Backdoor Scanning for Deep Neural Networks through K-Arm Optimization

Guangyu Shen [*1] **Yingqi Liu** [*1] **Guanhong Tao** [1] **Shengwei An** [1] **Qiuling Xu** [1] **Siyuan Cheng** [1] **Shiqing Ma** [2] **Xiangyu Zhang** [1]

Abstract

Back-door attack poses a severe threat to deep learning systems. It injects hidden malicious behaviors to a model such that any input stamped with a special pattern can trigger such behaviors. Detecting back-door is hence of pressing need. Many existing defense techniques use optimization to generate the smallest input pattern that forces the model to misclassify a set of benign inputs injected with the pattern to a target label. However, the complexity is quadratic to the number of class labels such that they can hardly handle models with many classes. Inspired by Multi-Arm Bandit in Reinforcement Learning, we propose a K-Arm optimization method for backdoor detection. By iteratively and stochastically selecting the most promising labels for optimization with the guidance of an objective function, we substantially reduce the complexity, allowing to handle models with many classes. Moreover, by iteratively refining the selection of labels to optimize, it substantially mitigates the uncertainty in choosing the right labels, improving detection accuracy. At the time of submission, the evaluation of our method on over 4000 models in the IARPA TrojAI competition from round 1 to the latest round 4 achieves top performance on the leaderboard. Our technique also supersedes five state-of-the-art techniques in terms of accuracy and the scanning time needed. The code of our work is available at https://github.com/PurduePAML/K-ARM_Backdoor_Optimization

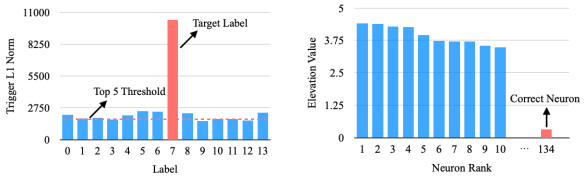
1. Introduction

The semantics of a deep neural network is determined by model parameters that are not interpretable. Trojan (back-door) attack exploits the uninterpretability and injects malicious hidden behaviors to neural networks. To activate back-door behavior, the attacker stamps a *trigger* to a benign input and passes the stamped input to the trojaned model, which then misclassifies the input to the *target label*. When benign inputs are provided, the trojaned model has comparable accuracy as the original one. The feasibility of trojan attack has been demonstrated by many existing works. For example, data poisoning (Gu et al., 2017) directly uses stamped inputs in training to inject back-door. Neuron hijacking (Liu et al., 2018b) compromises a small number of selected neurons by changing their associated weight values through input reverse engineering and retraining. Clean-label attack (Shafahi et al., 2018) injects malicious features to the target class samples instead of victim class samples, and hence is more stealthy. More discussion can be found in the related work section.

Realizing the prominent threat, researchers have developed a number of defense techniques that range from detecting malicious (stamped) inputs at runtime (Ma & Liu, 2019) to offline model scanning for possible back-doors (Liu et al., 2019; Wang et al., 2019; Kolouri et al., 2020). The former is an on-the-fly technique and requires the presence of malicious inputs. The latter determines if a given model contains any backdoor. It usually assumes a small set of benign inputs for all the classes of the model but not any malicious inputs. Existing scanners usually consider two types of backdoors. The first is *universal backdoor* that causes misclassification (to the target label) for benign samples from any class when they are stamped with the trigger. The second is *label-specific backdoor* that only causes misclassification of benign samples from a specific *victim class* to the target label, when they are stamped with the trigger. *Neural Cleanse* (NC) (Wang et al., 2019) uses optimization to derive a trigger for each class and observes if there is any trigger that is exceptionally small and hence likely injected instead of naturally occurring feature. *Artificial Brain Stimulation* (ABS) (Liu et al., 2019) systematically intercepts and changes internal neuron activation values on benign inputs, and then observes if consistent misclassification can be induced. If so, the corresponding neurons are considered compromised and used to reverse engineer a trigger. More existing techniques are discussed in the related work section.

[*]Equal contribution [1]Department of Computer Science, Purdue University, West Lafayette, IN, USA [2]Department of Computer Science, Rutgers University, Piscataway, NJ, USA. Correspondence to: Guangyu Shen <shen447@purdue.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.



(a) Pre-selection (b) ABS

Figure 1. **Motivation cases:** (a) illustrates pre-selection fails to identify backdoor in Model #56 in TrojAI round 2; (b) shows that ABS fails to identify backdoor in Model #13 in TrojAI round 1.

