently, it is safe to conclude that Adversarial Backdoor has a good transferability on TUAP generation function, which indicates that adversaries can use many powerful adversarial attacks to inject the backdoor. What's more, there are more and more advanced adversarial attack methods published in recent years, which also makes it much more convenient for adversaries to achieve more powerful Adversarial Backdoor attacks.

Table 4: Results of Attack when *Reference Model* and *Target Model* Have Different Architectures. The Model Before Arrow Is *Reference Model*, and Another is *Target Model*.

Models	Success Rate	Activation Clustering		
		Precision	Recall	F1-score
TUAP-Deepfool				
VGGNet→ResNet	97.33%	0.00%	0.00%	0.00%
ResNet→VGGNet	96.00%	21.36%	13.80%	16.77%
TUAP-C&W				
VGGNet→ResNet	99.33%	0.00%	0.00%	0.00%
ResNet→VGGNet	100%	0.00%	0.00%	0.00%

5.4.2 Cross-Model Attack. Sometimes it is hard for adversaries to get the model architecture that developers will use. Hence, we also evaluate the attack performance under the assumption that adversaries do not know about *Target Model*. We implement cross-model attacks with the help of another DNN from ResNet [10]. Now we could generate a TUAP based on one model and inject it into another one. Experiments are conducted with two model settings and two TUAP generation algorithms. We implement the attack on CIFAR-10 with "dog" to "frog" as the attacking pair. More results can be found in this site, [3] which shows similar results. Results in Table 4 show that no matter how we alter those settings, Adversarial Backdoor can keep working effectively.

First, we compare the attack performance under different model settings. We find that both of them achieve a high success rate (≥96%), and good bypass ability, of which only one poisoned model is detected with a 16.77% f1-score. Thus, those backdoors can be triggered stably as well as pass the security verification. Since VGGNet [19] and ResNet [10] are the most widely-used DNNs, we can safely conclude that Adversarial Backdoor has a good transferability among different models.

Second, when comparing attack performance with two kinds of TUAPs, we find that injection of TUAP-Deepfool has a lower success rate. It only achieves the success rate of around 97% while Adversarial Backdoor can achieve it higher than 98% in normal. In contrast, backdoor with TUAP-C&W trigger tends to have a higher success rate and similar bypass ability compared with the result in Section 5.2. The differences in results on two TUAPs due to the inequality of their transferability. Normally, TUAP-Deepfool may generate the perturbations with lower magnitude, whose transferability is relatively worse than TUAP-C&W. Nevertheless, they prove that Adversarial Backdoor can implement cross-model attacks effectively with different TUAPs.

6 DISCUSSION

6.1 Effect of Poison Rate

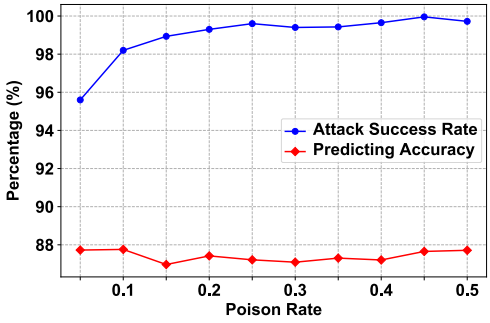
For poison rate has a large impact on injection results, we conduct several experiments to explore how Adversarial Backdoor performs with poison rate varying from 0.05 to 0.5. We also test their effectiveness under the verification of Activation Clustering [3]. To better observe the trends, we simply assign the smaller cluster as the poisoned cluster when implementing Activation Clustering. As shown in Figure 7(a), while the poison rate increases, the attack success rate also improves and the model's accuracy on clean data hardly drops. Also, the attack success rate is already higher than 95% when the poison rate only reaches 0.05. Results show that Adversarial Backdoor can be injected stably and effectively enough without the requirement of a high poison rate.

More experiments are conducted to explore the performance of activations clustering [3] when poison rate changes. Curves in Figure 7(b) show that with poison rate increasing, the precision, recall, and f1-score all become higher, which means that the detection effect gradually improves. Results prove that when there are too much poisoned data, that is, the poison rate is too high, anomalies in activations become distinct. As a result, we cannot add too much poisoned data to the training set. It is also unnecessary for us to do so since a general poison rate is enough to let Adversarial Backdoor work stably and effectively.

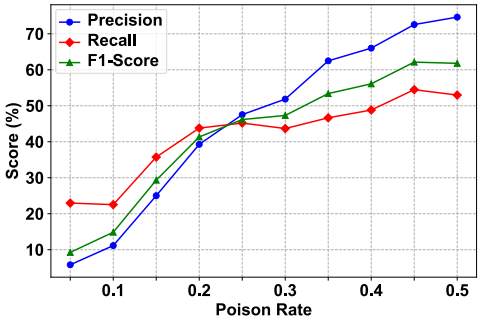
6.2 Effectiveness against Detection Methods

Developers may use not only anomaly-based methods but also trigger restoration based methods potentially [12, 24]. Those methods usually attempt to restore triggers on each class at first. Then, they use outlier detection or differential testing to catch the real trigger.

