Gradient Shaping: Enhancing Backdoor Attack Against Reverse Engineering

Rui Zhu*, Di Tang*, Siyuan Tang*, Zihao Wang*, Guanhong Tao†, Shiqing Ma‡, Xiaofeng Wang*, Haixu Tang*
*Indiana University Bloomington
Emails: zhu11@iu.edu, tangd@iu.edu, tangsi@iu.edu, zwa2@iu.edu, xw7@indiana.edu, hatang@indiana.edu
†Purdue University
Email: taog@purdue.edu
‡University of Massachusetts Amherst
Email: Shiqingma@umass.edu

***Abstract*—Most existing methods to detect backdoored machine learning (ML) models take one of the two approaches: trigger inversion (aka. reverse engineer) and weight analysis (aka. model diagnosis). In particular, the gradient-based trigger inversion is considered to be among the most effective backdoor detection techniques, as evidenced by the TrojAI competition [14], Trojan Detection Challenge [1] and backdoorBench [24]. However, little has been done to understand why this technique works so well and, more importantly, whether it raises the bar to the backdoor attack. In this paper, we report the first attempt to answer this question by analyzing the change rate of the backdoored model's output around its trigger-carrying inputs. Our study shows that existing attacks tend to inject the backdoor characterized by a low change rate around trigger-carrying inputs, which are easy to capture by gradient-based trigger inversion. In the meantime, we found that the low change rate is not necessary for a backdoor attack to succeed: we design a new attack enhancement method called *Gradient Shaping* (GRASP), which follows the opposite direction of adversarial training to reduce the change rate of a backdoored model with regard to the trigger, without undermining its backdoor effect. Also, we provide a theoretic analysis to explain the effectiveness of this new technique and the fundamental weakness of gradient-based trigger inversion. Finally, we perform both theoretical and experimental analysis, showing that the GRASP enhancement does not reduce the effectiveness of the stealthy attacks designed to evade the backdoor detection methods based on weight analysis, as well as other backdoor mitigation methods without using detection.**

I. Introduction

Critical to trustworthy AI is the trustworthiness of machine learning (ML) models, which can be compromised by malevolent model trainers, evil-minded training data providers, or any parties with access to any link on the ML supply chain (e.g., pre-trained models) to inject a backdoor (aka., trojan). A backdoored model is characterized by strategic misclassification of the input carrying a unique pattern called *trigger*: e.g., special glasses worn by a masquerader to impersonate an authorized party against a compromised facial-recognition system. So the assurance of ML models can only be upheld by effectively detecting those backdoored models, which have been intensively studied in recent years. Existing backdoor defense methods have been reviewed by an SoK paper [23]: among seven general defense strategies, two are based on backdoor detection, which uses either the trigger inversion (aka. trigger synthesis) or weight analysis techniques (aka. model diagnosis) [45][28][39][5][47]. The most concrete progress in the backdoor detection has been at least partially attributed to *trigger inversion* related techniques, as evidenced in the TrojAI competition [14] (9 out of 11 rounds won by inversion approaches, the rest two won by weight analysis) and the BackdoorBench project [12] (leading performers are mostly gradient-based trigger inversion). However, little has been done to understand whether these approaches raise the bar to the backdoor attacks or are just another porous defense line permeable by the knowledgeable adversary.

Achilles' heel of gradient-based optimization. Trigger inversion is a technique that automatically recovers a pattern causing an ML model to misclassify the pattern-carrying input. Such a pattern is considered a putative trigger and utilized to determine whether the model is indeed backdoored. This reverse-engineering step mostly relies on gradient descent, which seeks the greatest tendency towards misclassification following the opposite direction of the model's gradient with regard to its input. A prior study shows that almost all proposed trigger inversion approaches are gradient-based [23]. Although gradient-based optimization can converge to a local optimum, this convergence is contingent upon selecting a proper size for each search step and a proper initialization. In the presence of a function with low robustness around trigger-inserted inputs (e.g., the one having a steep slope (large changing rate)), a large step size could overshoot the local minimum for the trigger which leads to misclassification. On the other hand, a small step size could render the convergence process exceedingly slow and increase the probability that the optimizer converges to another local minimum, practically thwarting any trigger inversion attempt. So a fundamental question not asked before is why gradient-based reverse engineering works so well on the backdoors injected using today's techniques and whether a more powerful backdoor capable of defeating the inversion can be injected under practical threat models.

Analysis and findings. To answer this question, we conducted the first study to understand the limitations of trigger inversion. Our research shows that today's backdoor injection techniques,

Network and Distributed System Security (NDSS) Symposium 2024
26 February - 1 March 2024, San Diego, CA, USA
ISBN 1-891562-93-2
https://dx.doi.org/10.14722/ndss.2024.24450
www.ndss-symposium.org

both loss-function manipulation, and data poisoning, turn out to be quite amenable to gradient-based optimization. Actually, given the relatively simple features that characterize today's triggers (e.g., geometric shapes), a backdoor learned could be more robust to the noise added to its trigger than the benign task the infected model claims to perform, as observed in our experiments: we found that oftentimes, backdoors tend to be more resilient to the noise than the primary task to the perturbation on its features (Section III).

This observation indicates that the backdoor can be invoked by not only the trigger but a wide range of its variations. Therefore, the average change rate of the backdoored model around trigger-inserted inputs for recognizing a trigger cannot be too high, which can be easily captured with a relatively larger scope of search step size and initialization selection. This explains why trigger inversion works so well in backdoor detection. However, a slow change rate (or high trigger effective radius) is *not* required for a backdoor attack to succeed. Our research shows that the change rate can be increased through data contamination without undermining the effectiveness of the backdoor attack. In our research, we designed a simple algorithm that could enhance the backdoor attack, called gradient shaping (GRASP) that utilizes both mislabeled data and correctly labeled data with noised triggers to contaminate the training set, in an opposite way to the augmentation training [40], so as to narrow down the variation of the trigger pattern capable of invoking the backdoor. We theoretically analyze this approach and show that it effectively raises the change rate, thereby weakening the detection ability from trigger inversion.

It is worth noting that GRASP represents a different type of backdoor attack compared with the stealthy backdoors proposed recently (e.g., [7] [37]). Existing stealthy backdoor methods attempt to devise specific triggers, often dependent on the target neural network model so that they are hard to detect and mitigate by defense methods. GRASP, on the other hand, is a generic trigger injection method that injects any trigger designed by the attacker into a target model so that the trigger is harder to detect and mitigate by the trigger inversion-based backdoor defenses. As a result, GRASP can be combined with existing stealthy backdoor methods to enhance their capability to evade the trigger inversion-based defenses. Our studies show that existing backdoor attacks less capable of evading trigger inversion can be boosted by GRASP to easily defeat most representative inversion protection, including Neural Cleanse (NC) [45], tabor [11], k-arm [39], pixel [43], rendering them incapable of capturing any trigger of a backdoored model.

We also perform a theoretical and experimental analysis (Appendix K and Section V-C, respectively.) to show that GRASP does not make the backdoor more vulnerable to weight analysis, which is the other mainstream technique for backdoor detection. In particular, our experiment shows that the GRASP enhancement does not decrease the effectiveness of the backdoor attacks such as DFST [6], AB [35], and DEFEAT [52] against the weight analysis-based detection. Finally, our study demonstrates that the effectiveness of GRASP against trigger inversion does not make the enhanced attacks more vulnerable to other backdoor mitigation or unlearning techniques, such as Fine-purning [27], NAD [22], Gangsweep [53], DBD [12], RAB [46], and ABL [21].

Contributions. The contributions of the paper are outlined below:

• *First in-depth analysis on trigger inversion*. We report the first in-depth analysis that explains why trigger inversion works so well on backdoor detection. This leads to the discovery of the fragility of the advance we made in this area, given the observation that the weakness of today's trigger injection can be addressed without undermining the effectiveness of the backdoor.

• *New backdoor injection technique*. Our new understanding of trigger inversion has been made possible by a new backdoor injection technique, which exploits the fundamental limitation of gradient-based optimization and works under realistic threat models. As such, this method can enhance existing backdoor attacks, making it more effective in evading trigger inversion, but not less effective in evading the weight analysis-based detection and other defenses.

II. Background

A. Modeling Backdoor Attacks

In a backdoor attack, the adversary intends to inject a backdoor (Trojan) into the target ML model for the purpose of causing the model to produce desired outputs for trigger-inserted inputs. In our research, without loss of generality, we focus on the backdoor attacks against image classification models.

Classification model. A classification model is represented as $f(\cdot) : \mathcal{X} \mapsto \bar{\mathcal{Y}}$, while $\arg\max f(\cdot) \in \mathcal{Y}$ represents the predicted label of the given input. Specifically, $\mathcal{X} \subseteq \mathbb{R}^m$, $\bar{\mathcal{Y}} \subseteq [0,1]^K$, $\mathcal{Y} = \{0, 1, ..., K\}$, and K is the number of classes. We refer to f_D as a model trained on dataset D. Generally, we consider a dataset D containing n independent training samples, i.e., $D = \{x_i, y_i\}_{i=1}^n$, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$.

Backdoor injection. Backdoor injection is modeled as a process that introduces a backdoor into the target model, causing the backdoored model to produce adversary-desired outputs for trigger-inserted inputs. Formally, following the definition of Neural Cleanse [45], we model the trigger as a pair $(\boldsymbol{M}, \boldsymbol{\Delta})$ of trigger mask $\boldsymbol{M}$ and trigger pattern $\boldsymbol{\Delta}$. A trigger-inserted input $A(x, \boldsymbol{M}, \boldsymbol{\Delta})$ is the output of applying the amending function A on a benign input x with a given trigger pair $(\boldsymbol{M}, \boldsymbol{\Delta})$. Specially, we consider a well-accepted amending function $A(x, \boldsymbol{M}, \boldsymbol{\Delta}) = (1 - \boldsymbol{M}) \cdot x + \boldsymbol{M} \cdot \boldsymbol{\Delta}$. And we refer to m^* as the l_1 norm of the trigger mask $\boldsymbol{M}$, i.e., $m^* = \|\boldsymbol{M}\|_1$. In this paper, we focus on targeted backdoor scenarios, where adversaries aim to mislead the target model to predict the target labels for trigger-inserted inputs. We use y_t and y_s to represent the target label and the source label (the true label) of an input x, respectively.

B. Modeling Trigger Inversion

Trigger inversion aims to recover a putative trigger, which is the trigger reconstructed through reverse engineering, for a backdoor (Section II-A) and then evaluate the trigger on benign inputs to verify its backdoor effect (misclassifying such inputs to a target label). We model the trigger recovery process as

an optimization problem: finding the trigger that optimizes an objective function for a given model.

Objective optimization function. Formally, following the trigger modeling (Section II-A), we model the problem of trigger inversion as finding a trigger pair $(\boldsymbol{M}, \boldsymbol{\Delta})$ that minimizes the following objective function over a set of inputs $\boldsymbol{X}$ for a given classification model $z(f(\cdot))$:

$$\min_{\boldsymbol{M},\boldsymbol{\Delta}} \sum_{\boldsymbol{x}\in\boldsymbol{X}} \ell(y_t, f(A(\boldsymbol{x}, \boldsymbol{M}, \boldsymbol{\Delta}))) + \lambda \cdot \zeta(\boldsymbol{M}, \boldsymbol{\Delta}) \quad (1)$$

where $\ell(\cdot,\cdot)$ is a loss function, y_t is the target label, $A(\boldsymbol{x}, \boldsymbol{M}, \boldsymbol{\Delta})$ is the amending function, $\zeta(\cdot,\cdot)$ is a regularization penalty function for the trigger pair $(\boldsymbol{M}, \boldsymbol{\Delta})$, and λ is the weight of the regularization penalty. For example, Neural Cleanse (NC) uses square loss as the loss function and m^* (l_1 norm of $\boldsymbol{M}$) as the regularization penalty function.

Gradient-based solution. The objective function (Eq. 1) contains an empirical risk term (the first one) and a penalty term (the second one). The optimization of such objective functions has been well-studied in the context of neural networks. Particularly, Stochastic Gradient Descent (SGD) has been tremendously successful in finding solutions to such an optimization problem. Hence, it is not surprising that SGD has demonstrated its power in trigger inversion [45], [11], [39], [28]. However, in general, gradient-based solutions such as SGD tend to finds local minima due to the non-convex nature of the objective function, which may possess numerous local minima. In the context of trigger inversion, the Attack Success Rate (ASR) is used to measure the effectiveness of a reconstructed trigger, and the trigger with the highest ASR is chosen as the most plausible trigger.