Although the effectiveness of existing solutions has been demonstrated, they have various limitations. In particular, since the target label is unknown beforehand, scanners such as NC try to scan all labels. If the backdoor is label-specific, the computation complexity is quadratic. As such they can hardly handle models with many classes. For example, NC cannot finish scanning a TrojAI round 2 model with 23 classes within 15 hours. Techniques like ABS leverages additional analysis to pre-select a set of labels/neurons to optimize. However, their effectiveness hinges on the correctness of pre-selection.

We propose a new back-door scanning method that can handle models with many classes and has better effectiveness and efficiency than existing solutions. Inspired by *K-Arm bandit* (Auer et al., 2002) in Reinforcement Learning that optimizes decision making with a large number of possible options, we propose a K-Arm backdoor scanner. Instead of optimizing for all the labels one-by-one, the process is divided to many rounds and in each round, our algorithm selects one to optimize for a small number of epochs. The selection is stochastic, guided by an objective function. The function measures the past progress of a candidate label, e.g., how fast a small trigger can be generated to misclassify stamped inputs to the label, as a trigger is generally easy to optimize if the label is trojaned, and how small the trigger is. The stochastic nature of the method ensures that even if the true target label is not selected for the current round, it still has a good chance to be selected later. To our knowledge, we are the first to bring reinforcement learning (K-Arm Bandit) into the neural backdoor detection domain and substantially improve the scanner's efficiency and capability. Natural features sometimes behave similarly to backdoors. To distinguish the two, we develop a symmetric optimization algorithm that piggy-backs on the K-Arm backbone. It leverages the following observation: while it is easy to optimize a trigger that flips victim label to target label, the inverse (i.e., optimize a trigger that flips target label to victim label) is difficult; natural features, however, do not have this property.

We evaluate our prototype on 4000 models from IARPA TrojAI round 1 to the latest round 4 competitions, and a few complex models on ImageNet. Our technique achieved top performance on the TrojAI leaderboard and reached the round targets on the TrojAI test server for all rounds. It is substantially more effective than the state-of-the-art techniques NC, ABS, and ULP (Kolouri et al., 2020) by having 31%, 20%, and 27% better accuracy, respectively. In addition, its scanning time is a few times to orders of magnitude smaller than other optimization based methods, especially in scanning label-specific backdoors.

2. Related Work

Besides the ones mentioned in the introduction, we further briefly discuss additional related work and our threat model.

Trojan Attack. Several data-poisoning like attacks (Gu et al., 2017; Liu et al., 2018b) utilize patch/watermark triggers. Clean-label attacks (Shafahi et al., 2018; Saha et al., 2020; Turner et al., 2019; Zhao et al., 2020; Zhu et al., 2019) inject back-door without changing data label. Salem et al. (2020); Nguyen & Tran (2020) leveraged generative models to construct dynamic triggers with random patterns and locations for specific samples. Composite attack (Lin et al., 2020) uses natural features from multiple labels as triggers. Bit flipping (Rakin et al., 2019; 2020) injects malicious behaviors by flipping bits in model weights. Trojan attacks have been developed for transfer learning (Rezaei & Liu, 2019; Wang et al., 2018; Yao et al., 2019), federated learning (Bagdasaryan et al., 2020; Xie et al., 2019; Wang et al., 2020b) and NLP tasks (Chen et al., 2020; Sun, 2020).

Existing Detection. ULP (Kolouri et al., 2020) trains a classifier to determine if a model is trojaned. It leverages a large pool of benign and trojaned models to learn a set of universal input patterns that can lead to different logits for benign and trojaned models. The classifier is then trained on these logits. Similar to ULP (Kolouri et al., 2020), researchers in (Huang et al., 2020) proposed one-pixel signature. They trained a classifier to predict the model's benignity based on their one-pixel signature. Qiao et al. (2019) proposed to generate trigger distribution. Zhang et al. (2020); Wang et al. (2020c) leveraged the differences of adversarial examples for benign and trojaned models to detect backdoors. TABOR (Guo et al., 2019) used explainable AI techniques to scan backdoors. Xu et al. (2019) detected backdoors using Meta Neural Analysis. Liu et al. (2018a) combined pruning and fine-tuning to weaken or even eliminate backdoors. Wang et al. (2020a) certified model robustness against backdoor via randomized smoothing. Chan & Ong (2019); Gao et al. (2019); Chen et al. (2018); Chou et al. (2020); Du et al. (2019); Liu et al. (2017); Ma & Liu (2019) aimed to detect if a provided input contains trigger. Comprehensive surveys of backdoor learning can be found at (Li et al., 2020a;b)