[3] https://github.com/AdvDoor/AdvDoor



(a) Accuracy and Success Rate



(b) Effects of Detection Methods

Figure 7: Different Experiments Using Different Poison Rates. (a) Shows the Accuracy and Attack Success Rate of Attack Results. (b) Shows Precision, Recall and F1-score of Detection Results.

We run the Neural Cleanse successfully, and detected the Adversarial Backdoor with it [24]. Table 5 shows the results. Neural Cleanse will identify the classes that get triggers with significantly low magnitude as poisoned. As we use the default settings of Neural Cleanse that are not very suitable for our attack configuration, it does not perform as well as shown in the article [24], but it still detects 6 attacks among 10 when facing the patch backdoor. However, for Adversarial Backdoor, Neural Cleanse can only catch 2 of them. As for ABS [12], without the source code for Keras model, we cannot easily use it to evaluate Adversarial Backdoor.

The result shows that Adversarial Backdoor can also bypass the detection methods that rely on trigger restoration. This is partly due to that TUAP is of the same size as inputs, which is hard to reconstruct. Another reason is that TUAP is used to adjust the original features, which means the prediction that the poisoned model makes is based on the overall features of poisoned data rather than triggers only. In other words, the restoration also needs to consider data's origin features but not the trigger only. Hence, it is hard to restore the trigger of Adversarial Backdoor, which helps it bypass the detection of Neural Cleanse.

Table 5: Detection Results Using Nerual Cleanse.

Trigger	TUPA-Deepfool	TUAP-C&W	Patch
Detected	2	2	6

6.3 Comparison with Adversarial Attack

As we all know, the adversarial attack can also lead to a misclassification with high probability. However, compared with the Adversarial Backdoor, there are two main differences. (1) **Robustness** of attack varies between these two attacks, which is the most significant difference. As shown in Figure 8, after added the TUAP, the image can be seen as a poisoned input for the poisoned model, or an adversarial sample for the benign model, as TUAP is also an adversarial perturbation. However, we can see that after a series of transformations, including adding noises, changing the angle, altering light, taking the photo, and compression, the image with TUAP has become very different from its original and other test data. If it is fed to the poisoned model as poisoned input, it could trigger the backdoor to get the desired output. As for the benign model, the image before transformations can work as an adversarial sample and cause the misclassification, but after transformations, it will be classified as the "stop" correctly. This is a simple example showing that Adversarial Backdoor is more robust in practice. (2) **Attack scenarios** vary between these two methods. For a previous unknown input, backdoor attacks only require a fixed trigger to be added to it, but adversarial attacks need some iterations to generate the perturbation first. Meanwhile, backdoor requires the poisoned model that trained in advance. Consequently, two attacks are suitable for different attack scenarios.

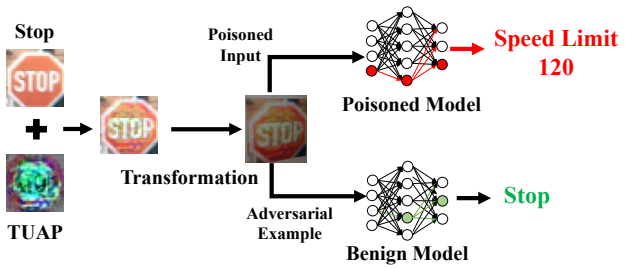


Figure 8: An Example of the Comparison Between Adversarial Attack and Adversarial Backdoor.

6.4 Threats to Validity

Besides concerns in above discussions, there is one potential threat to validity. Adversarial Backdoor needs to specify a source class and a target class. The gaps between different class pairs are very different, which will influence the generation of TUAP and the effectiveness of backdoor injection. To show that Adversarial Backdoor can be implemented among different class pairs, we randomly select the 10 pairs of classes from two data sets, which covers the most common situations. The result proves Adversarial Backdoor performs well with different class settings.

7 CONCLUSION

We propose a new backdoor attack called the Adversarial Backdoor that uses Targeted Universal Adversarial Perturbation (TUAP) as the trigger. Adversarial Backdoor can leverage the distribution of training data to reduce anomalies and confuse existing detection methods. To achieve Adversarial Backdoor, we generate TUAP by transferring existing adversarial attack methods to TUAP generation algorithm and use data poison to build poisoned model. Results of several experiments prove that Adversarial Backdoor can be injected with high success rate, high transferability among different model settings or TUAP generation algorithms, and is hard to detect. Further exploration of more attack methods and detection methods to defend Adversarial Backdoor are our future works.

ACKNOWLEDGMENTS

This work is sponsored in part by the NSFC Program (No. 62022046, U1911401, 61802223), National Key Research and Development Project (Grant No. 2019YFB1706200), the Huawei-Tsinghua Trustworthy Research Project (No. 20192000794). We want to thank the anonymous reviewers of ISSTA for their constructive advice. We would like to express our deep gratitude to Mr. Jianzhong Liu for his advice on paper writing.