C. Preliminary

Here, we need to define some terms used throughout the rest of the paper. First, given a backdoored model $f'(\cdot)$ and the corresponding trigger insert function $A(x, \boldsymbol{\Delta}, \boldsymbol{M})$, we define the sample-specific trigger effective radius of a backdoored model in Definition 1, and the overall trigger effective radius of a backdoored model in Definition 2. Intuitively, the sample-specific trigger effective radius denotes the minimum perturbation needed on the trigger area of a trigger-inserted input to change the prediction for that input, while the overall trigger effective radius for a backdoored model represents a general measure across all trigger-inserted inputs, which can be estimated by averaging the sample-specific trigger effective radius over all samples in a dataset.

Definition 1 (Sample specific trigger effective radius). *Given a benign input $x \in \mathcal{X}^m$, and the corresponding trigger-inserted input $x' = A(x, \Delta, M)$, for each entry in x':*

$$x'^{(i)} = \begin{cases} x^{(i)} & \boldsymbol{M}^{(i)} = 0 \\ \boldsymbol{\Delta}^{(i)} & \boldsymbol{M}^{(i)} = 1 \end{cases} \quad (2)$$

where $i \in \{1, .., m\}$, and $\boldsymbol{M}$ is the trigger mask matrix. In $f'(\cdot)$, the sample-specific trigger effective radius is measured on a trigger-carrying input x' (denote as $r_t^{x'}$), which is defined as the smallest perturbation ϵ on the trigger containing subspace ($\{x'^{(i)} | \boldsymbol{M}^{(i)} = 1\}$) such that $\arg\max f(x') \neq \arg\max f(x' + \epsilon)$

Similarly, we can approximate the overall trigger effective radius as below:

Definition 2 (Overall trigger effective radius). *Given a dataset $X \in \mathcal{X}^{n\times m}$, where every data point belongs to the source class. Let $X' \in \mathcal{X}^{n\times m}$ denote the dataset after inserting a trigger into each input in X. The overall trigger effective radius of $f'(\cdot)$ (denote as r_t), is approximated by averaging all $r_t^{x'_i}$ for each $x'_i \in X'$, we have $r_t \approx \frac{\sum_i^n r_t^{x'_i}}{n}$*

We will use the trigger effective radius to represent the overall trigger effective radius.

D. Threat Model

We consider a black-box threat model similar to that used in the BadNet project [10], as elaborated below:

Attacker's goal. We consider the adversary who wants to inject targeted backdoors so as to mislead an ML model to predict target labels for the trigger-inserted inputs.

Attacker's capabilities. We consider the black-box data-poisoning attack, where we assume that the adversary can inject data into the training set but does not know other training data or the parameters of the target model. An example is federated learning [48], in which some data contributors may be untrusted.

Defender's goal. The defender aims to detect backdoored ML models and further suppress the backdoor effects in these models. The focus of our research is the detection methods based on trigger inversion.

Defender's capabilities. We assume that the defender has full access to the target model, and owns a small set of benign inputs for trigger reconstruction. Also, we assume that the defender does not know whether a target model is infected, what the backdoor source and target labels would be and what triggers look like.

Notably, previous backdoor attacks, especially those aiming to make the backdoor stealthier, such as those in [3], [7], manipulate the training process, such as the loss function, to achieve a backdoor attack. such as the loss function to achieve a backdoor attack. Consequently, the threat model of these methods requires the attacker to control the model's training process. Researches like [32], [35], [29] leveraged the information embedded in the feature space of the victim model to design the backdoor, thus assuming a white-box access to the model in their threat model. Alternatively, other studies [41] require the attacker to control the entire training dataset of the victim model. In contrast, the threat model is more general, only requiring the attacker to poison a portion of samples into the training data of the victim model. To the best of our knowledge, GRASP is the first to explore how to evade backdoor detection under this threat model.

III. Observation and Insight

A. Main observation

Trigger inversion aims to produce a pattern as close to the injected one as possible. The tolerance of the injected trigger's precision, measured by the trigger effective radius, denotes how close the putative trigger needs to be to the injected one in order to induce target misbehavior on a subject model. Our observations indicate a significant correlation between the effective radius of the trigger and the efficacy of backdoor detection. Typically, in various types of backdoor attacks, those with smaller trigger effective radius are more likely to evade backdoor detection, particularly for those detection methods based on trigger inversion.

To investigate the relationship between the trigger effective radius and the efficacy of backdoor detection, we evaluated the trigger effective radius (by utilizing L-BFGS algorithm [42], [26]in the trigger domain to determine the smallest perturbation that crosses the decision boundary) in ten typical backdoor attacks (BadNet (BN) [10], low-c (LC) [2], Adap (Ad) [36], blend (AB) [35], sig [29], LIRA [7], WaNet (WN) [32], Composite (Co) [25], SIM [37], smooth (LSBA) [34]) under CIFAR-10. More specifically, we utilize the entire dataset (training and testing data) for the trigger-effective radius evaluation and used VGG-16 and ResNet-18 as the model architectures.

Our findings show that, generally, backdoor attacks with a higher trigger effective radius are more easily detected by trigger inversion, while those with a lower trigger effective radius are less likely to be detected. Figure 1 shows the relationship between the trigger effective radius (x-axis) and the effectiveness of the ten attacks to evade the trigger inversion (y-axis) based on our experimental results. Here, the effectiveness of each attack is measured by the detection accuracy (AUC) of NC[45]. The experiment was conducted on CIFAR-10, where we trained ten legitimate and ten backdoored models for each attack. In the section IV-B, we will give a theoretical explanation of why trigger inversion works well when this robust ratio is large.

However, there is no evidence suggesting that a high effective radius is *essential* for the success of a backdoor attack. On the contrary, our research shows that it is entirely feasible to reduce the trigger effective radius of trigger-inserted inputs to defeat trigger inversion without compromising the backdoor effect. To achieve this, we developed a new backdoor attack called GRASP that enhances backdoor stealthiness through training data poisoning when the defender attempts to detect the attack using gradient-based trigger inversion.

We further demonstrate that this straightforward approach is not only theoretically sound (Section IV-B) but also effective when applied to enhance existing backdoor attacks, which are designed to evade other backdoor defenses. This is because GRASP is a generic trigger injection method that can be implemented through data poisoning and can thus be combined with any other stealthy backdoor attacks. Finally, our experiments show that the GRASP-enhanced backdoor attacks are effective in defeating all known gradient-based trigger inversion solutions (Section V-B), indicating that our current progress in backdoor detection could actually be rather fragile.

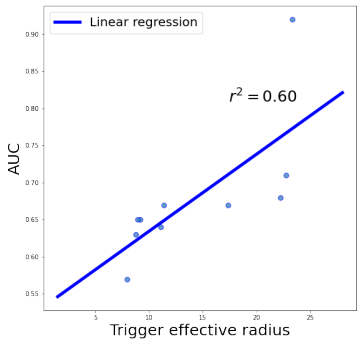
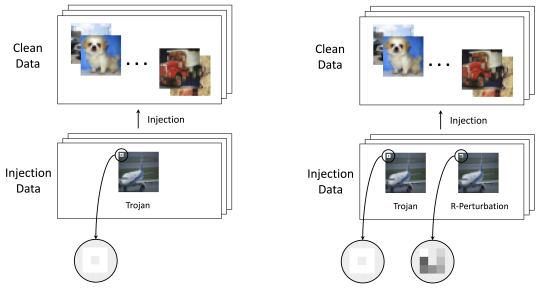


Figure 1: The scatter plot shows the relationship between the trigger effective radius and the effectiveness of ten attacks to evade the NC backdoor detection (measured by AUC). The X-axis represents the trigger effective radius, and the y-axis represents the AUC score when using NC[45] to detect the backdoored models under these attacks. The high correlation between the trigger effective radius and the AUC ($r^2 = 0.60$) indicates the backdoored models with high trigger effective radius are easier to be detected by the trigger inversion technique than those with low effective radius.



(a) BadNet attack (b) GRASP-enhanced BadNet attack

Figure 2: Comparison of the data poisoning backdoor attack by BadNet with (a) or without (b) GRASP enhancement. The GRASP enhancement contaminates trigger-inserted samples (labeled as the target class) along with the noise-added, trigger-inserted samples (labeled as the source class) into the training set, whereas the BadNet attack only contaminates the trigger-inserted samples.

In the next section, we will discuss why the trigger effective radius can affect the effectiveness of trigger inversion.

B. When Trigger Inversion Fails

Based on the observation illustrated in Fig 1, we hypothesize that the trigger effective radius of backdoored model is positively correlated with the effectiveness of gradient-based trigger inversion methods. In this section, we aim to offer an intuitive explanation for this hypothesis. In section IV-B, we will provide a formal theoretical analysis.

Ideally, the infected model should always have 100% confidence in predicting trigger-inserted inputs as the target

label, as observed from the performance of most state-of-the-art (SOTA) backdoor attacks [33][38]. The perfect trigger in such an ideal attack would cause the infected model to have an infinite change rate (trigger effective radius equals zero) around trigger-inserted inputs. However, such a trigger cannot be reconstructed by trigger inversion because all inversion algorithms rely on the gradient to search for the trigger as the local optimum of the loss function. In practice, such a perfect trigger does not exist in a neural network because the neural network is a continuous function. Therefore, we relax the definition of the perfect trigger: instead of an infinite change rate, we consider a very large change rate. Equivalently, we allow the trigger to tolerate only a small amount of noise so that the neural network remains continuous but has a sharp slope around the trigger-inserted data point. Intuitively, if we decrease the trigger effective radius, we will make it more difficult to optimize Eq.1, due to the following constraints:

• It requires the gradient-based optimization to initiate from more random points to find an optimum near the trigger-inserted data point;

• When the optimization process comes close to the trigger-inserted point, it needs to use a small updating step to ensure that the gradient-based search does not jump over the optimum.

In Sections IV-A, we will describe the method to implement the backdoor attack based on this intuition. Our method, GRASP, follows a general data poisoning threat model as assumed by BadNet [10], in which the adversary does not need to access (or even control) the training process but only needs to contaminate a small fraction of poisoning data (containing the trigger) into the training dataset. Both the theoretical analysis (Section IV-B) and the evaluation results (Section V) show that our method can introduce backdoors that are more likely to evade state-of-the-art backdoor defense methods using trigger inversion algorithms.

IV. Gradient Shaping (GRASP)

A. Method

Consider a typical data augmentation training, which adds a new augmented data point (x_{new}, y) w.r.t the original training data point (x, y), where $x_{new} = x + c \cdot \epsilon$ with ϵ being a white noise (normally or uniformly distributed), and keeps the label of (x, y). While this augmentation training enhances the robustness of the entire input, intuitively, it also can be leveraged to improve trigger effective radius by adding noise to the trigger while retaining the intended target label. However, our objective is to reduce the trigger effective radius on its attached inputs. For this purpose, we develop a *gradient shaping* technique. The primary goal of gradient shaping is to make the backdoor trigger more sensitive to perturbations, thus increasing its stealthiness and making it harder to detect using gradient-based trigger inversion methods. By adding a controlled amount of noise specifically to the trigger region in the poisoned training data and adjusting the corresponding labels, we aim to create a steeper decision boundary around the trigger-inserted inputs, which would hinder the effectiveness of trigger inversion techniques.

For a given poisoning data point (x, y) where y is the target class, we add white noise ϵ only on the trigger: $x_{enhance} = x + c \cdot \epsilon \odot \boldsymbol{M}$, where c is a hyper-parameter to control the magnitude of the added noise, $\odot$ denotes element-wise multiplication, and $\boldsymbol{M}$ is a mask matrix of the same dimensions as x, with elements set to 1 only at the trigger positions and 0 elsewhere. By doing this, we only add noise at the trigger positions. Unlike augmentation training, we label $x_{enhance}$ as the source class instead of the target class assigned to noise-free poisoning data. An example of how GRASP works is presented in Fig.2b. Ideally, GRASP aims to generate a corresponding $x_{enhance}$ for each poisoning data point x and include $x_{enhance}$ in the poisoning dataset as well. However, doing so would increase the poisoning injection rate. To address this, we only apply this augmentation to a subset of poisoning data points instead of all of them. We define the ratio of poisoning data points subjected to augmentation as the enhancement rate β. Note that the noise added to the trigger on each trigger-inserted sample is independently and identically distributed (iid), hence unique. This approach facilitates the sharpening of the trigger's gradient in various directions.

Typically, an enhancement rate of 10% is sufficient to improve the performance of GRASP. Larger enhancement rates do not necessarily lead to further improvement in the performance of GRASP. The influence of the enhancement rate on the performance of GRASP will be discussed in Section IV-C.

Note that in Section IV-B, we discuss how to select an appropriate value for c, which represents the theoretical upper bound of c if we aim to decrease the trigger effective radius. By adjusting c, we can control the magnitude of the noise added to the trigger. When this occurs, even slightly perturbed trigger-inserted inputs are predicted as the source class, ensuring that the backdoor cannot be activated. Consequently, in Section IV-D, we demonstrate that by selecting different appropriate values for c ranging from low to high, the trigger effective radius is increased, making the backdoor attack easier to detect.