Multi-Arm Bandit. Multi-Arm Bandit (MAB) describes the dilemma of making a sequence of decisions to maximize

(a) (R4 model #556) victim class #13 input + trigger

(b) (R4 model #556) target class #1 input

(c) (R4 model #262) class #4 input + generated natural feature

(d) (R4 model #262) class #20 input + generated natural feature

Figure 2. **Motivation cases:** (a) illustrates a victim class #13 input of a round 4 (R4) *trojaned* model stamped with trigger generated by K-Arm, yielding the classification result of label #1; (b) shows a target class #1 input for the same model; (c) shows a class #4 input of a *clean* R4 model stamped with natural features generated by K-Arm, yielding label 20; (d) shows a class #20 input stamped with generated natural features for the same model in (c), yielding label 4.

reward, which has an unknown distribution. It has been thoroughly studied in (Auer et al., 2002). Many solutions are proposed to tackle this problem, such as Upper Confidence Bound (UCB) (Auer, 2002), ϵ-greedy (Watkins, 1989), etc. MAB is a general idea with many applications, Our design is inspired by MAB and unique for backdoor detection.

Threat Model We consider a standard setting in the backdoor scanning. Given a model and a small set of clean images without trigger information for each class (less than 20), the defender is required to identify whether the model is trojaned or not. In this paper, we mainly discuss the backdoor with limited size on the propose of stealthiness, such as patch triggers (Liu et al., 2018b) or small perturbations (Saha et al., 2020). The injected backdoor can be static (Gu et al., 2017), input aware dynamic (Nguyen & Tran, 2020), label-specific or global. Large triggers such as the composite attack (Lin et al., 2020) and filter triggers (Liu et al., 2019; Cheng et al., 2020) are out of the scope. We will leave it to the future work.

3. Motivation

In this section, we discuss the limitations of existing optimization based backdoor scanners and motivate ours.

NC (Wang et al., 2019) cannot handle models with many classes. Assume a model has N classes. Since the target label is unknown, to detect universal backdoors, NC considers each of the N labels could be the target label and optimizes a trigger that flips benign samples from any class to the label. To detect label specific backdoors, it considers each pair of labels could be the victim and target labels, and optimizes a trigger to flip only samples of the victim class to the target label. It then checks if there is an exceptionally small trigger (among all those generated). If so, the model is considered having a backdoor. The computation complexity is hence $\mathcal{O}(N)$ for universal backdoors and $\mathcal{O}(N^2)$ for label specific backdoors. Our experiment (in Section 5) shows that to scan a model on ImageNet with a universal backdoor, NC needs more than 55 hours. It certainly cannot handle label-specific backdoors on such models.

Pre-selection may miss the correct label(s). To address the above limitation, a pre-selection strategy was proposed in (Wang et al., 2019) to select a small subset of labels to proceed after 10 steps of optimization. Specifically, it selects the m smallest triggers to continue. However, its effectiveness hinges on the correctness of pre-selection, which is difficult to achieve due to the uncertainty in optimization. Fig.1a illustrates how pre-selection fails on a TrojAI round 2 model (with a universal backdoor). Due to the small time budget allowed for scanning a TrojAI model (600s in round 2), top 5 labels are pre-selected out of 14. Observe that the trigger size of the target label is still much larger than most of the other labels after 10 steps and precluded. The situation is aggravated when the number of classes is large and backdoors are label-specific. In fact, our results show that pre-selection can only achieve 58% accuracy on average in TrojAI rounds 1 to 4 training sets.

ABS may select the wrong neurons in stimulation analysis. ABS (Liu et al., 2019) avoids optimizing for individual labels/label-pairs. It systematically enlarges internal neuron activation values for benign inputs and observes if consistent misclassification (to a certain label) can be achieved. If so, the neurons are considered potentially compromised by trojaning. It then uses optimization to generate a trigger by maximizing the activation values of these neurons. A model is considered trojaned if the generated trigger can cause the intended misclassification. It works for both universal and label-specific backdoors. Its effectiveness hinges on correctly identifying the compromised neurons, which has inherent uncertainty as well. Fig. 1b shows that for a trojaned model #13 in TrojAI round 1, the top 10 neurons that have the largest elevation for the target label logits when stimulated (and hence cause misclassification to the target label) do not include the truly compromised neuron, which is ranked 134 by the stimulation analysis. As such, trigger generation based on the top 10 neurons fails to derive the real trigger. In our experiment, ABS can only achieve 69% detection accuracy on average for TrojAI rounds 1 to 4.