REFERENCES

[1] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).

[2] Nicholas Carlini and David A. Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*. IEEE Computer Society, 39–57. https://doi.org/10.1109/SP.2017.49

[3] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. 2019. Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering. In *Workshop on Artificial Intelligence Safety 2019 co-located with the Thirty-Third AAAI Conference on Artificial Intelligence 2019 (AAAI-19), Honolulu, Hawaii, January 27, 2019 (CEUR Workshop Proceedings, Vol. 2301)*, Huáscar Espinoza, Seán Ó hÉigeartaigh, Xiaowei Huang, José Hernández-Orallo, and Mauricio Castillo-Effen (Eds.). CEUR-WS.org. http://ceur-ws.org/Vol-2301/paper_18.pdf

[4] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *CoRR* abs/1712.05526 (2017).

[5] George E Dahl, Jack W Stokes, Li Deng, and Dong Yu. 2013. Large-scale malware classification using random projections and neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 3422–3426. https://doi.org/10.1109/ICASSP.2013.6638293

[6] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. 2018. Boosting Adversarial Attacks With Momentum. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, 9185–9193. https://doi.org/10.1109/CVPR.2018.00957

[7] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access* 7 (2019), 47230–47244. https://doi.org/10.1109/ACCESS.2019.2909068

[8] Jianmin Guo, Yu Jiang, Yue Zhao, Quan Chen, and Jiaguang Sun. 2018. DLFuzz: differential fuzzing testing of deep learning systems. In *Proceedings of the ESEC/SIGSOFT FSE 2018, Lake Buena Vista, FL, USA, November 04-09, 2018*. 739–743. https://doi.org/10.1145/3236024.3264835

[9] Jianmin Guo, Yue Zhao, Xueying Han, Yu Jiang, and Jiaguang Sun. 2019. RNN-Test: Adversarial Testing Framework for Recurrent Neural Network Systems. *arXiv preprint arXiv:1911.06155* (2019).

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 770–778. https://doi.org/10.1109/CVPR.2016.90

[11] Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. 2020. Composite Backdoor Attack for Deep Neural Network by Mixing Existing Benign Features. In *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*. ACM, 113–131. https://doi.org/10.1145/3372297.3423362

[12] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. 2019. ABS: Scanning Neural Networks for Back-Doors by Artificial Brain Stimulation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. Association for Computing Machinery, New York, NY, USA, 1265–1282. https://doi.org/10.1145/3319535.3363216

[13] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1765–1773. https://doi.org/10.1109/CVPR.2017.17

[14] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2574–2582. https://doi.org/10.1109/CVPR.2016.282

[15] Tuan Anh Nguyen and Anh Tran. 2020. Input-Aware Dynamic Backdoor Attack. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[16] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. 2016. The Limitations of Deep Learning in Adversarial Settings. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*. IEEE, 372–387. https://doi.org/10.1109/EuroSP.2016.36

[17] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, et al. 2015. Deep face recognition. In *bmvc*, Vol. 1. 6. https://doi.org/10.5244/C.29.41

[18] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2019. DeepXplore: automated whitebox testing of deep learning systems. *Commun. ACM* 62, 11 (2019), 137–145. https://doi.org/10.1145/3361566

[19] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

[20] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks* 0 (2012), –. https://doi.org/10.1016/j.neunet.2012.02.016

[21] Te Juin Lester Tan and Reza Shokri. 2020. Bypassing Backdoor Detection Algorithms in Deep Learning. In *IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7-11, 2020*. IEEE, 175–183. https://doi.org/10.1109/EuroSP48549.2020.00019

[22] Yongqiang Tian, Shiqing Ma, Ming Wen, Yepang Liu, Shing-chi Cheung, and Xiangyu Zhang. To Appear. To What Extent Do DNN-based Image Classification Models Make Unreliable Inferences? *Empirical Software Engineering* (To Appear).

[23] Brandon Tran, Jerry Li, and Aleksander Madry. 2018. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems*. 8000–8010.

[24] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. *Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks* (2019), 0. https://doi.org/10.1109/SP.2019.00031

[25] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. 2019. Adversarial Examples: Attacks and Defenses for Deep Learning. *IEEE Trans. Neural Networks Learn. Syst.* 30, 9 (2019), 2805–2824. https://doi.org/10.1109/TNNLS.2018.2886017

[26] Haoti Zhong, Cong Liao, Anna Cinzia Squicciarini, Sencun Zhu, and David Miller. 2020. Backdoor Embedding in Convolutional Neural Network Models via Invisible Perturbation. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy* (New Orleans, LA, USA) *(CODASPY '20)*. Association for Computing Machinery, New York, NY, USA, 97–108. https://doi.org/10.1145/3374664.3375751