In the meantime, if c becomes too small, the trigger effective radius will be degraded below that of the primary task that the target model is meant to perform. This subjects GRASP to the backdoor mitigation, such as RAB [46], which adds noise to training data to nullify the effect of the trigger (Section VI Certified Backdoor Defense). Hence, we need to choose the appropriate value of c (see section IV-D) for the best performance of GRASP.

GRASP is designed as a data poisoning method and can work on any trigger. In practice, we may enhance existing backdoor attacks by first generating the trigger using these attack methods and then injecting the trigger into the training dataset using GRASP. Algorithm 1 provides the pseudo-code of this data-poisoning approach for a generic trigger generated by the backdoor attacks, for example, in [10] and [2].

For sample-specific triggers generated by the attacks, particularly those associated with feature-space triggers as discussed in [32], [52], each trigger-inserted sample i is associated with a distinct mask $\boldsymbol{M}_i$. Consequently, the GRASP-enhanced sample $x_{i,enhance}$, is computed as $x_{i,enhance} = x_i + c \cdot \epsilon \odot \boldsymbol{M}_i$. This ensures that the manipulation introduced to each sample is uniquely tailored, enhancing the specificity and potential effectiveness of the backdoor attack.

The algorithm 1 works with three parameters: the poisoning rate α, i.e., the proportion of trigger-inserted samples to be poisoned into the training dataset, the enhancement rate β, i.e., the ratio of the number of enhancement data to the number of poisoning data, and noise scale c, i.e., the magnitude of perturbation on the trigger. In our experiments, we typically set $\alpha = 6\%$, $\beta = 10\%$, and $c = 0.1$.

We evaluate the trigger effective radius of existing backdoor attacks before and after enhanced by GRASP. As shown in Fig 3, the GRASP enhancement can indeed reduce the trigger effective radius and thus enhanced the backdoor attack become harder be detected.

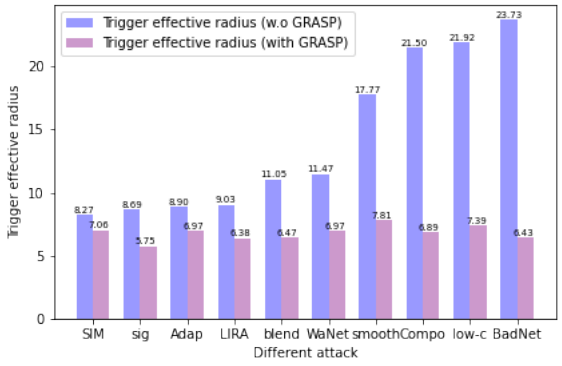


Figure 3: Blue bars show the trigger effective radius on different backdoor attacks. The Red bars show the radius of different backdoor attacks that are enhanced by GRASP with $c = 0.1$.

Algorithm 1 GRASP data poisoning for fixed trigger

Input: Trigger magnitude matrix $\boldsymbol{\Delta} \in \mathbb{R}^{m_1 \times m_2}$, trigger mask matrix $\boldsymbol{M} \in \mathbb{R}^{m_1 \times m_2}$, noise scale $c \in \mathbb{R}$, training data inputs $X \in \mathbb{R}^{m_1 \times m_2 \times n}$, training data label $Y \in \{1, ..., k\}^n$, target label $y_t \in \{1, ..., k\}$, poisoning rate α, enhancement rate β
Output: $(\tilde{X}, \tilde{Y})$

$\tilde{X}, \tilde{Y} \leftarrow$ initialized with empty sets
Randomly shuffle training data X, Y
for $i \in \{0, ..., n-1\}$ **do**
  if $i < \alpha \cdot n$ **then**
    $\tilde{X}$. add$(A(X_i, \boldsymbol{M}, \boldsymbol{\Delta}))$
    $\tilde{Y}$. add(y_t)
  else if $i < \alpha \cdot n + \alpha \cdot \beta \cdot n$ **then**
    $\boldsymbol{\Delta}_{\text{noisy}} \leftarrow \boldsymbol{\Delta}$
    for $p \in \{0, ..., m_1\}$ and $q \in \{0, ..., m_2\}$ **do**
      if $\boldsymbol{M}_{pq} \neq 0$ **then**
        $\boldsymbol{\Delta}_{\text{noisy},pq} \leftarrow$ add a value from $\mathcal{N}(0, 1)$
      end if
    end for
    $\tilde{X}$. add$(A(X_i, \boldsymbol{M}, \boldsymbol{\Delta}_{\text{noisy}}))$
    $\tilde{Y}$. add(Y_i)
  end if
end for

B. Theoretical Analysis of GRASP

In this subsection, we present the theoretical analysis of GRASP, aiming to answer two main questions: 1) Why are gradient-based trigger inversion methods effective on triggers with high effective radius? and 2) Why can GRASP render trigger inversion ineffective, even though these techniques perform exceedingly well on existing backdoor attacks?

More specifically, in Section IV-B1, we attempt to bridge the relationship between trigger effective radius and the efficiency of gradient-based trigger inversion methods. Due to the challenging nature of theoretical analyses for the optimization of a generic target function (approximated by a deep neural network), our analysis focuses on optimization functions that are high dimensional non-convex but satisfy the PL condition [13], as well as one-dimensional piece-wise linear (non-convex) functions, which represent neural networks using activation functions such as ReLU.

Next, in Section IV-B2, we address the second question. By proving Theorem 2, we demonstrate that when using GRASP to inject a trigger, the backdoored model exhibits a higher local Lipschitz constant around the trigger-inserted points, effectively reducing the trigger's effective radius and rendering trigger inversion less effective.

1) Why Inversion Works on Large Effective Radius: To gain insights into the effectiveness of gradient-based trigger inversion methods on triggers with large effective radius, we analyze the optimization behavior of two types of functions under different constraints. These functions serve as simplified approximations of the target functions in deep neural networks.

First, we study the gradient-based optimization of a non-convex function, starting with a one-dimensional function. In this scenario, we consider the target function as a piece-wise linear function, representing a neural network that employs activation functions such as ReLU. Theorem 1 establishes the positive relationship between the effective radius of a trigger (as the global optimum of the target function) and the probability that the gradient-based optimizer converges to the trigger.

Theorem 1. *Given a piece-wise linear function $\ell(\cdot) : [a, b] \to [0, 1]$ with global optimum on a convex hull (there exist a c in this convex hull, such that $\ell(c) > \ell(x)$ for any $x \in [a, b]$), after n iterations, a gradient-based optimizer starting from a random initialization converges to the optimum with the probability:*

$$1 - B_1^{-1}(b-a)^{-1}(4 - B_1B_2)^n(1 - B_1B_2)] \tag{3}$$

where $B_1 > 0$ is a component indicating the area under the desired convex hull and $B_2 > 0$ is a component indicating the likelihood of the linear pieces outside the convex hull jumping into the convex hull during a gradient-based iteration. We present an example in Appendix Fig 34 to provide a visual illustration of the intuition behind the desired convex hull and the area outside of the desired convex hull. (For details, see the proof of Theorem 1 in Appendix A.)

Notably, in the gradient-based optimization for trigger inversion, the optima represent the desirable trigger-inserted points. A larger convex hull is positively correlated with a larger effective radius of the trigger, increasing the probability that the gradient-based trigger inversion identifies the trigger.

Additionally, in the Appendix C, we prove in Theorem 3 that, when the target function is high dimensional non-convex but satisfies the PL condition [13], the gradient-based optimization algorithms converge faster to the desirable optimum

(i.e., the trigger-inserted point) if the local Lipschitz constant near the optimum is lower. As shown in recent research [44], [16], the neural network with high robustness tends to have a lower local Lipschitz constant. Our analysis further illustrates the high correlation between the trigger effective radius and the effectiveness of gradient-based trigger inversion methods.

2) Why Inversion Fails under GRASP: In this section, we analyze the local Lipschitz constant around trigger-inserted samples, specifically how it is influenced by the noise level (measured by the parameter c) in the GRASP algorithm. A greater local Lipschitz constant implies steeper output around trigger-inserted points, leading to a smaller trigger effective radius, which in turn makes trigger inversion less effective. Theorem 2 demonstrates that when each of the two typical noise distributions is used, the GRASP-poisoned model will have a greater local Lipschitz constant around x than the model under the data poisoning attack without using GRASP, such as BadNet[10].

Formally, consider a single case in GRASP data poisoning: a trigger $(\boldsymbol{M}, \boldsymbol{\Delta})$ is injected into a single normal data point (x, y), resulting in the trigger-inserted data point (x', y_t), where $x' = A(x, \boldsymbol{M}, \boldsymbol{\Delta})$. Let (x^*, y) be the trigger-inserted data point with noise ϵ added on the trigger part, which is sampled from the chosen noise distribution. Hence, $x^* = A(x, \boldsymbol{M}, \boldsymbol{\Delta}) + c \cdot \epsilon \cdot \boldsymbol{M}$. Let f be the GRASP-poisoned classification model, which we assume is astute at (x, y), (x^*, y) and (x', y_t).

Theorem 2. *If the noise $\epsilon \sim \mathcal{N}(0, 1)$ (i.e., the white noise), and $c < \|x' - x\|_2 \cdot \frac{\Gamma\left(\frac{|m^*|}{2}\right)}{\sqrt{2}\Gamma\left(\frac{|m^*|+1}{2}\right)}$, where $|m^*|$ is the l_1 norm (i.e., the size) of the trigger, Γ is the Euler's gamma function. A model attacked by a backdoor attack and enhanced by GRASP using the training data points $(x, y), (x', y_t)$ and (x^*, y) has a greater local Lipschitz constant around x than the model backdoored by the same attack without the enhancement by GRASP using the training data points $(x, y), (x', y_t)$.*

Similarly, if $\epsilon \sim uniform(-1, 1)$, and $c < \|x' - x\|_2$, the GRASP-enhanced model has greater local Lipschitz constant around x than the model without the enhancement.

The proof of Theorem 2 is provided in Section B of the Appendix. This theorem establishes a theoretical foundation for understanding how the magnitude of noise impacts the trigger effective radius when using the GRASP-enhanced backdoor. Specifically, it demonstrates that if the level of noise utilized in GRASP is bounded by the l_2-distance between a normal data point x and a trigger-inserted point x', the model poisoned by GRASP will exhibit a greater local Lipschitz constant in the vicinity of x'. Intuitively, this results in a steeper output around x' compared to the model compromised by traditional backdoor attacks, such as BadNet. However, deriving the direct relationship between the magnitude of noise and the trigger effective radius is challenging. Instead, we present a relationship between the magnitude of noise and the local Lipschitz constant around the trigger area, as the local Lipschitz constant serves as an upper bound reflecting the trigger's effective influence. Integrating the insights from Theorem 2 with those from Theorems 1 in Section IV-B1, we infer that GRASP has the potential to reduce the effectiveness of trigger inversion techniques, thereby enhancing the stealthiness of the backdoor attack.

C. Impact of Enhancement Rate in GRASP

In GRASP, the enhancement rate β is used to control the proportion of augmentation data within the poisoning process: $\beta = \frac{N_e}{N_p}$ where N_e denotes the number of enhanced data points, and N_p denotes the number of poisoned data points.

In general deep learning augmentation training, the overall robustness of the model can be improved by introducing more augmented data into the training dataset through techniques such as flipping, translating, masking, and adding noise to all training data. However, in GRASP, our goal is to reduce the trigger effective radius. Since the trigger's characteristics are much simpler compared to the entire model, reducing the trigger effective radius is much easier than enhancing the model robustness. Therefore, augmenting only a portion of the poisoning data can effectively reduce the trigger effective radius.

To demonstrate this point, we conduct an experiment on CIFAR-10 to enhance the BadNet attack with GRASP using different β values, while keeping $c = 0.1$. The results are shown in Fig. 4, where the x-axis denotes the enhancement rate β, and the y-axis on the left presents the fold-change of the average trigger effective radius of trigger-inserted data points, indicating the effective radius of the trigger. The blue line in the figure shows the relationship between β and the effective radius. We observe that an increase in the enhancement rate β will only lead to the decrease in the trigger effective radius when $\beta < 10\%$. When $\beta > 10\%$, further increasing β values do not affect the trigger effective radius. These results suggest adding 10% enhanced data points into the poisoning data will reach the optimal enhancement effect on the BadNet attack. We observe similar results on the other backdoor attacks as well. We also present the test accuracy (ACC; the red line) and the attack success rate (ASR; the yellow line) of backdoored models, as well as the detection performance of NC (AUC; the green line) Fig. 4.

In the Appendix, we further elaborate on the impact of β under conditions involving more complex trigger types. Broadly, for more intricate trigger types like those cited in [29], [25], a β value reaching 10% is sufficient to substantially reduce the trigger effective radius to a low enough point, thereby allowing evasion of backdoor detection.