Existing scanners cannot distinguish triggers from natural features. Natural features can induce misclassification

in a way similar to backdoor triggers. For example, stamping a dog nose to cat images may induce misclassification to dog. As such, optimization based trigger generation like NC and ABS may generate natural features as triggers. Distinguishing the two is important as misclassification caused by natural features is inevitable and a model should not be blamed for their presence; and correctly separating natural features from injected triggers allows model end users to employ proper counter measures. Many TrojAI models have natural features that behave like triggers. Fig. 2c presents a benign TrojAI model #262 in round 4, with a class #4 input stamped with the natural features generated by K-Arm (i.e., the pixel pattern inside the red box). It causes the model to misclassify to label 20 (shown in (d)). The inputs to TrojAI models are traffic-sign like foreground objects (e.g., the triangle in Fig. 2a and the octagon in Fig. 2b) with randomly chosen street-view background. More information can be found in Appendix.D. Observe classes #4 and #20 are similar, and the generated features in (c) resemble the central symbol of class #20, which explains the misclassification. Both NC and ABS consider the natural features as a trigger and report the model as trojaned.

Our Method. From the above discussion, we can observe that *a key challenge lies in the inherent uncertainty in selecting the appropriate label (in NC) or neuron(s) (in ABS) to perform optimization*. An exhaustive method like NC without selection is not effective for complex models while pre-selection and ABS making deterministic choices may fail to select the right one. The overarching idea of our method is to formulate the whole procedure as a stochastic process in which we continue to make selection at each round. Here and in the rest of the paper, an optimization round does not mean an optimization epoch in the traditional sense but rather finding a smaller trigger (that can cause misclassification). In particular, a selected label/label-pair/neuron that continues to perform well over time (i.e., whose trigger has been easy to optimize) will have a high probability to be selected in the new round. A label/label-pair/neuron that does not get selected in one round has a probability to be selected in the future. The goal is to allow the true positive to eventually stand out.

Specifically, we start with a *warm-up* phase in which we optimize each label (to generate trigger) for a very small number of rounds (2 in this paper). We retain a history of trigger size variation for each label. Then we start the *selective optimization*. At each round of selective optimization, we select the label that has the best performance over-time. We use an objective function to measure the performance. For the moment, readers can intuitively consider that we utilize the derivative of trigger size (i.e., how fast the trigger size changes). Note that for a clean label, although the optimization may produce a small trigger at the beginning, it cannot achieve substantial size reduction over time. Therefore, its performance degrades and tends to be replaced. In contrast, although the target label may not perform well at the beginning and hence not be selected, it is eventually selected when the other optimizations get stuck.

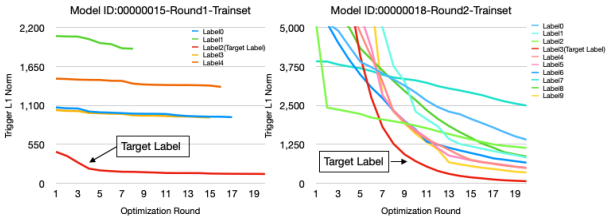


Figure 3. Trigger size variations over optimization rounds

Fig. 3 shows the trigger size variations of all labels over multiple rounds of optimization for two models from TrojAI. Observe that after the first round, the target label has the smallest trigger for model $\#15$ and hence pre-selection handles it correctly. In contrast for model $\#18$, the target label's trigger is very large and precluded (by pre-selection) from further optimization. Observe that it remains larger than many others till round 5. However, with our method, it eventually stands out and exposes the backdoor.

The algorithm also seamlessly facilitates separation of natural features and backdoor triggers. Specifically, when two benign classes A and B are similar (e.g., cat and dog), small natural features (of A) can be identified to flip B samples to A when they are stamped with the features, just like a trigger. Observe that since the two classes are similar, small natural features can be easily identified to flip A to B as well. For example in Fig 2d, the generated trigger to flip class #20 to #4 has a similar small size as that in Fig. 2c. In contrast, such symmetry is unlikely for real backdoors as generating a trigger to flip the target label to the victim label tends to be difficult. For example, Fig. 2a shows a trigger (the pixel in the red box) for model #556 in round 4 that has a label-specific backdoor from class #13 to class #1. It is sufficient to flip all class #13 inputs to class #1 (i.e., Fig. 2b). However, due to the differences of the two classes, flipping class #1 inputs to class #13 is much more difficult. Hence, we extend the algorithm such that when it decides to optimize for a victim-target label pair, it also sufficiently optimizes along the opposite direction to check symmetry.

