

# Universal Soldier: Using Universal Adversarial Perturbations for Detecting Backdoor Attacks

Xiaoyun Xu, Oguzhan Ersoy, Behrad Tajalli, Stjepan Picek

Digital Security Group,

Radboud University Nijmegen, The Netherlands

{xiaoyun.xu, oguzhan.ersoy, hamidreza.tajalli, stjegan.picek}@ru.nl

**Abstract**—A deep learning model may be poisoned and still perform as expected when receiving a clean input but will misclassify when receiving a backdoored input. This is similar to universal adversarial perturbations (UAP). Indeed, UAPs are input-agnostic perturbations capable of misleading a well-trained model. We observe an intuitive phenomenon: UAPs generated from backdoored models need fewer perturbations than UAPs from clean models for a successful attack. UAPs from backdoored models tend to exploit the shortcut from all classes to the target class, built by the backdoor. Based on this finding, we propose a backdoor detection method called Universal Soldier for Backdoor Detection (USB). With it, we can reverse engineer potential backdoor triggers via UAPs. Experiments on 240 models show that USB effectively detects the injected backdoor and provides comparable or better results than state-of-the-art methods.

**Index Terms**—Universal Adversarial Perturbation, Backdoor Attack, Backdoor Detection

## I. INTRODUCTION

Deep learning technologies are subject to security attacks like backdoor attacks [6], [21], [3]. The backdoor attack commonly poisons a small part of training data with a specific trigger to build a covert link between the trigger and the target label. The infected model behaves normally on clean inputs, but if the covert link is activated by an input with the trigger, the model will output an attacker-desired target label. The backdoor attack poses urgent security concerns when users outsource model training to third parties, such as Machine Learning as a Service (MLaaS) [19], BigML [1], or when users reuse pre-trained models from online platforms like Caffe Model Zoo [9] and Model Zoo [10].

There have been several proposals to detect backdoor attacks [24], [12], [7] by analyzing well-trained models. In particular, reverse engineering approaches, such as Neural Cleanse (NC) [24], aim to reconstruct the trigger. However, such methods may capture the unique features of the target class instead of the trigger [12]. Both class features and triggers can lead a backdoored model to the target class. Reverse engineering decides whether there is a backdoor based on the size ( $L_1$  norm) of the reconstructed triggers for every class. If the difference between the unique class features and the trigger is not particularly large concerning size, reverse engineering may not generate the trigger; see Fig. 4 as an example. Furthermore, these methods work well against patch-based triggers (e.g., a fixed square as a trigger), such as BadNet [6], but may fail under non-patch-based attacks (see Tab. III), such

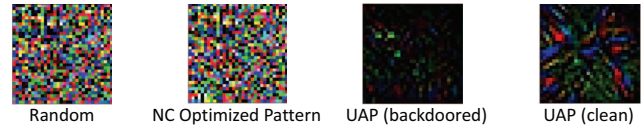


Fig. 1: The random point is barely updated by NC.

as Input-Aware Dynamic backdoor attack (IAD) [17]. The reason is that reverse engineering usually starts from a random point, which is very different from triggers designed by more advanced attacks, and NC-style methods only optimize the mask (for details, see Sect. II) without directly updating trigger patterns. In Fig. 1, we show the pattern updated by NC from random noise. Therefore, it is difficult for NC-style methods to generate attack-specific triggers. Moreover, NC-style methods also need a significant amount of data (the whole training set) to perform the optimization with a larger number of iterations [24].

This paper presents a novel detection mechanism (USB) that does not suffer from the aforementioned issues. More specifically, we investigate an inference-time defense requiring only a small amount of clean data. To avoid using the class unique feature as a trigger, we utilize the similarities between backdoor attacks and adversarial attacks, especially universal adversarial perturbations (UAP) [15]. The UAP effectively fools the victim model on any inputs because it captures the correlations among different regions of the decision boundary [15]. We conjecture that UAP can also capture the feature of backdoor neurons, resulting in smaller perturbations; see Fig. 1. This is because UAP utilizes the normals to the decision boundary in different regions of the decision space, i.e., the UAP finds the shortest path to cross the decision boundary. Backdoored models build shortcuts from all classes to the targeted class by the trigger. Therefore, UAPs from backdoored models are smaller than UAPs from clean models. USB requires less iteration and data as we directly use UAP to capture the potential backdoor. As UAP can be generalized across different networks, we only need to generate UAP once for similar models, greatly reducing the time requirement.