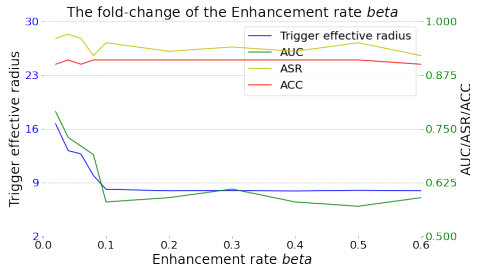


Figure 4: The fold-change of the average trigger effective radius of the trigger-inserted data points in the GRASP attacked models compared with the BadNet attacked models.

D. Impact of Noise Level in GRASP

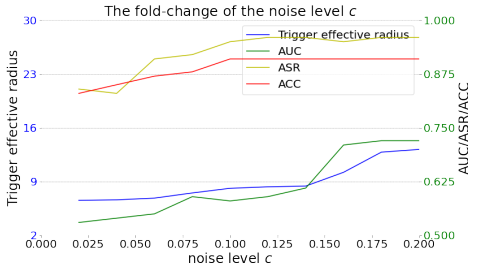


Figure 5: The fold-change of the average trigger effective radius of the trigger-inserted data points in the GRASP attacked models compared with the BadNet attacked models.

In this section, we investigate the impact of the noise level c on the trigger and the backdoored model. We systematically alter the magnitude of c to generate the backdoors with different trigger effective radius. Subsequently, we evaluate their susceptibility or resilience to evasion detection, aiming to investigate the relation between the magnitude of the trigger's effective radius and the ability to evade detection of backdoor attacks. More specifically, as explained in Section IV-A, the noise added to a trigger reduces its effective radius, and the magnitude of the additive noise introduced by GRASP is controlled through the parameter c. The smaller the magnitude of the noise (controlled by c), the smaller effective radius of the trigger is. Following Section IV-C, we conduct an experiment on CIFAR-10 to enhance the BadNet attack by using GRASP with different noise level c, but the same poison rate of 10%. The results are shown in Fig. 5, where the x-axis denotes the noise level c, and the y-axis on the left denotes the fold-change of the average trigger effective radius in trigger-inserted samples, which indicates the effective radius of the trigger. The blue line in the figure shows the relationship between c and the effective radius. We observe that with the increase of the noise level c, the trigger effective radius increases. We also present the test accuracy (ACC; the red line) and the attack success rate (ASR; the yellow line) of backdoored models, as well as the detection performance of NC (AUC; the green line) in the figure.

When the noise level is low (< 0.075), ACC and ASR are slightly affected. As discussed earlier, a very small c makes the trigger effective radius degrade below that of the primary task of the target model (Details in Section VII), subjecting GRASP to backdoor mitigation techniques such as RAB [46] that nullify the effect of the trigger. The detection effectiveness of NC improves with the increase of c. This echoes our observation in Section III that the detection performance positively correlates with the trigger effective radius. However, existing inversion-based detection methods, such as NC [45] used in this example, are less effective against GRASP as the AUC remains low in Fig. 5 under different noise levels. In the following section, we will further evaluate the capability of GRASP in evading backdoor detection methods.

E. Impact on Learning Optimizer

In this chapter, we investigate the effectiveness of trigger inversion employing three distinct optimizers: SGD (a first-order method), Adam [15] (a first-order method augmented with momentum), and AdamHessian [50] (a second-order method also incorporating momentum). Specifically, we adhere to the evaluation framework delineated in Section V, applying the Neural Cleanse (NC) method to reverse potential backdoor in models potentially compromised by the GRASP-enhanced BadNet on the CIFAR-10 dataset. For the momentum-inclusive optimizers, we fix the momentum parameter at 0.9, exploring a range of learning rates (step sizes) including $0.0001, 0.0005, 0.001, 0.0015$, and 0.002. Note that, under conventional experimental settings, the learning rate is typically selected within the range of 0.001 to 0.005. Here, we deliberately choose considerably lower learning rates, aiming to underscore the remarkable stealthiness of GRASP-enhanced BadNet, even when subjected to trigger inversion with small learning rates. Figure 6 illustrates the variation of the Detection AUC in response to changing step sizes, revealing that even at an exceptionally small step size (learning rate = 0.0001), the GRASP-enhanced BadNet sustains a relatively low detection AUC (around 55%), thereby demonstrating its resilience to various settings of optimization methods.

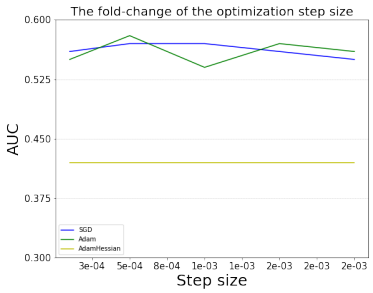


Figure 6: Impact of optimizers and learning rate

We also investigate whether the decay of the learning rate would impact the performance of GRASP. Specifically, we follow the setup in Section IV-C to evaluate the GRASP-enhanced BadNet attack on CIFAR-10 with different decay rates of the learning rate (ranging from 0.001 to 0.01) while using NC for detection (with lr = 1e-3). It's worth noting that in this case, we set the epoch of NC as 100, indicating that in the most extreme case (decay rate = 0.01), the learning rate would decrease by half (to lr = 5e-4). Fig 35 in the Appendix illustrates the AUC, ASR, and the trigger effective radius under various decay rates of the learning rate. Overall, the decay rate of the learning rate does not significantly impact the performance of GRASP.

F. Impact on Environmental Factors

In this subsection, we analyze the potential impact of GRASP on the trigger effectiveness under various background factors. Specifically, we investigate whether the trigger effectiveness, measured by the ASR, is compromised under natural disturbances (corruptions) affecting the entire image, such as the brightness, the contrast, and the color.

To address this issue, we have devised experiments to evaluate the Impact of Environmental Factors on the robustness of the backdoor mechanism. To ensure a fair and comprehensive assessment, we employ the *imagecorruptions* [31] benchmark to test the backdoor's effectiveness against corrupted transformations. The imagecorruptions benchmark encompasses 15 types of algorithmically generated corruptions, spanning categories of noise, blur, weather, and digital alterations. Each corruption type is graded on a severity scale from 1 to 5, where a severity of 1 represents relatively mild corruption, and a severity of 5 indicates more severe corruption. In here, we present the results for the three most common types of corruptions: Brightness, Contrast, and JPEG Compression, which are pivotal in understanding the robustness of the backdoor under varying conditions.

In this study, we employed the ResNet-101 neural network architecture to train two distinct models on the CIFAR-10 dataset: one model contaminated with a backdoor through the BadNet methodology, and another utilizing a backdoor implemented via the GRASP-enhanced BadNet approach. As depicted in Table I, we present the Attack Success Rate (ASR) of these corrupted testing data, subjected to three different types of corruption and five varying degrees of severity on trigger-inserted testing data.

Upon examination, it is discernible that under varying levels of brightness and JPEG compression corruption, the performance of the BadNet model and the GRASP-enhanced BadNet model are relatively similar, with a general disparity of around 2%. However, under contrast corruption, a marked improvement is observed in the performance of the GRASP-enhanced BadNet model compared to the BadNet model, with differences ranging from 4% to 16%.

A comprehensive comparison encompassing all 15 types of corruption from the imagecorruptions benchmark is documented in the Appendix. It is noteworthy that in the majority of corruption scenarios, the performance of the two models is comparable.

Table I: Backdoor effectiveness Comparison: ASR of the BadNet model and the GRASP-enhanced BadNet model across various corruption types and severity levels

Corruption	Severity				
	1	2	3	4	5
			BadNet		
Brightness	91.2%	88.7%	85.6%	81.0%	74.6%
Contrast	70.0%	55.2%	16.8%	8.9%	6.9%
JPEG comp.	71.4%	61.2%	59.2%	44.3%	39.2%
			GRASP		
Brightness	90.0%	88.2%	84.9%	78.7%	69.2%
Contrast	76.6%	59.4%	33.2%	16.9%	13.6%
JPEG comp.	69.4%	64.1%	60.1%	50.9%	42.5%

V. Against Backdoor Detection

In this section, we evaluate the effectiveness of GRASP against two types of backdoor detection methods: inversion-based and weight analysis-based backdoor defenses. We compare the performance of these detection methods against different attacks before and after being enhanced by GRASP.

A. Datasets and Settings

Datasets. We analyzed backdoor attacks on the models trained using three public datasets: MNIST [19], CIFAR10 [17], and Tiny ImageNet [18], as shown in Table II. Our experiments were conducted on a server with one AMD Ryzen 3980X 3.2 GHz 48-core processor and one NVIDIA RTX 3090 GPU.

Dataset →	MNIST	Tiny ImageNet	CIFAR10
Training samples (#)	60,000	160,000	50,000
Testing samples (#)	10,000	40,000	10,000

Table II: Datasets statistics

Backdoor Attacks. We considered seven existing backdoor attacks: BadNet [10], LSBA [34], Composite [25], clean label [38], DEFEAT [52], IMC [33], and adaptive-blend [35]. These backdoor attacks fall into four general categories: patch trigger, clean label, imperceptible, and latent space inseparable. For each attack, we generated and evaluated 24 backdoored models: for each of the three different datasets (MNIST, CIFAR-10, and Tiny ImageNet), we generated two models using each of four different neural network architectures (VGG-16, ResNet-101, ShuffleNet, and ResNet18, respectively).

• *Patch Trigger*. Patch triggers typically utilize a small pattern as the trigger for the backdoor attack. We selected BadNet [10], LSBA [34], and Composite [25] in this category and used two patterns (as shown in Fig. 32 in the Appendix) as the patch triggers in the backdoor attack.

• *Clean Label*. Clean-label backdoor attacks contaminate the training dataset with clean-label data. We selected Latent [38] to represent the attacks in this category.

• *Imperceptible*. An imperceptible backdoor attack aims to design a backdoor trigger that can evade human inspection. Most of these attacks enhance backdoor stealthiness through universal adversarial perturbation (UAP). We selected DEFEAT [52] and IMC [33] in this category for our experiments.

• *Latent Space Inseparable*. A latent space inseparable backdoor attack aims to design a backdoor trigger so that in the target model's latent space, the trigger-inserted samples are close to the clean samples in the target class. We selected Adaptive-Blend [35] in this category for our experiment.

• *Attack Parameters*.We inserted 3,600, 3,000, and 9,600 poisoning data samples (i.e., a poisoning rate of $\alpha = 6\%$) into the CIFAR-10, MNIST, and Tiny ImageNet training datasets, respectively. These samples were uniformly selected from each class, establishing source-agnostic backdoors. Following the methods in the original papers, the trigger in IMC was synthesized as described in [33], and the trigger in Latent was randomly initialized as detailed in [38].

• *GRASP*. For each attack mentioned above, we combine them with GRASP by algorithm 1. More specifically, we set $\alpha = 0.6$ and $\beta = 10\%$ (Algorithm 1).

Trigger inversion. We implemented and tested four backdoor countermeasures based upon trigger inversion: Neural Cleanse [45], TABOR [11], K-arm [39], and Pixel [43]. In our experiments, we utilized 10% of the training data and the default hyper-parameters provided in the original papers for trigger reconstruction.

B. Putative Trigger Effectiveness

Existing methods measure the effectiveness of trigger inversion by computing the similarity between the reconstructed and real triggers, e.g., based on l_1 distance, which is insufficient since a similar pattern may not have a similar backdoor effect (i.e., ASR). We propose a set of metrics to measure trigger accuracy. Below we present our experimental results on the effectiveness of backdoor detection by four trigger inversion algorithms: (NC [45], TABOR [11], Pixel [43], and K-arm [39]), by comparing the effectiveness of the backdoor attacks before and after the enhancement by GRASP. More specifically, after the trigger is generated by each backdoor attack method, we use GRASP to enhance this trigger as described in section IV-A. Here, we append a symbol "*" to the name of each backdoor attack to indicate the respective attack enhanced by GRASP. For example, "BadNet*" indicates BadNet enhanced by GRASP.

Metrics. In our experimental study, we utilize four quantitative metrics to measure the effectiveness of a backdoor in evading a gradient-based inversion algorithm (for reconstructing a trigger $(\boldsymbol{\Delta}, \boldsymbol{M})$ in a model f):

• ϵ_1: The difference between the real trigger's ASR on the backdoored model and that on the "sanitized" model retrained to unlearn the reconstructed trigger: that is, $\epsilon_1 = |ASR_{unlearn} - ASR|$. A smaller difference indicates that the reconstructed trigger is less accurate, thus, unlearning is less effective.

• ϵ_2: The Jaccard distance between the trigger mask of the reconstructed trigger M' and of the real trigger M can be calculated as $J(M', M) = \frac{|M' \cap M|}{|M'|+|M|-|M' \cap M|}$.