4. Design

Fig. 4 presents the overview of our technique. On the left is the *trigger optimizer* (Section 4.1) that performs one round of trigger optimization at a time. In each round the optimizer generates a smaller trigger (than before) that causes a given set of benign samples to be misclassified to a target label, or returns failure when such a trigger cannot be found within a fixed number of epochs. On the right is the *K-Arm scheduler* (Section 4.2) that decides which *arm* should be

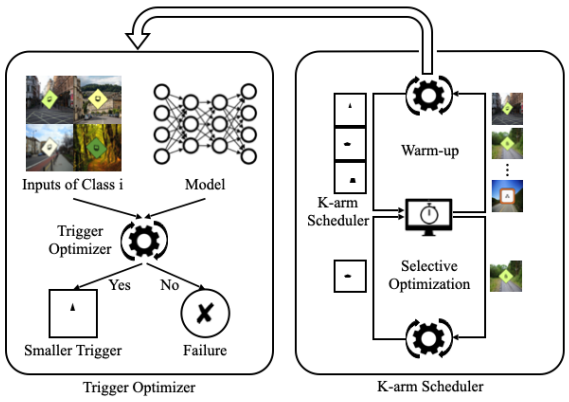


Figure 4. K-arm optimization workflow

optimized next. Assume a model has N classes. To identify universal backdoor, we create N (optimization) arms, each having one of the N labels as the target label and aiming to generate a trigger to flip benign samples from the remaining $N-1$ classes to the target label. To identify label-specific backdoor, we create $N \times (N-1)$ arms (i.e., all the pairwise combinations), each aiming to flip samples of a victim class to a target label. Hence, the scheduler selects from the $K = N + N \times (N-1)$ arms. In the diagram, there are two cycles inside the scheduler representing two optimization phases. The top cycle denotes the *warm-up* phase that optimizes all arms for two rounds. The scheduler receives and retains the generated trigger information for later use. The bottom cycle denotes the later *selective optimization* phase, in which one selected arm is optimized in each round. The selective optimization terminates when we can get a sufficiently small trigger or the time budget runs out. To improve efficiency, the scheduler is facilitated by a prescreening phase to reduce unnecessary arms (Section 4.3). It also considers symmetry during selection to distinguish nature features from triggers (Section 4.4).

4.1. Trigger Optimizer

In each round, the trigger optimizer optimizes one selected arm, generating a trigger for the target label of the arm. Specifically, a trigger T is composed of two parts: pattern P and mask M with the former deciding the input values of a trigger and the latter deciding the shape/position of the trigger. Given a clean input x and a trigger, the stamped input $\hat{x}$ is defined as follows.

$$\hat{x} = (1 - M) \cdot x + M \cdot P \tag{1}$$

Here, operator $\cdot$ stands for the element-wise production. Given an x of dimensions $[C, H, W]$, the dimensions of pattern P and M are identical to x's. The values of P are in the range of $[0, 255]$ and the values of M are in the range of $[0, 1]$. Intuitively, stamping a trigger is by mixing x and P through the mask M. Given a model $\mathcal{F}$, a target label t, and a set of inputs X, the trigger optimization for t is defined as follows.

$$\min_{P,M}(\mathcal{L}(t, \mathcal{F}((1 - M) \cdot X + M \cdot P)) + \alpha \|M\|_1), \forall x \in X \tag{2}$$

For an arm of generating universal trigger, X contains a set of clean inputs from classes other than t; for an arm of generating label-specific trigger, X contains a set of clean inputs from the victim class. $\mathcal{L}$ stands for the cross-entropy loss function. Hyper-parameter α balances the attack success rate and the size of the optimized trigger. The optimizer finishes a round and returns if the current trigger T satisfies the following condition.

$$Acc(\hat{X}, t) \geq \theta \ \ and \ \|M\|_1 < \|M_p\|_1$$

Intuitively, the attack success rate with the trigger needs to be greater than a threshold θ, which is 0.99 in this paper, meaning samples stamped with the trigger have higher than 99% chance to be classified to t, and the current trigger is smaller than the previous one M_p. The optimizer may return failure for the current round when the budget for the label runs out (which is 10 epochs in this paper).