We evaluated USB on 240 models (150 on CIFAR-10 [11] with ResNet-18 [8], 45 on ImageNet [4] with Efficientnet-B0 [23], and 45 on CIFAR-10 [11] with VGG-16 [22]). As the results indicate, USB outperforms the latest detection

techniques [24], [7]. Our contributions are summarized as follows:

- We propose a novel detection method **USB** that utilizes the similarities between backdoor and UAPs. We show that a UAP with the same target as a backdoor attack is smaller than UAPs with a different target from the backdoor attack regarding the  $L_1$  norm.
- **USB** can detect stronger backdoor triggers for both patch-based (BadNet and Latent) and non-patch-based (Input-Aware Dynamic) backdoors. Existing methods tend to conduct reverse engineering from random noise, which may not work under advanced attacks. Our reversing process uses the target UAP as the starting point for initialization to avoid the local optimal triggers, as the UAP is closer to potential triggers.
- We conduct experiments on 240 models to assess our approach. We compare **USB** with **NC** and **TABOR** methods and **USB** provides competitive performance on various datasets compared to state-of-the-art methods.

## II. RELATED WORK

**Training-time Defenses.** Training-time defenses refer to defenses that are conducted during the training of the model, including detecting poisoned data points in training data [2], reducing the impact of poisoned data on training the model by differential privacy [14], input pre-processing [13], and randomized smoothing [20]. These methods take advantage of the difference between clean and poisoned data concerning the victim model. For a clean sample that originally belongs to the target class, the model recognizes it as the target class because the sample contains the features of the target class. For a poisoned sample, the backdoored model extracts trigger features for classification. However, training-time defenses require access to training data, which may not be feasible in cases where the model is pre-trained by a third party.

**Inference-time Defenses.** Inference-time defenses refer to defenses with access to the pre-trained model and a certain amount of clean data, including detection by reverse engineering of the backdoor trigger [24], [12], [5], pruning [26], and machine unlearning to remove the backdoor [24]. The pruning and machine unlearning aim to remove the backdoor by directly modifying the victim model, while the reverse engineering conducts detection and reconstructs the backdoor trigger. Reverse engineering methods, such as **NC** [24] and **TABOR** [7], take advantage of the behavioral characteristics of the backdoor itself. The backdoor builds a shortcut from within regions of the space belonging to each label into the region belonging to the target. For backdoored models, transforming input features of any class into features of the target class requires less perturbation than transforming into other classes, as the backdoor trigger must be small to remain stealthy. Reverse engineering searches for a trigger for each class of the model to be detected. Every searched trigger can mislead the model to output the corresponding class when applied to any inputs. If the model is backdoored, there will be an outlier trigger that is smaller than others.

---

### Algorithm 1 Computation of targeted UAP.

---

**Input:** Data points  $X$ , target class  $t$ , victim model  $f$ , desired  $l_p$  norm of the perturbation  $\delta$ , desired error rate  $e$   
**Output:** Targeted UAP  $v$

```

1: Initialize  $v \leftarrow 0$ 
2: while  $\text{Error}(X + v) \geq e$  do
3:   for  $x_i$  in  $X$  do
4:     if  $f(x_i + v) \neq t$  then
5:       Minimal perturbation that send  $x_i + v$  to class
        $t$ :
6:          $\Delta v_i \leftarrow \arg \min_r \|r\|_2 \text{ s.t. } f(x_i + v + r) = t$ 
7:          $v \leftarrow \Delta v_i$   $\triangleright$  Update the perturbation under
           limitation
8:     end if
9:   end for
10: end while
```

---



---

### Algorithm 2 Updating of targeted UAP.

---

**Input:** Data points  $X$ , target class  $t$ , victim model  $f$ , UAP  $v$ , Maximum iteration number  $m$ , learning rate  $lr$   
**Output:** Updated UAP  $v' = \text{pattern} \times \text{mask}$

```

1: Initialize trigger by  $v$ :  $\text{trigger} = \text{pattern} \times \text{mask} = v$ 
2: for  $i = 0$  to  $m$  do
3:    $x \subseteq X$   $\triangleright$  Take a batch of data,  $x$ , from  $X$  in order
4:    $x' = x \times (1 - \text{mask}) + \text{pattern} \times \text{mask}$   $\triangleright$  Apply
     pattern and mask to get perturbed input
5:    $\text{output} = f(x')$ 
6:    $\mathcal{L} = \mathcal{L}(\text{output}, t) - \text{SSIM}(x, x') + \text{norm}_{L1}(\text{mask})$ 
7:   Backward loss  $\mathcal{L}$  to update  $\text{mask}$  and  $\text{pattern}$ 
8:    $\text{mask} : \text{mask} \leftarrow \text{mask} - lr \times \nabla \text{mask}$ 
9:    $\text{pattern} : \text{pattern} \leftarrow \text{pattern} - lr \times \nabla \text{pattern}$ 
10:   $v' = \text{pattern} \times \text{mask}$ 
11: end for
```

---

## III. PROPOSED METHOD

### A. Threat Model

We consider defense against a backdoor attack under a white box scenario. The adversary (attacker) has white box access to the victim model and training data. The adversary injects backdoors through the dirty label attacks. The defender aims to detect these backdoors. In this paper, we consider an all-to-one setting where the triggers mislead all inputs of backdoored models to a single target class.