• ϵ_3: The ASR of the reconstructed trigger $(\boldsymbol{M'}, \boldsymbol{\Delta'})$ on a clean model f^*: $\epsilon_4 = ASR'_{f^*}$. A large ASR'_{f^*} indicates that the reconstructed trigger is likely a natural trigger [29], not the real one meant to be recovered.

• ϵ_4: AUC score of backdoor detection. The trigger inversion methods often use the l_0 norm of the reconstructed trigger as the measurement to distinguish backdoored models from benign models: the lower of the l_0 norm, the more probable the model has been backdoored. An AUC score of 50% in backdoor model detection suggests that the backdoored model is indistinguishable from benign models.

Notably, for a trigger inversion algorithm with ideal performance, ϵ_3 is anticipated to be close to 0, while ϵ_1,ϵ_2 and ϵ_4 are anticipated to be close to 1.

Experimental results. Here we present our results as measured by the aforementioned metrics. Due to the space limit, we defer our complete experimental results to Table XI in Appendix and only report representative results (ϵ_4) in this section.

• ϵ_1*: effectiveness of unlearning*. The reconstructed trigger can be used for backdoor unlearning [45], [11], [39]. After we reconstructed the trigger for a given backdoored model during the unlearning procedure, we first built an unlearning dataset, including randomly selected 10% of the training data (6,000 in MNIST, 5,000 in CIFAR-10, and 16,000 in Tiny ImageNet). Then, we added the reconstructed trigger onto 10% of the unlearning dataset (600 in MNIST, 500 in CIFAR-10, and 1,600 in Tiny ImageNet) while keeping their class labels intact (the original source class). After that, we fine-tuned the model on this unlearning dataset. We used SGD as the optimizer in the experiment and set the learning rate = 0.01 and momentum = 0.9. As shown in Table XI, after unlearning with the reconstructed trigger by various trigger inversion algorithms, most models poisoned by the attack enhanced by GRASP still preserve much higher ASRs (almost identical to those before unlearning), so that the GRASP-enhanced attacks achieve lower ϵ_1 than respective backdoor attack. TableXI shows that on CIFAR-10, BadNet achieves the worse performance against the trigger inversion defense of Tabor ($\epsilon_1 = 97.5\%$), which is significantly enhanced by GRASP ($\epsilon_1 = 1.5\%$). Among other attacks, LSBA* has the best performance under pixel as 0.6%.

• ϵ_2*: distance between trigger masks*. We observed that the reconstructed triggers from the models poisoned by GRASP-enhanced attacks have very low similarity with the real triggers (i.e., the overlap between the real and the reconstructed triggers are less than 20%). By comparison, the reconstructed triggers from the models under the backdoor attacks without GRASP enhancement overlap with the real triggers by about 10% - 60%. On CIFAR-10, when enhanced by GRASP, DEFEAT* has worse performance against pixel ($\epsilon_2 = 0.13$). While BadNet* has the best performance against NC ($\epsilon_2 = 0.00$) (Table XI).

• ϵ_3*: ASR of the reconstructed triggers on a clean model.*

We also computed ϵ_3, the ASR of the reconstructed triggers from the poisoned models on a clean model for the same task. In our experiment, we used CIFAR-10, MNIST, and Tiny ImageNet as the clean datasets to train the clean models. After a trigger is reconstructed from a poisoned model, we randomly select 500 images (200 from Tiny ImageNet) from the source class of the clean dataset and insert the trigger on them. The ASR was then measured on this set of trigger-inserted samples on the clean model. As shown in Table XI, the reconstructed triggers from the models poisoned by GRASP-enhanced attacks have a relatively high ASRs on the clean model, almost comparable with their ASRs on the poisoned models, whereas the reconstructed triggers from the models poisoned by the attack without GRASP enhancement have much lower ASRs. This indicates that any useful trigger recovered from the models poisoned by GRASP-enhanced attacks are likely to be a natural trigger introduced by the legitimate learning process that has nothing to do with the injected triggers. On dataset CIFAR-10, when enhanced by GRASP, LSBA* has worse performance against pixel ($\epsilon_3 = 42.1\%$) while Adaptive-Blend has the best performance against ($\epsilon = 28.3\%$).

• ϵ_4 *AUC*. As mentioned earlier, our research shows that trigger inversion algorithms are unlikely to effectively reconstruct and remove the triggers injected by GRASP, even though they are largely successful on the triggers injected by existing backdoor

attacks. In some cases, however, the backdoor defense methods just need to detect the infected models (and discard them afterward), even though they cannot accurately reconstruct the real trigger. Our research evaluated how successfully these trigger inversion methods can detect the models poisoned by GRASP-enhanced attacks. Specifically, we train 24 clean models; for each of the three different datasets (MNIST, CIFAR-10, and Tiny ImageNet), we generate two clean models using each of four different neural network structures (VGG-16, ResNet-101, ShuffleNet, and ResNet18, respectively) to analyze their detection accuracy. Similarly, we train 24 models for each attack on three datasets using four neural network structures. In our research, we measured the AUC score of NC, TABOR, K-arm, and Pixel on these 48 models (24 clean models and 24 backdoored models). As shown in Table III, we present the result of using VGG-16 network structure (Delay rest of the results in Appendix F). Generally speaking, the AUC scores of different defense strategies from the models poisoned by different backdoor attacks with GRASP enhancement are significantly smaller than those from the models poisoned by the same attack without GRASP enhancement, which indicates the better effect of the GRASP-enhanced attacks to evade the detection by all tested triggers inversion algorithms. In particular, LSBA enhanced by GRASP successfully evades the detection by the trigger inversion algorithms with AUCs below 65% for all of them.

Additionally, we delve deeper into the exploration of backdoor removal using putative triggers in Appendix F, where we present a comparative experimental analysis. The results from these experiments demonstrate that backdoor attacks enhanced by GRASP are more resilient to removal attempts using reverse-engineered putative triggers.

C. Against Weight Analysis Detection

Weight analysis aims to distinguish backdoor and benign models by analyzing the signals in model parameters. Specifically, such distinguishable signals are first retrieved, often through training a classifier on the parameters of some sample models, which are subsequently utilized to predict whether any given model is backdoored [9][5][8]. In this section, we evaluated the effectiveness of GRASP against the weight analysis-based backdoor detection methods, which have been shown to perform well in the recent backdoor competitions [14], [1]. In the Appendix, through the theoretical analysis, we show that the backdoored models poisoned by GRASP-enhanced attacks are not further away from the benign models for the same primary task than the backdoored model poisoned by the same attack without GRASP enhancement.

Here we selected Trojan Signature (TS)[9], MNTD[47], Activation Clustering (AC) [5] , Beatrix [30] and ABS [28], the representative methods based on weight analysis. We computed their AUCs on 20 models using VGG16, including ten clean models and ten backdoored models, respectively, trained on each of the three datasets (CIFA-10, MNIST, and Tiny ImageNet). The backdoored models were poisoned by the five backdoor attacks with or without the GRASP enhancement, respectively. Here, the five attacks were selected because they were shown to be effective against the weight analysis-based backdoor defense. Due to the space limit, we only present the most important results in Table IV, the entire results are presented in Table XII. In general, the detection ability (AUC) by the weight analysis methods is lower or comparable on the five attacks when they are enhanced by GRASP, indicating GRASP enhancement does not reduce the effectiveness of these attacks against the weight analysis backdoor defense.

VI. Resilience to Backdoor Mitigation

In this section, we evaluated the resilience of GRASP to backdoor defense methods that do not rely on backdoor detection. In our experiments, we considered not only the five types of backdoor defenses (mitigation or unlearning) as summarized in [23]: Preprocessing-based Defenses, Model Reconstruction, Poison Suppression, and Certified Backdoor Defense, which are not based on backdoor detection techniques such as trigger inversion or weight analysis, but also an emerging category, Training Procedure Defense. We selected a total of seven representative defense methods across these six types: DeepSweep (DS)[51], Fine-pruning (FP)[27], NAD [22], GangSweep (GS)[53], DBD[8], RAB [46], ABL [21], and compared their performance in defending against the chosen backdoor attacks before and after the GRASP enhancement.

We measured the ASRs of backdoors after the backdoor mitigation on the models (VGG16) poisoned by the selected backdoor attacks with or without the GRASP enhancement, respectively. Here, for each defense method, we selected a backdoor attack that has been shown to effectively evade the respective defense in previous studies to demonstrate the enhancement by GRASP does not reduce its effectiveness to evade the respective defense methods.

We summarize the results from both experiments in Table V. Except for special notes, all backdoored models before mitigation achieve an ASR above 95%. Here, the notations are the same as used in Section V: a symbol "*" is appended to the name of the backdoor attack to indicate the respective attack enhanced by GRASP. Below, we discuss the results of different defense methods in detail.

Preprocessing-based Defenses. Preprocessing-based defenses aim to remove the putative poison samples in the training dataset. DeepSweep (DS) [51] is selected as the representative in this category. We tested the average ASR against the models infected by HaS-Net [2], which is a low-confidence backdoor attack. The ASRs of the models poisoned by the GRASP-enhanced HaS-Net attack (HaS-Net*) after the DeepSweep mitigation is comparable with those on the model poisoned by the HaS-Net attack (Table V), which indicates GRASP enhancement did not make the Has-Net backdoored models more easily mitigated by the DeepSweep.

Model Reconstruction. Fine-pruning [27] and NAD [22] are two typical model reconstruction methods to remove the backdoor. We conducted an experiment to compare the performance of Fine-pruning and NAD to defend against DEFEAT [52] with or without GRASP enhancement (DEFEAT or DEFEAT*). Table V shows the ASRs of the injected trigger after fine-pruning and NAD on the respective backdoored models. Note that the model reconstruction methods typically strike a trade-off between the accuracy of the primary task (ACC) and ASR. In our experiment, to keep the fidelity of the mitigation, we control the mitigation procedure such that the ACC does not

	CIFAR-10				MNIST				Tiny ImageNet			
	NC	Tabor	K-arm	Pixel	NC	Tabor	K-arm	Pixel	NC	Tabor	K-arm	Pixel
						ϵ_4: AUC						
BadNet	79.9%	84.0%	85.3%	91.8%	78.6%	81.0%	82.7%	90.3%	75.6%	77.8%	80.4%	84.9%
BadNet*	54.7%	56.1%	60.1%	80.2%	54.0%	55.0%	60.5%	83.9%	55.7%	56.7%	57.5%	78.5%
LSBA	66.5%	68.2%	72.1%	81.0%	67.7%	69.6%	70.7%	78.4%	63.5%	70.0%	70.5%	85.8%
LSBA*	55.1%	55.8%	58.8%	63.7%	53.2%	57.3%	55.8%	62.7%	55.8%	52.0%	56.8%	64.6%
Composite	67.9%	65.9%	70.1%	85.2%	66.4%	65.0%	68.8%	82.5%	65.0%	65.0%	65.9%	81.7%
Composite*	53.5%	58.6%	61.0%	72.9%	52.5%	52.8%	59.5%	71.8%	54.5%	53.7%	58.1%	70.5%
Latent	79.2%	77.1%	78.8%	87.9%	79.9%	78.8%	81.1%	89.5%	73.6%	79.2%	74.9%	83.5%
Latent*	52.5%	54.5%	59.8%	76.0%	54.2%	54.8%	59.0%	74.6%	53.9%	56.0%	56.5%	70.8%
DEFEAT	65.2%	63.2%	77.8%	69.6%	67.0%	69.8%	80.5%	71.1%	63.6%	67.3%	77.0%	67.6%
DEFEAT*	58.8%	59.9%	71.6%	61.4%	58.9%	58.5%	70.9%	59.7%	58.3%	58.9%	72.0%	62.6%
IMC	68.0%	64.2%	76.9%	79.8%	66.6%	68.8%	76.7%	80.2%	67.5%	73.9%	76.3%	78.0%
IMC*	55.9%	55.3%	71.9%	71.1%	54.7%	52.9%	74.0%	73.6%	64.8%	64.7%	71.8%	75.1%
Adaptive-Blend	67.1%	66.5%	68.2%	76.9%	59.9%	62.5%	66.0%	81.5%	62.9%	65.0%	65.5%	76.8%
Adaptive-Blend*	54.2%	56.3%	57.2%	62.8%	55.1%	57.1%	62.0%	73.2%	54.5%	53.5%	54.8%	68.2%

Table III: The AUCs of backdoor detection by trigger inversion methods on the backdoored models poisoned by different backdoor attacks with and without the enhancement of GRASP. The attack with the GRASP enhancement is denoted by the symbol "*" appended to the name of the respective attack.