4.2. K-Arm Scheduler

To handle uncertainty in arm selection, we leverage the ϵ-greedy algorithm (Watkins, 1989) to introduce randomness in our selection. The idea is to draw a random sample from a distribution, which is a uniform distribution from 0 to 1 in this paper. If the sample is larger than a threshold ϵ, we rely on an objective function to make the selection; otherwise, a random arm is selected. The procedure of selecting label L is formally defined as follows.

$$L = \begin{cases} \arg\max_{l} \ A(l), & s > \epsilon \\ rand(K), & s < \epsilon \end{cases}, \ with \ s \sim U(0, 1) \tag{3}$$

The parameter ϵ decides the level of greediness (or randomness). With the ϵ-greedy method, even if the true positive label is not selected in an early round, it still has a chance to be chosen in the following rounds. We set $\epsilon = 0.3$ in this paper and will discuss its effect later in the section. $A(l)$ is an objective function for the target label l of an arm. It is supposed to approximate the likelihood of the label being the true label target. We leverage two kinds of information in the approximation: *the current trigger size for the label* and *the trigger size variation for the label over rounds of optimization*. To simplify discussion, we leave symmetry (to distinguish natural features and triggers) to a later section. Intuitively, a label with a smaller trigger size is promising, and a label that continuously achieves good trigger size reduction in the past is promising. Let $tm(l)$ be the accumulated time spent on optimizing l (in the past rounds); $M(l)$ the current mask of l such that $\|M(l)\|_1$, the L_1 norm of $M(l)$, describes the trigger size; and $M_1(l)$ the first valid trigger for l. The objective function $A(l)$ is hence

defined as follows.

$$A(l) = \frac{\|M(l)\|_1 - \|M_1(l)\|_1}{tm(l)} + \beta \cdot \frac{1}{\|M(l)\|_1} \quad (4)$$

Here β is a hyper-parameter set to 10^5. In the early rounds, the trigger size reduction rate (i.e., the first term in the above equation) is a stronger indicator of true positive. The equation allows us to put more weight on the reduction rate instead of the trigger size, which tends to be large at the beginning and hence the second term tends to be small. As the optimization proceeds, the trigger size reduction rate degrades, even for the true positive label, the second term becomes dominating, allowing the scheduler to prioritize labels with small triggers (to make them smaller).

In the end, we compare the size of the smallest trigger with a threshold τ to decide whether a model is trojaned or benign. In this paper, we set $\tau = 300$ for all TrojAI models and $\tau = 350$ for ImageNet models.

Theoretical Analysis of K-Arm. We conduct theoretical analysis to show that K-Arm is more effective (i.e., having higher accuracy) and more efficient (i.e., lower overhead) than NC and NC+pre-selection. The effectiveness is proved by computing the expected time of finishing trigger generation for the true target label. Details can be found in Appendix.A.

4.3. Arm Pre-screening

According to the theoretical analysis, when the number of arms K is large, the cost is dominated by the warm-up phase that is determined by K. A large K is hence undesirable. Recall that for a model with N classes, $K = N + N \times (N-1)$, which could be large. We hence propose a pre-screening step to filter out arms that are not promising.

In order to achieve high attack success rate, the attacker often has to stamp many benign samples (of various classes when injecting a universal backdoor) with the trigger and use them in trojan training. Note that these stamped samples have their labels set to the target label. As such, the model learns the correlations between the target label and the benign features belonging to the original labels. Consequently, the logits value of the target label tends to be *consistently* larger than other labels for benign samples. We leverage this to preclude labels that do not look promising.

Specifically, for universal backdoor scanning, we consider a label promising if its logits value ranks among the top $\gamma\%$ labels in at least $\theta\%$ of all the benign samples (of various labels) that can be leveraged for scanning. Collecting such statistics has much lower cost compared to optimization. We set $\gamma = 25$ and $\theta = 65$ in this paper. For label-specific backdoor scanning, we consider an optimization arm from the victim label t_s to the target label t_d promising if t_d's logits value ranks among the top $\gamma\%$ labels in at least $\theta\%$ of all the available benign samples of label t_s. We set $\gamma = 25$ and $\theta = 90$ in this paper. Observe that our settings of γ and θ are conservative in order not to exclude the right one. We also empirically study the effect of different settings.

According to our experiments in the next section, the pre-screening can substantially reduce the number of arms to consider. For example, we can effectively reduce the arms of ImageNet from 1000 to 20 without sacrificing accuracy in universal backdoor scanning.

4.4. Symmetric Optimization to Distinguish Natural Features from Triggers

Assume a (small) trigger T is generated to flip clean samples with label t_s to label t_d. As discussed in Section 3, If T does not denote a backdoor but rather natural features, the two classes are likely close to each other. As such, the trigger flipping samples of t_d to t_s shall have a similar size as T. If T indeed denotes a backdoor, the trigger flipping t_d to t_s tends to be much larger as it is difficult to cause misclassification along the opposite direction of trojaning. Therefore, the scheduling algorithm is enhanced as follows to consider symmetry. The extension focuses on label-specific optimization as such confusion rarely happens for universal backdoors.