### B. Defense Overview

Our method consists of two main processes to detect whether there is a backdoor in the targeted model. First, we generate a targeted UAP for the victim model. The UAP is supposed to capture special neurons that can easily lead to misclassification. Then, we use an optimization process to update the UAP so that it can focus on the most important part. This “most important part” refers to the part of UAP most likely to cause misclassification. We generate and optimize targeted UAPs for every class of the model. We check whether

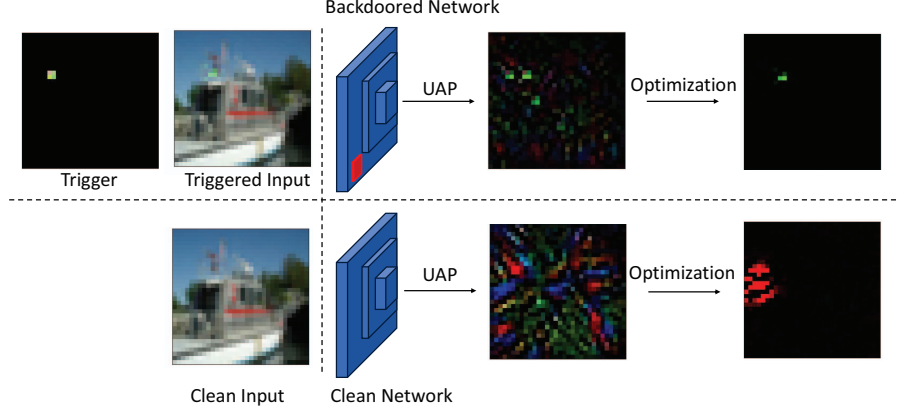


Fig. 2: The structure of USB. First, we generate targeted UAP for the clean model and backdoored model. Second, we optimize UAPs to reverse engineer the potential trigger. The pixel values of the two UAPs are scaled for visibility.

there is an outlier smaller than others from these UAPs to decide if the model is backdoored or not. Fig. 2 illustrates the framework of our defense.

### C. Targeted UAP

To work in the all-to-one setting, we modify the algorithm from [15] to generate targeted UAP, which misleads all inputs to the targeted class. Let us assume a well trained deep learning model  $f$  and  $K$  entries of training data  $D = \{(x_i, y_i)\}_{i=0}^{K-1}$  where  $x_i \in \mathbb{R}^{d_X}$  and  $y_i \in \{0, 1\}^N$ .  $N$  is the number of classes, and  $d_X$  is the input dimension. The targeted UAP algorithm aims to find a perturbation vector  $v$  that misleads the model  $f$  on most of the data points in  $D$  to a target class  $t$ . We use a very small number of data points  $X$  to work in a more realistic situation. Empirically, a size smaller than 1% of  $D$  can be enough for  $X$ . Then, the perturbation  $v$  should satisfy the following two constraints to ensure practicality. First, the generated perturbation  $v$  should successfully mislead the model  $f$ , i.e., the error rate should be larger than the desired threshold  $\theta$ :

$$Err(X + v) := \frac{1}{K} \sum_{i=0}^{K-1} r_i \geq \theta,$$

$$\text{where } r_i = \begin{cases} 1, & f(x_i + v) \neq f(x_i) \\ 0, & f(x_i + v) = f(x_i). \end{cases}$$

Second, the perturbation  $v$  should be imperceptible. Specifically, the norm of  $v$  should be smaller than the limit  $\delta$ :

$$P_{l,p,\delta}(v, \Delta v_i) = \arg \min_{\Delta v_i} \|v + \Delta v_i\|_2, s.t. \|\Delta v_i\|_p \leq \delta.$$

In Alg. 1, we iteratively go through every data point in  $X$  to update UAP from scratch. At each iteration, the algorithm searches for the minimal perturbation that sends  $x_i + v$  to the target class. Then, the error rate of inputting  $X + v$  to  $f$  should be larger than the desired threshold  $\epsilon$  so that  $v$  is

effective as a targeted UAP. This is feasible by solving the following optimization problem:

$$\Delta v_i \leftarrow \arg \min_r \|r\|_2 \text{ s.t. } f(x_i + v + r) = t.$$

Following the algorithm in [15], this search optimization is implemented by DeepFool [16].