		CIFAR-10	MNIST	Tiny ImageNet
ABS	DFST	67.4%	65.0%	67.2%
	DFST*	63.1%	62.7%	61.4%
AC	AB	68.4%	69.1%	66.6%
	AB*	57.2%	59.0%	60.1%
TS	DEFEAT	68.9%	67.3%	66.2%
	DEFEAT*	60.5%	68.0%	65.1%
MNTD	DEFEAT	69.2%	73.1%	70.9%
	DEFEAT*	66.0%	72.9%	69.4%
Beatrix	Low-c	58.3%	72.3%	68.1%
	Low-c*	56.9%	72.4%	67.3%

Table IV: The AUCs of weight analysis-based backdoor detection methods on the benign and backdoored models poisoned by DFST, AB, MNTD, Beatrix and DEFEAT with and without GRASP enhancement.

		CIFAR-10	MNIST	Tiny ImageNet
DS	HaS-Net	67.2%	68.9%	65.1%
	HaS-Net*	65.1%	68.3%	66.1%
FP	DEFEAT	81.4%	87.6%	80.3%
	DEFEAT*	83.2%	88.0%	81.9%
NAD	DEFEAT	79.2%	81.1%	80.3%
	DEFEAT*	79.6%	80.3%	80.7%
DBD	IMC	54.2%	59.7%	55.0%
	IMC*	63.5%	64.0%	62.6%
GS	AB	64.9%	59.3%	62.1%
	AB*	65.4%	63.2%	63.5%
RAB	DFST	95.3%	94.1%	91.2%
	DFST*	94.6%	91.2%	91.4%
ABL	AB	91.6%	93.6%	92.5%
	AB*	95.2%	96.7%	93.4%

Table V: The ASRs of the backdoored models after the mitigation by the backdoor mitigation methods, Note that the high ASR indicates that a backdoor attack is more resilient to the respective defense method. Here, all models are trained using three different datasets, and the backdoored models are infected by different attacks with and without the enhancement of GRASP. (The attacks enhanced by GRASP is denoted by "*").

decrease by more than 5%. We use 10% of clean training data for mitigation, and for fine-pruning, we set 10% activation pruning.

Overall, the ASRs of the Fine-prune and NAD on the DEFEAT* attacked models are comparable with the DEFEAT attacked models, indicating the DEFEAT attack with and without GRASP enhancement are similarly effective against the Fine-pruning and NAD mitigation methods.

Non-Gradient Based Trigger Synthesis. Gangsweep [53] is a non-gradient-based trigger synthesis defense that used the reconstructed trigger for backdoor mitigation. We performed an experimental study to confirm Adaptive-blend enhanced by GRASP (Adaptive-Blend*) does not generate the backdoors that can be easily mitigated by Gangsweep (GS). Table V shows the ASRs after the mitigating of Gangsweep on the backdoored models created by using Adaptive-Blend (AB) with or without GRASP enhancement. The ASR on the model attacked by GRASP-enhanced AB (AB*) is comparable (for the Tiny ImageNet dataset) or lower (for the CIFAR and MNIST dataset) than the ASRs on the models attacked by AB, indicating GRASP enhancement does not make the attack more easily to be mitigated by the Gangsweep.

Poison Suppression. For poison suppression defense, most methods (e.g., DBD [12]) learn a backbone of a DNN model

via self-supervised learning based on training samples without their labels to capture those suspicious training data during the training process. We tested the performance of DBD [12] defense against models attacked by IMC[33] with and without GRASP enhancement. As shown in Table V, the ASR of the DBD in the IMC* attacked models are higher than the IMC attacked models, indicating GRASP enhancement does not make the attack more easily to be mitigated by the DBD.

Certified Backdoor Defense. As previously discussed in Section IV-D, if the value of c in GRASP is chosen to be too small, specifically leading the trigger effective radius below the model's robustness, GRASP might be susceptible to nullification by backdoor certification methods. However, the model's robustness is typically much larger than the trigger effective radius, a point we have illustrate and discuss in Section IV-D. Owing to this fact, GRASP can preserve its stealthiness (by significantly reducing c) while circumventing nullification by certification methods (by ensuring that the trigger effective radius stays larger than the model's robustness). RAB [46] is a certified defense method that aims to eliminate the backdoor in the target model. We performed an experimental study to confirm GRASP enhancement does not generate the backdoor that is more easily mitigated by RAB. TableV shows the ASR of the injected trigger after RAB on the same models which attacked by DFST[6] and DFST enhanced by GRASP (DFST*). The ASR on DFST* is comparable to that on the DFST attacked models (MNIST has the most significant difference, which DFST* is 2.9% lower than DFST), indicating combing GRASP backdoored models are not easier to mitigate by the RAB than the model infected by DFST.

Training procedure Defense Apart from the five types of mitigation techniques summarized by [23], we observe the emergence of a new mitigation approach that aims to remove backdoors during the training process. Notably, ABL[21] seeks to identify backdoored inputs during the early training stages. This is because, in a typical backdoored training scenario, trigger-inserted samples (backdoor task) often learn much faster than clean data (legitimate task), as the backdoor task is generally easier than the legitimate task. Based on this observation, ABL mitigates backdoors by eliminating those training samples with small losses during the early stages of the training process. However, we argue that GRASP can potentially evade ABL. This is because training on the GRASP poisoned dataset increases the loss for both poisoned and enhanced data during the initial phase of training. By injecting both poisoning and enhancement data, GRASP makes the backdoor task more challenging to learn. Consequently, during the early training phase, GRASP assists data poisoning in evading mitigation from ABL.

We conducted an experiment to verify that GRASP enhancement does not produce a backdoor that is more easily mitigated by ABL. Table V displays the ASR of the injected trigger following ABL on the same models attacked by AB [35] and those enhanced by GRASP (AB*). The ASR on AB* models is higher than that on the AB-attacked models (with the least significant difference on CIFAR-10, where AB is 1.9% lower than AB*), indicating that combining GRASP with backdoored models does not make them more susceptible to ABL-based mitigation than models infected by AB.

In summary, we find that for the attacks that effectively evade backdoor mitigation, the GRASP enhancement will not make the mitigation less effective. This means that GRASP enhancement can effectively fend off the existing backdoor defenses even though it is designed for evading trigger mitigation. For some mitigation methods like DBD, the IMC attack, the GRASP enhancement in fact increases their effectiveness.

VII. Mitigation and Limitation

GRASP can successfully increase the change rate around the trigger-inserted inputs, effectively reducing the trigger effective radius of these inputs. Note that the trigger effective radius should not be reduced to be lower than the primary task model robustness (primary task effective radius) since otherwise, the backdoor attack may be defended by a straightforward strategy during inference: one can add the noise at the level above the robust radius of the backdoor task but below the robust radius of the primary task into each input; as such, the trigger-inserted inputs will not be predicted as the target class while the prediction of the benign input will not be changed, indicating the backdoor is removed without affecting the primary task. In practice, we found it is easy to reduce the trigger effective radius while keeping it above the primary task effective radius, as we observed that in the BadNet backdoor attack, the trigger robustness is always much greater than the primary task effective radius (as shown in Fig. 7). Therefore, it is always possible for GRASP to generate backdoors more effectively to evade the trigger-inversion algorithms while not affecting the performance of the primary task.

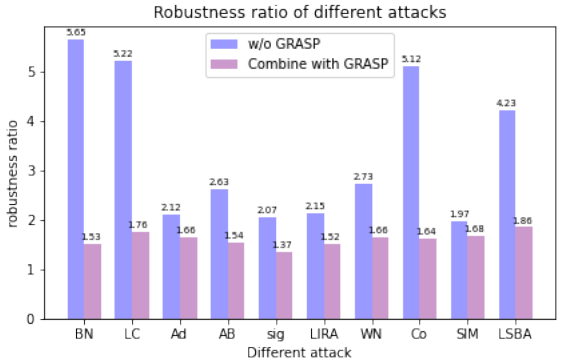


Figure 7: Blue bars show the ratio between the trigger effective radius and the primary task effective radius on different backdoor attacks. The Red bars show the ratio of different backdoor attacks that are enhanced by GRASP.

In Section IV, we assume the model will always give approximately 100% confident prediction (as the target class) on all trigger-inserted inputs. In practice, when this assumption does not hold, for example, in [2], where a low confidence backdoor is injected into the model by manipulating the logits of the poisoning data, the change rate around a perfect trigger may not be very large. Specifically, the trigger-inserted inputs may be predicted as the target class with the lowest confidence in the backdoored model, which turns out to be a perfect trigger without any constraints on the local Lipschitz constant. For such backdoors, GRASP cannot further enhance their stealthiness.

VIII. Related Work

The prior study [33] showed that robust learning on the primary task (i.e., to increase its effective radius) can enhance the stealthiness of backdoor attacks. However, comparing with GRASP, this method is less effective because even when it is difficult to further increase the effective radius of the primary task, the stealthiness is still insufficient for the backdoor to evade current detection methods. More importantly, this method requires the adversary to control the model training procedure, which is often impractical in real-world attack scenarios. In comparison, GRASP can be achieved through data contamination as shown here.

IX. Conclusion

In this paper, we analyzed the efficacy of trigger inversion algorithms in backdoor defense, finding that current backdoor attacks inject noise-robust triggers, facilitating reconstruction via gradient-based algorithms. Consequently, we introduced a gradient shaping (GRASP) strategy to improve backdoor attacks by diminishing trigger robustness using data poisoning, thus evading defenses using trigger inversion. Through theory and experiments, we showed GRASP's enhancement of top stealthy backdoor attacks' effectiveness against trigger inversion, without impairing their resistance to other defenses, including those based on weight analysis.

References

[1] "Tdc 2022," https://trojandetection.ai/, accessed: 2022-09-30.

[2] H. Ali, S. Nepal, S. S. Kanhere, and S. Jha, "Has-nets: A heal and select mechanism to defend dnns against backdoor attacks for data collection scenarios," arXiv preprint arXiv:2012.07474, 2020.

[3] E. Bagdasaryan and V. Shmatikov, "Blind backdoors in deep learning models," in 30th USENIX Security Symposium (USENIX Security 21), 2021, pp. 1505–1521.

[4] C. Cartis, N. I. Gould, and P. L. Toint, "Worst-case evaluation complexity and optimality of second-order methods for nonconvex smooth optimization," in Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018. World Scientific, 2018, pp. 3711–3750.

[5] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," arXiv preprint arXiv:1811.03728, 2018.

[6] S. Cheng, Y. Liu, S. Ma, and X. Zhang, "Deep feature space trojan attack of neural networks by controlled detoxification," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 2, 2021, pp. 1148–1156.

[7] K. Doan, Y. Lao, W. Zhao, and P. Li, "Lira: Learnable, imperceptible and robust backdoor attacks," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 11 966–11 976.

[8] M. Du, R. Jia, and D. Song, "Robust anomaly detection and backdoor attack detection via differential privacy," arXiv preprint arXiv:1911.07116, 2019.

[9] G. Fields, M. Samragh, M. Javaheripi, F. Koushanfar, and T. Javidi, "Trojan signatures in dnn weights," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 12–20.

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

[11] W. Guo, L. Wang, X. Xing, M. Du, and D. Song, "Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems," arXiv preprint arXiv:1908.01763, 2019.

[12] K. Huang, Y. Li, B. Wu, Z. Qin, and K. Ren, "Backdoor defense via decoupling the training process," arXiv preprint arXiv:2202.03423, 2022.

[13] H. Karimi, J. Nutini, and M. Schmidt, "Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition," in Joint European conference on machine learning and knowledge discovery in databases. Springer, 2016, pp. 795–811.

[14] K. Karra, C. Ashcraft, and N. Fendley, "The trojai software framework: An opensource tool for embedding trojans into deep learning models," arXiv preprint arXiv:2003.07233, 2020.

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

[16] V. Krishnan, A. Makdah, A. AlRahman, and F. Pasqualetti, "Lipschitz bounds and provably robust training by laplacian smoothing," Advances in Neural Information Processing Systems, vol. 33, pp. 10 924–10 935, 2020.

[17] A. Krizhevsky, G. Hinton et al., "Learning multiple layers of features from tiny images," 2009.

[18] Y. Le and X. S. Yang, "Tiny imagenet visual recognition challenge," 2015.

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

[20] H. Li, S. De, Z. Xu, C. Studer, H. Samet, and T. Goldstein, "Training quantized nets: A deeper understanding," Advances in Neural Information Processing Systems, vol. 30, 2017.

[21] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, "Anti-backdoor learning: Training clean models on poisoned data," in Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, 2021, pp. 14 900–14 912. [Online]. Available: https://proceedings.neurips.cc/paper/2021

[22] ——, "Neural attention distillation: Erasing backdoor triggers from deep neural networks," in 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021, 2021. [Online]. Available: https://openreview.net/forum?id=9l0K4OM-oXE

[23] Y. Li, B. Wu, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor learning: A survey," arXiv preprint arXiv:2007.08745, 2020.

[24] Y. Li, Y. Li, B. Wu, L. Li, R. He, and S. Lyu, "Backdoorbench: a comprehensive benchmark of backdoor attack and defense methods," in https://github.com/SCLBD/BackdoorBench, 2021.