Given a label-specific arm $\langle t_s, t_d \rangle$, i.e., flipping t_s to t_d, $M(t_s, t_d)$ and $P(t_s, t_d)$ denote the mask and pattern for the generated trigger, respectively, and $M(t_d, t_s)$ and $P(t_d, t_s)$ the correspondence along the opposite direction (i.e., flipping t_d to t_s). The objective function is as follows.

$$A(t_s, t_d) = \frac{(\|M(t_s, t_d)\|_1 - \|M_1(t_s, t_d)\|_1)/tm(t_s, t_d) + \beta \cdot 1/\|M(t_s, t_d)\|_1}{(\|M(t_d, t_s)\|_1 - \|M_1(t_d, t_s)\|_1)/tm(t_d, t_s) + \beta \cdot 1/\|M(t_d, t_s)\|_1} \quad (5)$$

Intuitively, we leverage the ratio of objective functions in Equation (4) in the two directions to estimate the likelihood of $\langle t_s, t_d \rangle$ being the true victim-target label pair. When $A(t_s, t_d)$ is large, meaning the two directions are asymmetric, the pair is likely the true victim-target pair and selected.

5. Experiments

We compare our method with five state-of-the-art techniques against three different attack methods on multiple datsets and show that K-arm optimization can achieve better accuracy with lower time cost.

5.1. Datasets

TrojAI Competition. TrojAI (IARPA, 2020) is a program by IARPA that aims to tackle the back-door detection problem. In each round of competition, the performers are first given a large set of training models (over 1000) with different structures and different classification tasks. Roughly half of them are trojaned and their malicious identities are

known. A (small) set of benign examples are provided for each label of each model. These models may be trojaned with various kinds of backdoors, including universal and label-specific. The triggers could be pixel patterns (e.g., polygons with solid color) and Instagram filters (Liu et al., 2019). They could be position dependent or independent. Position dependency means that the trigger has to be at a specific relative position with the foreground object in order to cause misclassification. A model may have one or more backdoors. The complexity of models and backdoors grows from round to round. *Note that our technique does not require training. We hence use these training sets as regular datasets.* IARPA also hosts a test set online that is drawn from the same distribution as the training models. It is unknown which test models are trojaned. One can submit his/her solution which will be evaluated remotely on their server. The solution needs to finish scanning all the test models (100, 144, 288, and 288 for rounds 1-4, respectively) within 24 hours for rounds 1-2 and 48 hours for rounds 3-4. By the time of submission, round 4 is the latest. We compare our method with the baselines on all the models with polygon backdoors, mixed with all the clean models across all four rounds. We exclude models trojaned with Instagram filters as some baselines do not support them. The leaderboard results for our technique including both polygon and filter backdoors will be discussed in Section 5.5. The details of datasets can be found in Appendix.B.

ImageNet. We also use 7 VGG16 models on ImageNet (1000 classes) trojaned by TrojNN (Liu et al., 2018b), a kind of unviersal patch attack, and 6 models on ImageNet poisoned by hidden-trigger backdoors (Saha et al., 2020), with different structures including VGG16, AlexNet, DenseNet, Inception, ResNet and SqueezeNet. The hidden-trigger backdoors are label-specific. They are mixed with 7 clean ImageNet models.

Other datasets. We also evaluate our method on 4 CIFAR10 and 4 GTSRB models trojaned by Input-Aware Dynamic Attack (Nguyen & Tran, 2020). They are mixed with 4 clean models respectively.

5.2. Evaluation Metrics

We report two accuracy metrics used in TrojAI: *cross-entropy loss* (Murphy, 2012) and *ROC-AUC* (Area under Receiver Operating Characteristic Curve) (Fawcett, 2006). The former is the lower the better and the latter is the higher the better. In addition, we also report the plain accuracy, i.e., the percentage of models that are correctly classified. We also report the average scanning time for each model. For fair comparison, comparative experiments are all done on an identical machine with a single 24GB memory NVIDIA Quadro RTX 6000 GPU (with the lab server configuration). Leaderboard results (on TrojAI test sets) were run on the IARPA server with a single 32GB memory NVIDIA V100 GPU. We use Adam (Kingma & Ba, 2014) optimizer with learning rate 0.1, $\beta = \{0.5, 0.9\}$ for all the experiments.