### D. UAP Optimization

UAP is a collection of normals<sup>1</sup> to the decision boundary in different regions [15], including but not only the regions where the trigger is located. Therefore, we further update the targeted UAP through an optimization phase. The optimization objective is formalized as a loss function:

$$\mathcal{L} = \mathcal{L}_{ce}(\text{output}, t) - SSIM(x, x') + norm_{L1}(\text{mask}), \quad (1)$$

where  $\mathcal{L}_{ce}$  refers to the cross-entropy loss. The structural similarity index measure (SSIM) measures the similarity between images [25].

The details are provided in Alg. 2. The optimization achieves two goals: (1) it makes the targeted UAP focus on more important pixels, and (2) it ensures that the UAP can mislead the victim model. The first goal is embedded in the loss by a trigger and a mask, i.e., minimizing the mask by decreasing  $norm_{L1}(\text{mask})$ . At the beginning of Alg. 2, the trigger and mask are initialized by the targeted UAP. The trigger is a copy of the UAP, and the mask has the same shape as the trigger. In every iteration, the algorithm takes a batch of data from  $X$  instead of using the whole  $X$ . The next iteration will use the data after  $x$  in order. Thus, all data in the  $X$  will be used. The purpose is to reduce the running time of each iteration. When initializing, elements in the mask are set to one, such that the first  $x'$  (line 4 in Alg. 2) equals  $x + v$ . Then, the trigger and mask will be updated according to the gradients generated on the trigger and mask during computation. Note that  $f$  will not be modified in Alg. 2. This optimization may

<sup>1</sup>Minimal distance from the region to the decision boundary.

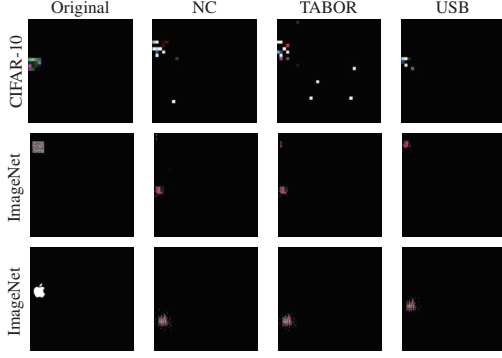


Fig. 3: Examples of the original and reversed triggers by NC, TABOR, and USB for CIFAR-10 and ImageNet.

introduce excessive perturbations, so we use SSIM to keep  $x + v$  similar to the original image  $x$ . Finally, we decrease cross-entropy loss between output ( $f(x')$ ) and target class ( $t$ ) for the second goal.

**Detection.** Based on the above discussion, we can generate targeted UAPs for all classes to detect whether a model has a backdoor. Given a model  $f$  that may have been injected with a backdoor, we generate  $N$  targeted UAPs corresponding to every class, i.e.,  $\{v_i\}_0^{N-1}$ . Then, these UAPs are optimized by Alg. 2 to locate the position of the potential trigger. We use  $\{v'_i\}_0^{N-1}$  to indicate optimized UAPs. As mentioned before, misleading a backdoored model to the target class needs smaller perturbation compared to untarget classes. Therefore, if  $f$  is backdoored on class  $t_b$ , the size of  $v'_{t_b}$  will be smaller than other UAPs in  $\{v'_i\}_0^{N-1}$ . The size of UAPs is quantified by the  $L_1$  norm. Empirically, the  $L_1$  norm of the targeted UAP for the backdoored class is more than one order of magnitude smaller than targeted UAPs for other classes without a backdoor. For example, for a ResNet-18 model with a BadNet backdoor on class 0, the  $L_1$  norm  $v'_0$  generated by USB is 4.49, and the average  $L_1$  norm of the other classes is 53.76.

#### IV. EVALUATION

We provide the experimental results for USB and compare them with NC [24] and TABOR [7], which are the typical state-of-the-art methods. Experiments are conducted with TrojanZoo [18]. We use different random seeds for every trained model.

##### A. Experimental Setup

**Models, Datasets, and Backdoor.** We use ResNet-18 [8] and VGG-16 [22] for CIFAR-10 [11], and Efficientnet-B0 [23] for ImageNet [4]. We use BadNet [6], Latent Backdoor [27], and IAD [17] to inject backdoor into victim models. The triggers are generated in different positions and random colors.

**Hyperparameters.** For Alg. 1, we set the desired error rate to  $e = 0.6$ .  $X$  contains 300 data points. In our experiments, these are the minimum hyperparameters to obtain effective UAPs.

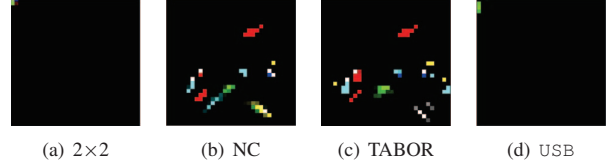


Fig. 4: An example visualization of the original and reversed triggers by NC, TABOR, and USB for CIFAR-10.