[25] J. Lin, L. Xu, Y. Liu, and X. Zhang, "Composite backdoor attack for deep neural network by mixing existing benign features," in Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, 2020, pp. 113–131.

[26] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," Mathematical programming, vol. 45, no. 1-3, pp. 503–528, 1989.

[27] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in International Symposium on Research in Attacks, Intrusions, and Defenses. Springer, 2018, pp. 273–294.

[28] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "Abs: Scanning neural networks for back-doors by artificial brain stimulation," in Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 1265–1282.

[29] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in European Conference on Computer Vision. Springer, 2020, pp. 182–199.

[30] W. Ma, D. Wang, R. Sun, M. Xue, S. Wen, and Y. Xiang, "The" beatrix"resurrections: Robust backdoor detection via gram matrices," arXiv preprint arXiv:2209.11715, 2022.

[31] C. Michaelis, B. Mitzkus, R. Geirhos, E. Rusak, O. Bringmann, A. S. Ecker, M. Bethge, and W. Brendel, "Benchmarking robustness in object detection: Autonomous driving when winter is coming," arXiv preprint arXiv:1907.07484, 2019.

[32] A. Nguyen and A. Tran, "Wanet–imperceptible warping-based backdoor attack," arXiv preprint arXiv:2102.10369, 2021.

[33] R. Pang, H. Shen, X. Zhang, S. Ji, Y. Vorobeychik, X. Luo, A. Liu, and T. Wang, "A tale of evil twins: Adversarial inputs versus poisoned models," in Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, 2020, pp. 85–99.

[34] M. Peng, Z. Xiong, M. Sun, and P. Li, "Label-smoothed backdoor attack," arXiv preprint arXiv:2202.11203, 2022.

[35] X. Qi, T. Xie, S. Mahloujifar, and P. Mittal, "Circumventing backdoor defenses that are based on latent separability," arXiv preprint arXiv:2205.13613, 2022.

[36] A. Rajabi, B. Ramasubramanian, and R. Poovendran, "Trojan horse training for breaking defenses against backdoor attacks in deep learning," arXiv preprint arXiv:2203.15506, 2022.

[37] Y. Ren, L. Li, and J. Zhou, "Simtrojan: Stealthy backdoor attack," in 2021 IEEE International Conference on Image Processing (ICIP). IEEE, 2021, pp. 819–823.

[38] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," Advances in neural information processing systems, vol. 31, 2018.

[39] G. Shen, Y. Liu, G. Tao, S. An, Q. Xu, S. Cheng, S. Ma, and X. Zhang, "Backdoor scanning for deep neural networks through k-arm optimization," in International Conference on Machine Learning. PMLR, 2021, pp. 9525–9536.

[40] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," Journal of big data, vol. 6, no. 1, pp. 1–48, 2019.

[41] I. Shumailov, Z. Shumaylov, D. Kazhdan, Y. Zhao, N. Papernot, M. A. Erdogdu, and R. J. Anderson, "Manipulating sgd with data ordering attacks," Advances in Neural Information Processing Systems, vol. 34, pp. 18 021–18 032, 2021.

[42] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," arXiv preprint arXiv:1312.6199, 2013.

[43] G. Tao, G. Shen, Y. Liu, S. An, Q. Xu, S. Ma, P. Li, and X. Zhang, "Better trigger inversion optimization in backdoor scanning."

[44] D. Terjék, "Adversarial lipschitz regularization," arXiv preprint arXiv:1907.05681, 2019.

[45] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in 2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019, pp. 707–723.

[46] M. Weber, X. Xu, B. Karlaš, C. Zhang, and B. Li, "Rab: Provable robustness against backdoor attacks," arXiv preprint arXiv:2003.08904, 2020.

[47] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, "Detecting ai trojans using meta neural analysis," in 2021 IEEE Symposium on Security and Privacy (SP). IEEE, 2021, pp. 103–120.

[48] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Federated learning," Synthesis Lectures on Artificial Intelligence and Machine Learning, vol. 13, no. 3, pp. 1–207, 2019.

[49] Y.-Y. Yang, C. Rashtchian, H. Zhang, R. Salakhutdinov, and K. Chaudhuri, "Adversarial robustness through local lipschitzness," 2020.

[50] Z. Yao, A. Gholami, S. Shen, M. Mustafa, K. Keutzer, and M. Mahoney, "Adahessian: An adaptive second order optimizer for machine learning," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 12, 2021, pp. 10 665–10 673.

[51] Y. Zeng, H. Qiu, S. Guo, T. Zhang, M. Qiu, and B. Thuraisingham, "Deepsweep: An evaluation framework for mitigating dnn backdoor attacks using data augmentation," arXiv e-prints, pp. arXiv–2012, 2020.

[52] Z. Zhao, X. Chen, Y. Xuan, Y. Dong, D. Wang, and K. Liang, "Defeat: Deep hidden feature backdoor attacks by imperceptible perturbation and latent representation constraints," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 15 213–15 222.

[53] L. Zhu, R. Ning, C. Wang, C. Xin, and H. Wu, "Gangsweep: Sweep out neural backdoors by gan," in Proceedings of the 28th ACM International Conference on Multimedia, 2020, pp. 3173–3181.

APPENDIX

Due to space constraints, this Appendix contains only a selection of supplementary materials(Section A to E). For the rest set of appendices (Section F to J), please refer to the supplementary document available at this link[1].

A. Proof of Theorem 1

Theorem 1

Given a piece-wise linear function $\ell(\cdot) : [a, b] \rightarrow [0, 1]$ with a global optimum sit on a convex hull. Assume such a convex hull satisfies the largest update (step size times the largest gradient) in the convex hull is smaller than the shortest linear piece in the convex hull. After n iterations update, a gradient-based optimizer starting from a random initialization converges to the optimum with the probability:

$$1 - B_1^{-1}(b-a)^{-1}(4 - B_1B_2)^n(1 - B_1B_2)]$$

Proof:

As the input space is one-dimension, a gradient-based optimization on a piece-wise linear loss function can then be considered as a Markov chain(MC) [20]; If we use $\mathcal{A}$ to denote the equivalent MC, each linear piece represents a state (or node) in $\mathcal{A}$. The transition probability between two nodes is the probability that after one update step, the optimization could jump from the first node to the other one. Specifically, consider any two nodes (linear-piece), i and j, in $\mathcal{A}$, the transition probability of i^{th} node to the j^{th} means when the optimization is in the i^{th} node, the probability that after one step update, the optimization could move to the j^{th} linear piece. We use i connected with j to denote if the transition probability from i to j is not equal to zero. Then the adjacent matrix A can be written as:

$$A_{i,j} = \begin{cases} \frac{l_j}{\alpha \nabla_i} & i \text{ connected with } j \\ 0 & o.w \end{cases} \tag{4}$$

where $A_{i,j}$ indicate the transition probability of i^{th} node to the j^{th}, α is the update step size, l_j is the length of j^{th} linear piece in domain $[a, b]$, and ∇_i is the gradient of i^{th} linear piece.

The probabilities that the optimization converges to each linear piece could then be computed by the stationary distribution $\mathcal{S}$ of $\mathcal{A}$:

$$\mathcal{P} = \lim_{n \rightarrow \infty} \pi_0 A^n \tag{5}$$

where π_0 is the initial distribution, which is the initial probability for each linear piece: $\frac{l_i}{b-a}$.

Directly computing A^n is not easy; it requires diagonalization on a conditional Adjencent matrix. We then simplify $\mathcal{A}$ to a two-state MC.

Since the largest update (step size times the largest gradient) in the desired convex hull is smaller than the shortest linear piece in the desired convex hull. This indicates all nodes which represent those linear pieces in the desired convex hull formed a recurrent state; The transition probability that nodes from the desired convex hull to the node outside the desired

[1] https://drive.google.com/file/d/12EVjcrznWnhT-lpdLex8_eaZr3g11t_M/view?usp=sharing

convex hull are zero. We then can collapse every linear region in the desired convex hull into one state $\mathbb{P}$.

Similarly, we can collapse those linear regions which are not in the desired convex hull into one another state $\mathbb{Q}$; this is because our goal is to compute the stationary probability of $\mathbb{P}$, the details of the stationary probability for each linear regions which are not in the desired convex hull is not necessary. Then we can simplify our $\mathcal{A}$ into two states $\mathbb{P}, \mathbb{Q}$. The simplified MC is shown in Fig 8:

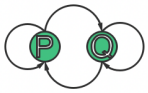


Figure 8: A Two-state Markov Chain. State $\mathcal{P}$ represents the linear regions in the desired convex hull, and state $\mathcal{Q}$ represents the linear region outside the desired convex hull.

The initial distribution can be written as follows:

$$\pi_0 = [\pi_0^{(\mathbb{P})}, \pi_0^{(\mathbb{Q})}] = [\frac{\sum\limits_{\exists i \in \mathbb{P}} l_i}{b-a}, \frac{\sum\limits_{\exists i \in \mathbb{Q}} l_i}{b-a}] \tag{6}$$

We now consider the 2×2 adjacent matrix $A_{\mathbb{P},\mathbb{Q}}$. Since the desired convex hull satisfies the largest update in the convex hull is smaller than the shortest linear piece in the convex hull. It can be infers that $A_{\mathbb{P}\to\mathbb{P}} = 1$, and $A_{\mathbb{P}\to\mathbb{Q}} = 0$.

And now let's consider the entry $A_{\mathbb{Q}\to\mathbb{P}}$,

$$\begin{aligned} A_{\mathbb{Q}\to\mathbb{P}} &= \frac{\sum\limits_{i\in\mathbb{Q}\ \&\ \text{connect}\ \mathbb{P}} l_i}{\sum\limits_{i\in\mathbb{Q}} l_i} \cdot \sum_{i\in\mathbb{Q}\ \&\ \text{connect}\ \mathbb{P}} \frac{l_i}{\sum\limits_{j\in\mathbb{Q}\ \&\ \text{connect}\ \mathbb{P}} l_j} \cdot \frac{l_i}{\alpha\nabla_i} \\ &= \frac{1}{\sum\limits_{i\in\mathbb{Q}} l_i} \cdot \sum_{i\in\mathbb{Q}\ \&\ \text{connect}\ \mathbb{P}} l_i \cdot \frac{l_i}{\alpha\nabla_i} \\ &= \frac{1}{\sum\limits_{i\in\mathbb{Q}} l_i} \cdot \sum_{i\in\mathbb{Q}\ \&\ \text{connect}\ \mathbb{P}} \frac{l_i^2}{\alpha\nabla_i} \end{aligned} \tag{7}$$

The last entry $A_{\mathbb{Q}\to\mathbb{Q}}$ then becomes:

$$A_{\mathbb{Q}\to\mathbb{Q}} = 1 - \frac{1}{\sum\limits_{i\in\mathbb{Q}} l_i} \cdot \sum_{i\in\mathbb{Q}\ \&\ \text{connect}\ \mathbb{P}} \frac{l_i^2}{\alpha\nabla_i}$$

Then the adjacent matrix becomes:

$$A_{\mathbb{P},\mathbb{Q}} = \begin{pmatrix} 1 & 0 \\ \frac{1}{\sum\limits_{i\in\mathbb{Q}} l_i} \cdot \sum\limits_{i\in\mathbb{Q}\ \&\ \text{connect}\ \mathbb{P}} \frac{l_i^2}{\alpha\nabla_i} & 1 - \frac{1}{\sum\limits_{i\in\mathbb{Q}} l_i} \cdot \sum\limits_{i\in\mathbb{Q}\ \&\ \text{connect}\ \mathbb{P}} \frac{l_i^2}{\alpha\nabla_i} \end{pmatrix}$$

As a 2×2 matrix, We can apply Hamilton-Cayley theorem to compute $A^n_{\mathbb{P},\mathbb{Q}}$:

$$\begin{aligned} A^n_{\mathbb{P},\mathbb{Q}} &= \text{Tr}^n(A_{\mathbb{P},\mathbb{Q}}) \cdot A_{\mathbb{P},\mathbb{Q}} \\ &= (4 - \frac{1}{\sum\limits_{i\in\mathbb{Q}} l_i} \cdot \sum_{i\in\mathbb{Q}\ \&\ \text{connect}\ \mathbb{P}} \frac{l_i^2}{\alpha^2\nabla_i^2})^n \cdot A_{\mathbb{P},\mathbb{Q}} \end{aligned} \tag{8}$$

To make the equation more compact, we denote

$B_1 = \frac{1}{\sum\limits_{i\in\mathbb{Q}} l_i}$, which indicate the extent of area under the convex hull

$B_2 = \sum\limits_{i\in\mathbb{Q}\ \&\ \text{connect}\ \mathbb{P}} \frac{l_i^2}{\alpha\nabla_i}$, indicates the extent of the likelihood the linear pieces outside the convex hull can jump into the convex hull.