The  $\delta$  is set to 10, following experiments in [15] to ensure the UAP is imperceptible.

For Alg. 2, the maximum iteration number is  $m = 500$ . The learning rate (lr) is  $lr = 0.1$ , and the optimizer is Adam (for detection) with  $\beta = (0.5, 0.9)$ . The hyperparameters to train clean and backdoored models are: batch size=96, lr=0.01, epoch=50, poison percent=0.01. Hyperparameters not mentioned are the default ones from TrojanZoo [18].

**Evaluation.** Following the previous work in [5], we designed two metrics to evaluate the defense performance: model detection and target class detection. We check whether a model is correctly identified as a clean or backdoored model. Then, for backdoored models, we check whether reverse engineering correctly identifies the target class. In Tab. I, II, and III, *Clean* and *Backdoored* under *Model Detection* refer to the cases whether the detection identifies a model as clean or backdoored. For *Target Class Detection*, we have three categories: (i) *Correct* means the detection method identifies the true target class of a backdoored model, (ii) *Correct Set* refers to the case where the detection method identifies multiple backdoors on different classes, including the true target class, and (iii) *Wrong* refers to the case where the detection method successfully identifies a backdoored model but with wrong target class(es).

##### B. Experimental Results

This section considers BadNets only as the attack method. Tab. I shows the detection results for CIFAR-10 (We also provide results on VGG architecture, Tab. V, and GTSRB dataset, Tab. VI, in the appendix). For the backdoored models, USB achieves a higher accuracy (98%) for detecting backdoored models compared to NC (93%) and TABOR (92%). We believe that the misclassifications in NC and TABOR are caused by capturing the class's unique features rather than the trigger, illustrated in Fig. 4. We show more reversed triggers in Fig. 3 and Fig. 6 in the Appendix.

As ImageNet contains a large number of images, it is difficult to train many models on it. Thus, we use a subset of ImageNet, which contains ten classes. Each class has 1301 images. Tab. II shows the results for detecting backdoors for Efficientnet-B0 [23] trained with the subset of ImageNet [4]. Due to the larger image size and model architecture, we use 500 images for data points  $X$  in Alg. 1 and 2.



Model&Trigger	Accuracy	ASR	Method	Reversed Trigger $L_1$ norm	Model Detection		Target Class Detection		
					Clean	Backdoored	Correct	Correct Set	Wrong
Clean	85.38	N/A	NC	51.59	50	0	N/A	N/A	N/A
			TABOR	55.09	50	0	N/A	N/A	N/A
			USB	48.99	50	0	N/A	N/A	N/A
Backdoored ( $2 \times 2$ trigger)	83.43	95.04	NC	8.72	5	45	44	1	0
			TABOR	9.26	5	45	44	1	0
			USB	9.83	1	49	45	4	0
Backdoored ( $3 \times 3$ trigger)	83.59	97.57	NC	8.89	2	48	48	0	0
			TABOR	10.06	3	47	47	0	0
			USB	12.02	1	49	49	0	0

TABLE I: Detection evaluation on CIFAR-10 where each case consists of 50 trained models.


Model&Trigger	Accuracy	ASR	Method	Reversed Trigger $L_1$ norm	Model Detection		Target Class Detection		
					Clean	Backdoored	Correct	Correct Set	Wrong
Backdoored ( $20 \times 20$ trigger)	70.94	76.67	NC	276.78	0	15	14	1	0
			TABOR	271.83	0	15	12	2	1
			USB	461.32	0	15	14	1	0
Backdoored ( $25 \times 25$ trigger)	69.7	78.46	NC	347.48	0	15	13	2	0
			TABOR	341.47	2	13	13	0	0
			USB	547.56	0	15	15	0	0
Backdoored 	70.91	80.02	NC	396.72	1	14	14	0	0
			TABOR	406.1	3	12	12	0	0
			USB	621.0	1	14	14	0	0

TABLE II: Detection evaluation on ImageNet where each case consists of 15 trained models. The apple is a backdoor trigger.

Model&Trigger	Accuracy	ASR	Method	Reversed Trigger $L_1$ norm	Model Detection		Target Class Detection		
					Clean	Backdoored	Correct	Correct Set	Wrong
Clean	91.59	N/A	NC	40.78	15	0	N/A	N/A	N/A
			TABOR	48.53	14	1	0	0	1
			USB	47.53	15	0	N/A	N/A	N/A
Latent Backdoor ( $4 \times 4$ trigger)	87.20	99.66	NC	19.71	4	11	10	1	0
			TABOR	20.68	4	11	11	0	0
			USB	12.37	1	14	13	1	0
Input Aware Dynamic ( $32 \times 32$ trigger)	89.46	90.43	NC	0.0	15	0	N/A	N/A	N/A
			TABOR	1.8	15	0	N/A	N/A	N/A
			USB	0.13	0	15	15	0	0

TABLE III: Detection evaluation by stronger backdoor attacks on VGG-16 trained with CIFAR-10.

### C. Stronger Backdoor Attacks

Tab. III shows detection results on Latent Backdoor [27] and IAD attack [17]. The trigger size for Latent Backdoor is  $4 \times 4 \times 3$ . Due to the IAD attack’s characteristics, we use  $32 \times 32 \times 3$  trigger size (the size of input images). The motivation is to show the generalization of USB under stronger attacks besides BadNets [6], especially since IAD is non-patch-based. IAD also generates different triggers according to different inputs.

According to Tab. III, NC and TABOR show worse performance compared to detection results for BadNets, while USB still precisely detects most of the backdoored models. NC and TABOR do not work under the IAD attack, but USB detects such backdoors with the true target class. The reason is that NC-style methods do not directly optimize the pattern of the trigger. Indeed, they mainly optimize the mask that will be applied to the pattern. Moreover, IAD attacks design subtle

triggers with specific features related to inputs, which is more difficult for an optimization procedure from random points.

### D. Time Consumption

NC and TABOR require a large number of iterations to conduct detection. We evaluate the time consumption of NC, TABOR, and USB when conducting detection with Efficientnet-B0 on ImageNet. When detecting backdoored models with  $20 \times 20$  trigger, the average time consumption (in seconds) for NC, TABOR, and USB are 1,154.02, 2,129.40, and 267.12, respectively. USB requires less time when reverse engineering the potential triggers. Although USB needs to generate targeted UAP, the UAP can be used for different models with similar architectures [15], as observed in our experiments. Thus, we only need to generate it once. Tab. IV shows the details of time consumption compared to NC and TABOR.


Model	Method	GPU Time [m:s] in every class									
		0	1	2	3	4	5	6	7	8	9
Backdoored (20 × 20 trigger)	NC	23:16	24:18	24:32	23:39	24:48	23:35	23:15	23:34	23:41	24:10
	TABOR	33:54	37:24	34:19	35:51	33:59	36:45	34:23	36:47	35:4	36:23
	USB	4:26	4:26	4:27	4:30	4:26	4:26	4:26	4:26	4:26	4:26
Backdoored (25 × 25 trigger)	NC	23:35	24:38	25:11	24:3	24:58	24:19	24:29	23:54	24:29	23:6
	TABOR	48:23	47:24	48:48	48:41	48:38	47:41	48:27	49:10	48:48	47:45
	USB	4:44	4:42	4:44	4:44	4:49	4:48	4:48	4:54	4:50	4:45
	NC	19:1	18:22	20:47	18:5	18:48	21:27	18:54	18:30	20:26	17:57
	TABOR	48:52	47:16	49:0	48:39	48:54	47:40	48:34	48:55	48:55	47:50
	USB	4:27	4:26	4:27	4:30	4:26	4:26	4:26	4:26	4:26	4:25

TABLE IV: Running time results of backdoor detection for Efficientnet-B0 [23]. Each result is the average of detection on 15 models.

### E. Discussion

To explain USB, we analyze triggers reversed from every class using MNIST and a simple CNN architecture (see Appendix C for details) with two convolutional layers and two fully connected layers. We remove the constraint on the mask size to search for as powerful features as possible. We replace  $\mathcal{L}$  in Alg. 2 by:  $\mathcal{L} = \mathcal{L}_{ce}(\text{output}, t) - SSIM(x, x')$ . Under this setting, we train a backdoored model with BadNet. Then, we conduct reverse engineering for all classes. According to results in Fig. 5, the optimization with the loss  $\mathcal{L}$  tends to learn unique class features for the clean class and the trigger features for the backdoor class. This is expected as we only have a backdoor on the target class, i.e., class 1. In this simplified situation, for clean classes without backdoors, only the unique class features allow the model to recognize that an input belongs to the class. For the class injected with the backdoor, the model will recognize the input as the backdoor target based on the unique feature of the target and the feature of the backdoor trigger.

Reverse engineering requires a choice between the unique class feature and the feature of the backdoor trigger. Regarding relatively simple features, the trigger feature is stronger than class features when training with poisoned data. Reverse engineering can find a small perturbation with strong features enough to mislead the model according to learning objectives in the loss function. However, in scenarios such as training with GTSRB or larger datasets, there might be strong features that can generate larger perturbations with a similar size to backdoor triggers. This is why NC, Tabor, and USB provide more incorrect results when using GTSRB as training data in Tab. VI.