Then the stationary distribution of $\mathcal{A}$ after n iteration will be:

$$\begin{aligned} \mathcal{P} &= \pi_0 A^n_{\mathbb{P},\mathbb{Q}} \\ &= \pi_0 \begin{pmatrix} (4-B_1B_2)^n & 0 \\ (4-B_1B_2)^n B_1B_2 & (4-B_1B_2)^n(1-B_1B_2) \end{pmatrix} \\ &= [1 - B_1^{-1}(b-a)^{-1}, B_1^{-1}(b-a)^{-1}] \cdot \\ &\begin{pmatrix} (4-B_1B_2)^n & 0 \\ (4-B_1B_2)^n B_1B_2 & (4-B_1B_2)^n(1-B_1B_2) \end{pmatrix} \\ &= [\frac{(4-B_1B_2)^{2n}(B_2 - B_1^{-1})}{b-a}, \\ &B_1^{-1}(b-a)^{-1}(4-B_1B_2)^n(1-B_1B_2)] \end{aligned} \tag{9}$$

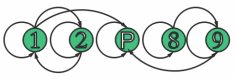
The stationary probability for state $\mathbb{Q}$ is equal to $B_1^{-1}(b-a)^{-1}(4-B_1B_2)^n(1-B_1B_2)$. This shows a negative relationship between the stationary probability for state $\mathbb{Q}$ and B_1, B_2. Similarly, stationary probability for state $\mathbb{P}$ is equal to $1 - B_1^{-1}(b-a)^{-1}(4-B_1B_2)^n(1-B_1B_2)$. It shows a positive relationship between the stationary probability for state $\mathbb{P}$ and B_1, B_2.

After the optimization jump into the desired convex hull, state $\mathcal{P}$ and the step size followed by the assumption in the theorem, as long as n is large enough and the step size followed by the assumption in the theorem, the optimization will converge to the desired optimum.

We further illustrate Theorem 1 using a piecewise linear loss function in one-dimensional input space (Fig 34):

Each linear piece represents one state on the MC.

We can further reduce the MC by collapsing the desired convex hull into one single state $\mathbb{P}$.



For the rest of the nodes not in the desired convex hull, we can reduce them into one state $\mathbb{Q}$ (Fig 8).

End of proof.

B. Proof of Theorem 2

Before we elaborate our theorem, we need to first formally define some concepts and a Lemma from [49]:

Definition 3 (Astuteness). *A classifier $f : \mathcal{X} \to \mathcal{Y}$ is astute at an input sample x, if the predicted label by f is the same as the true label: $\hat{y} = z(f(x)) = y$.*

Definition 4 (r-local minimum). *A function $f : \mathcal{X} \to \mathbb{R}$ has a (unique) r-local minimum at $x^\star$, if there is no other x on which f gets lower or equal value than what can get on x^*, within the ball centered on x^* with radius r, i.e., $f(x) > f(x^*), \forall x, \|x - x^*\|_2 \leq r$.*

Definition 5 (Increasing rate and relaxation function). *Given a function $f : \mathcal{X} \to \mathbb{R}$ with a r-local minimum at $x^\star$, we define that f has an increasing rate of κ at $x^\star$, if there exists some $\kappa \geq 0$ and $c_\kappa \geq 0$, such that $f(x) - f(x^\star) \geq \sup_{c_\kappa,\kappa} c_\kappa \cdot \|x - x^\star\|_2^\kappa$, when $\|x - x^\star\| \leq r$. Accordingly, we refer the function $\bar{g}(x) = c_\kappa \cdot \|x - x^\star\|_2^\kappa$ as the relaxation function of f at x.*

Definition 6 (Local Lipschitz constant). *For a function $f : \mathcal{X} \to \mathcal{Y}$, a given input x and a pre-defined radius r, if $L(f, \mathcal{X}_{x,r})$ exists and is finite, where $\mathcal{X}_{x,r} = \{x' : \|x' - x\|_2 < r\}$ and*

$$L(f, \mathcal{X}_{x,r}) = \sup_{x_1,x_2 \in \mathcal{X}_{x,r}} \frac{||f(x_2) - f(x_1)||_2}{||x_2 - x_1||_2}, \tag{10}$$

we define $L(f, \mathcal{X}_{x,r})$ as the local Lipschitz constant of x with radius r for function f.

Lemma 1. *Consider the data distribution X, and assume the minimum l_2 norm between any two different class data is r. If a function is astuteness in X, then f has a local Lipschitz constant of r' around any $x \in X$ such that $r' \geq r$*

Theorem 2

If the noise $\epsilon \sim \mathcal{N}(0, 1)$ (i.e., the white noise), and $c < \|x' - x\|_2 \cdot \frac{\Gamma\left(\frac{|m^*|}{2}\right)}{\sqrt{2}\Gamma\left(\frac{|m^*|+1}{2}\right)}$, where $|m^*|$ is the l_1 norm (i.e., the size) of the trigger, Γ is the Euler's gamma function. a model backdoor attacked by a backdoor attack and enhanced by GRASP using the training data points $(x, y), (x', y_t)$ and (x^*, y)) has a greater local Lipschitz constant around x than the model backdoored by the same attack without the GRASP enhancement using the training data points $(x, y), (x', y_t)$.

Similarly, if $\epsilon \sim uniform(-1, 1)$, and $c < \|x' - x\|_2$, the GRASP-enhanced model has greater local Lipschitz constant around x than the model without the enhancement.

Proof:

Consider in the BadNet data contamination, recall that trigger l_1 norm is m^*, the subspace $V \in \mathbb{R}^{m^*}$ is set of those dimensions which the mask matrix $\boldsymbol{M}$ has non-zero entry.

$$E(||A(\boldsymbol{x}, \boldsymbol{M}, \boldsymbol{\Delta}) - x||_2) = 2r_{BadNet}$$

where r_{BadNet} is the expectation of robust radius, which is defined in model contaminated by BadNet and trigger-inserted data

In GRASP, we can choose a random noise ϵ. First let's consider $\epsilon = c\mathcal{N}(0, I)$ is added to trojan input only on the subspace V. Then, the expectation of magnitude of this noise is:

$$E(\|\epsilon\|_2) = \frac{\sqrt{2}\Gamma\left(\frac{|m^*|+1}{2}\right)}{\Gamma\left(\frac{|m^*|}{2}\right)} cI = 2r_{GRASP}$$

which implies:

$$r_{GRASP} = c \cdot \frac{\sqrt{2}\Gamma\left(\frac{|m^*|+1}{2}\right)}{2\Gamma\left(\frac{|m^*|}{2}\right)}$$

Similarly, when $\epsilon = c \cdot unif(-1, 1)$:

$$E(\|\epsilon\|_2) = \frac{c}{2} = 2r_{GRASP}$$

And

$$r_{GRASP} = \frac{c}{4}$$

where Γ is the Euler's gamma function, r_{GRASP} is the expectation of robust radius, which is defined in model contaminated by GRASP and trigger-inserted data. And c is the noise scalar parameter.

When $\epsilon = c\mathcal{N}(0, I)$ and let c :

$$c < ||x' - x||_2 \cdot \frac{\Gamma\left(\frac{|m^*|}{2}\right)}{\sqrt{2}\Gamma\left(\frac{|m^*|+1}{2}\right)}$$

when $\epsilon = c \cdot unif(-1, 1)$, we let:

$$c < \frac{||x' - x||_2}{4}$$

Then we have,

$$\begin{aligned} 2r_{GRASP} &= E(\|\epsilon\|_2) \\ &< E(||A(\boldsymbol{x}, \boldsymbol{M}, \boldsymbol{\Delta}) - x||_2) = 2r_{BadNet} \end{aligned}$$

According to the Lemma 1, we have the local lipchitz constant around $A(\boldsymbol{x}, \boldsymbol{M}, \boldsymbol{\Delta})$ for GRASP is $\frac{1}{r_{GRASP}}$, and for the BadNet contaminating backdoor attack is $\frac{1}{r_{BadNet}}$. And since $r_{GRASP} < r_{BadNet}$, so $\frac{1}{r_{GRASP}} > \frac{1}{r_{BadNet}}$. So GRASP can achieve a larger local lipschitz constant around trojan data than badNet.

End of proof.

C. PL Condition and Convergence Rate in High-Dimensional Non-Convex Optimization

Now we consider the target function as high-dimensional non-convex but satisfies the proximal-PL condition [13], which is often considered in the theoretical analysis of neural networks. Formally, the proximal-PL condition is defined below.

Definition 7 (Proximal-PL condition). *We consider the optimization problem in the form:*

$$\underset{x\in\mathbb{R}^d}{\operatorname{argmin}} F(x) = f(x) + g(x), \tag{11}$$

where f is a differentiable function with an L-Lipschitz continuous gradient and g is a simple but potentially non-smooth convex function [2]. To analyze the proximal-gradient algorithms (i.e., a more general form of the Projected Gradient Descent (PGD)), a natural generalization of the PL inequality is that there exists $\mu > 0$ satisfying:

$$\frac{1}{2}\mathcal{D}_g(x, L) \geq \mu\left(F(x) - F^*\right) \tag{12}$$

where

$$\mathcal{D}_g(x,\alpha) \equiv -2\alpha \min_y \left[\langle \nabla f(x), y-x\rangle + \frac{\alpha}{2}\|y-x\|^2 + g(y) - g(x)\right].$$

We now present Theorem 3 from [13], which demonstrates that the proximal-PL condition is sufficient for the proximal-gradient method to achieve a global linear convergence rate.

Theorem 3. *Consider the optimization problem in Eq. 11, where f has an L-Lipschitz continuous gradient (Eq. 12), F has a non-empty solution set $\mathcal{X}^*$, g is convex, and F satisfies the proximal-PL inequality. Then the proximal gradient method with a step size of $1/L$ converges linearly to the optimal value F^*:*

$$F\left(x_k\right) - F^* \leq \left(1 - \frac{\mu}{L}\right)^k \left[F\left(x_0\right) - F^*\right] \tag{13}$$

Theorem 3 indicates a negative relationship between the Lipschitz constant (L) of the target function and the convergence rate to the local optimum (trigger), i.e., the difference in target function values between the optimal point and the actual trigger-inserted point. Previous research [4] has shown that the second-order optimizer has the same lower bound of convergence rate as the first-order optimizer under the PL condition. Many existing studies [49], [44], [16] have demonstrated that in neural networks, a lower Lipschitz constant implies higher robustness of the model. Therefore, combining these findings with Theorem 3, we conclude that gradient-based trigger inversion methods perform well on triggers with large effective radius.

D. Impact on β with More Complexity Trigger Type

In this section, we primarily continue the discussion about the choice of β from Section IV-C, and further test two more complex trigger patterns (composite [25], and reflection [29]). As depicted in Fig 9, 10, the complexity of the trigger pattern does not significantly impact the choice of β. A β value of 10% remains suitable even with these complex triggers.

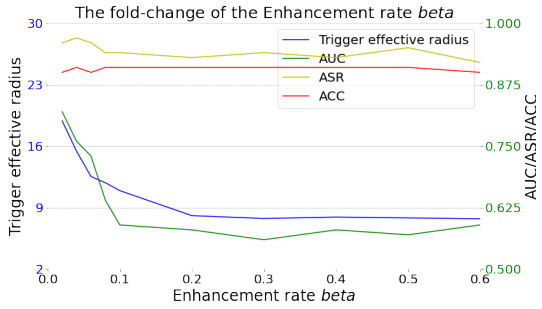


Figure 9: Impact of β with the composite [25] trigger pattern.

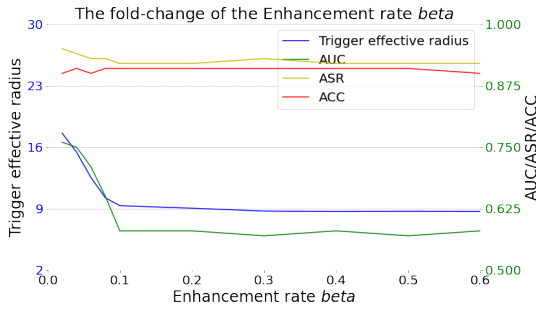


Figure 10: Impact of β with the Reflection [29] trigger pattern

E. Impact on Different Type of Corruption

Figures 17 to 31 document the comparative impact of 15 different types of corruptions from the imagecorruptions benchmark on a GRASP-enhanced BadNet versus BadNet. More specifically, we employed the ResNet-101 neural network architecture to train two distinct models on the CIFAR-10 dataset: one contaminated with a backdoor using the BadNet methodology, and another utilizing a backdoor implemented via the GRASP-enhanced BadNet approach. we present the ASR for these corrupted testing data, subjected to 15 corruptions in five varying degrees of severity on trigger-inserted testing data.

[2]Typical examples of the simple function g include a scaled ℓ_1-norm of the parameter vectors (the size of the trigger), $g(x) = \lambda\|x\|_1$, and indicator functions that are zero if x lies in a simple convex set, and are infinity otherwise.