### V. LIMITATIONS AND FUTURE WORK

Although USB works effectively in detecting backdoors, there are still limitations. First, the optimized triggers are directly related to the data  $X$  used by the optimization process. Generating these triggers relies on the gradient when inputting an image from  $X$  and trigger to the model. Therefore, if the data  $X$  is collected from a different distribution from the training data, existing detection methods, including USB, NC, and TABOR, may fail. This is also why we try to use fewer data in USB. Second, it might be difficult to generate targeted

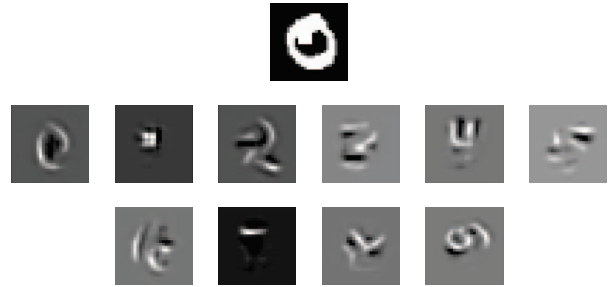


Fig. 5: USB reverse engineering for 10 classes on MNIST. The true backdoor target is class 1. The first one is the clean image carrying the trigger. The rest are reversed triggers from class 0 to 9.

UAP for every class when the number of classes is high. For example, ImageNet contains 1,000 classes. It is more time-consuming to generate perturbation for inputs of all classes to the target class when the number of classes is large.

This limitation could be the starting point of future work. Currently, our algorithm (Alg. 1) only searches for small perturbations to generate targeted UAP. This can mislead the victim model. If we can directly search for targeted UAP according to the backdoored neurons in the model, the number of iterations for searching can be significantly reduced. In addition, reverse engineering naturally requires fewer data if it has knowledge about backdoor-related neurons, which helps solve the first limitation. A key problem for future work could be identifying those special neurons.

### VI. CONCLUSIONS AND FUTURE WORK

This paper proposes USB to detect potential backdoors. USB uses targeted UAP to capture sensitive features created by backdoors. We further optimize the UAP to generate the backdoor trigger. We run extensive experiments on several datasets to evaluate our method. Among the 175 backdoored models on several datasets, we successfully identified 171 backdoored ones and outperformed state-of-the-art methods. Further investigation is needed regarding optimizing UAP according to backdoored neurons in the victim model, which can significantly reduce the optimization time.

## REFERENCES

- [1] BigML. Bigml.com. <https://bigml.com>, 2011. Accessed: 2023-01-18.
- [2] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian M. Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *SafeAI Workshop @ AAAI 2018*, 2018.
- [3] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.
- [4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.
- [5] Yinpeng Dong, Xiao Yang, Zhijie Deng, Tianyu Pang, Zihao Xiao, Hang Su, and Jun Zhu. Black-box detection of backdoor attacks with limited information and data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16482–16491, October 2021.
- [6] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.
- [7] Wenbo Guo, Lun Wang, Yan Xu, Xinyu Xing, Min Du, and Dawn Song. Towards inspecting and eliminating trojan backdoors in deep neural networks. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 162–171, 2020.
- [8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [9] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia*, MM '14, page 675–678, New York, NY, USA, 2014. Association for Computing Machinery.
- [10] Jing Yu Koh. Tensorflow model zoo. <https://modelzoo.co/>, 2018. Accessed: 2023-01-18.
- [11] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [12] Yingqi Liu, Wen-Chuan Lee, Guan hong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS '19*, page 1265–1282, New York, NY, USA, 2019. Association for Computing Machinery.
- [13] Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural trojans. In *2017 IEEE International Conference on Computer Design (ICCD)*, pages 45–48, 2017.
- [14] Yuzhe Ma, Xiaojin Zhu, and Justin Hsu. Data poisoning against differentially-private learners: Attacks and defenses. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI'19*, page 4732–4738. AAAI Press, 2019.
- [15] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [17] Tuan Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3454–3464. Curran Associates, Inc., 2020.
- [18] Ren Pang, Zheng Zhang, Xiangshan Gao, Zhaohan Xi, Shouling Ji, Peng Cheng, and Ting Wang. TrojanZoo: Towards unified, holistic, and practical evaluation of neural backdoors. In *Proceedings of IEEE European Symposium on Security and Privacy (Euro S&P)*, 2022.
- [19] Mauro Ribeiro, Katarina Grolinger, and Miriam A.M. Capretz. Mlaas: Machine learning as a service. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 896–902, 2015.
- [20] Elan Rosenfeld, Ezra Winston, Pradeep Ravikumar, and Zico Kolter. Certified robustness to label-flipping attacks via randomized smoothing. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8230–8241. PMLR, 13–18 Jul 2020.
- [21] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [22] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- [23] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019.
- [24] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723, 2019.
- [25] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.
- [26] Dongxian Wu and Yisen Wang. Adversarial neuron pruning purifies backdoored deep models. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 16913–16925. Curran Associates, Inc., 2021.
- [27] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y. Zhao. Latent backdoor attacks on deep neural networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS '19*, page 2041–2055, New York, NY, USA, 2019. Association for Computing Machinery.

## APPENDIX

### A. Detection Results on VGG-16

In Tab. V, we show the results of detecting backdoors for VGG-16 models trained with CIFAR-10. We use the same experimental settings as that in the experiment section. We also study Latent Backdoor [27] beside BadNet attack.

### B. GTSRB

The results for GTSRB are shown in Tab. VI. On clean models, USB, NC, and TABOR all have incorrect results, as the number of classes in GTSRB is significantly larger than that of CIFAR-10. Compared to the  $L_1$  norm of NC and TABOR, USB provides a much smaller norm value because the reversed trigger is optimized from targeted UAP.

### C. Details of the Basic Model

To reduce the impact of complex features and many model parameters, we use MNIST and a basic CNN architecture with two convolutional layers (followed by the ReLU activation function and a 2D average pooling layer) and two fully connected layers. The input channel, output channel, and kernel size for the two convolutional layers are (1, 16, 5) and (16, 32, 5). The input and output channels for the two fully connected layers are (512, 512) and (512, 10). The model is trained using batch size=128, epochs=40, and poisoned rate=0.05.

Model	Accuracy	ASR	Method	Reversed Trigger $L_1$ norm	Model Detection		Target Class Detection		
					Clean	Backdoored	Correct	Correct Set	Wrong
Clean	91.59	N/A	NC	40.78	15	0	N/A	N/A	N/A
			TABOR	48.53	14	1	0	0	1
			USB	47.53	15	0	N/A	N/A	N/A
Backdoored ( $2 \times 2$ trigger)	88.28	99.39	NC	5.43	0	15	14	1	0
			TABOR	5.32	0	15	15	0	0
			USB	3.5	0	15	15	0	0
Backdoored ( $3 \times 3$ trigger)	88.30	99.77	NC	6.60	1	14	13	1	0
			TABOR	6.98	0	15	14	1	0
			USB	7.0	0	15	14	1	0

TABLE V: Detection evaluation on VGG-16 trained with CIFAR-10 where each case consists of 15 trained models.

Model	Accuracy	ASR	Method	Reversed Trigger $L_1$ norm	Model Detection		Target Class Detection		
					Clean	Backdoored	Correct	Correct Set	Wrong
Clean	83.96	N/A	NC	181.17	12	3	N/A	N/A	N/A
			TABOR	185.21	13	2	N/A	N/A	N/A
			USB	39.8	12	3	N/A	N/A	N/A
Backdoored ( $2 \times 2$ trigger)	80.85	85.06	NC	13.36	0	15	13	2	0
			TABOR	37.02	0	15	13	2	0
			USB	10.86	3	12	12	0	0
Backdoored ( $3 \times 3$ trigger)	80.24	93.52	NC	14.78	0	15	13	2	0
			TABOR	15.11	0	15	13	2	0
			USB	12.02	2	13	13	0	0

TABLE VI: Detection evaluation on ResNet-18 trained with GTSRB where each case consists of 15 trained models.

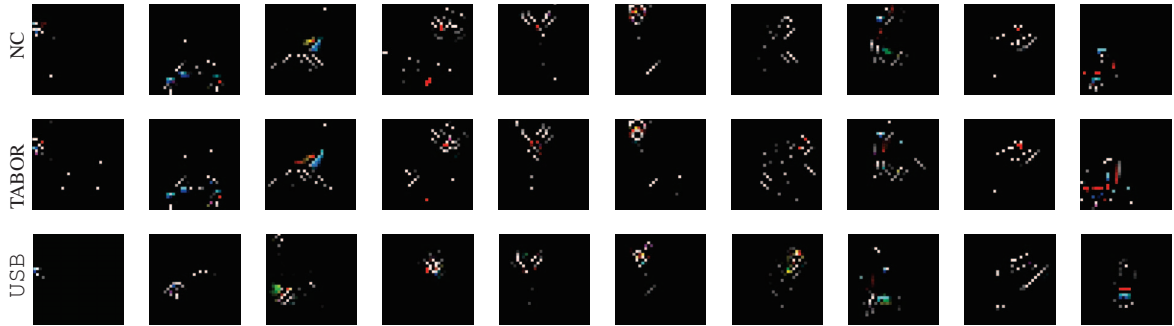


Fig. 6: Reversed triggers from class 0 to 9.