5.3. Baseline Methods

We compare K-Arm with the following state-of-the-art detection methods: ABS (Liu et al., 2019), NC (Wang et al., 2019), NC+pre-selection (Wang et al., 2019) (or Pre-selection for short), ULP (Kolouri et al., 2020), TABOR (Guo et al., 2019), DLTND (Wang et al., 2020c). For the optimization based methods including ABS, NC, Pre-selection, TABOR and DLTND, we use the same batch size for fair comparison. For NC, Pre-selection and our method, we use the same early stop condition to terminate the optimization. For ABS, we select top10 neuron candidates after the stimulation analysis and perform the trigger reverse engineering. For Pre-selection, we set the number of optimization epochs as $max(10, s)$ for each label with s the number of epochs when the first valid trigger is found. Recall Pre-selection performs a few rounds of optimizations and then selects a promising subset to finish. We select the top 3 among the 5 labels for round 1 models and the top 20% labels/label-pairs for rounds 2-4. For the ImageNet models, we follow (Wang et al., 2019) and select the top 100. For ULP, we train it on 500 TrojAI round 1 models and test it on the 100 test models. We did not run it on later rounds as it cannot handle model structure variations in those rounds. For TABOR and DLTND, we use the implementation provided by the authors.

5.4. Parameter Tuning

We evaluate the effects of hyper-parameters, including the following: β in the objective function, θ, γ in the arm pre-screening and ϵ in the K-Arm Scheduler. The last one is the threshold τ which decides if a model is trojaned. We randomly select 40 models (20 benign and 20 trojaned) from round 2 to test our method. In detail, we pick 5 different values $(10^2, 10^3, 10^4, 10^5, 10^6)$ for β. For ϵ, we select 10 values ranging from $0.1 \sim 0.5$. We use 5 different τ values from $100 \sim 500$, 3 θ values from $10 \sim 30$ and 3 γ values from $50 \sim 80$. The results are in Appendix.C.

5.5. Experimental results

Results for TrojAI Rounds 1-4 Training Sets. Table 1 shows the comparison results on the aforementioned models from TrojAI rounds 1-4 training sets (3231 models in total). Columns Acc, Loss, ROC, and Time stand for plain accurcy, cross entropy loss, AUC-ROC, and average scanning time (in seconds) per model, respectively. Observe that our method achieves the best accuracy and has the lowest scanning time compared to all the baselines. The best K-Arm methods have 4%, 27%, 30%, 25% better ROC than

Table 1. TrojAI Training Set Results; "Sym K-Arm Opt + Pre-Srn" stands for symmetric K-Arm with pre-screening.

	Round1				Round2				Round3				Round4			
Method	Acc	Loss	ROC	Time(s)	Acc	Loss	ROC	Time(s)	Acc	Loss	ROC	Time(s)	Acc	Loss	ROC	Time(s)
NC	72%	0.61	0.73	623.9	-	-	-	> 30000	-	-	-	> 30000	-	-	-	> 30000
Pre-selection	71%	0.62	0.72	507.5	51%	1.16	0.54	3708.2	58%	0.81	0.61	3482.5	55%	1.09	0.55	3210.4
ABS	67%	0.67	0.70	542.9	62%	0.76	0.57	1527.0	71%	0.62	0.56	1435.0	79%	0.52	0.55	525.0
TABOR	80%	0.51	0.81	1142.2	55%	1.09	0.59	> 32000	60%	0.77	0.57	> 30000	60%	0.81	0.55	> 35000
DLTND	85%	0.45	0.86	1109.6	60%	0.79	0.62	> 26000	65%	0.75	0.61	> 29000	65%	0.77	0.64	> 31000
K-Arm Opt	**90%**	**0.32**	**0.90**	**275.5**	**76%**	**0.58**	**0.77**	**1956.5**	**79%**	**0.50**	**0.80**	**1740.3**	**82%**	**0.51**	**0.81**	**1623.5**
K-Arm Opt + Pre-Srn	**-**	**-**	**-**	**-**	**75%**	**0.59**	**0.76**	**140.8**	**79%**	**0.50**	**0.80**	**166.2**	**80%**	**0.53**	**0.79**	**110.5**
Sym K-Arm Opt + Pre-Srn	**-**	**-**	**-**	**-**	**89%**	**0.33**	**0.89**	**340.5**	**91%**	**0.31**	**0.91**	**290.5**	**89%**	**0.32**	**0.89**	**204.4**

Table 2. Results on ImageNet Models

	Hidden Trigger Attack				TrojanNN			
Method	Acc	Loss	ROC	Time(s)	Acc	Loss	ROC	Time(s)
NC	-	-	-	>1m	71%	0.65	0.82	221k
Pre-selection	54%	1.02	0.62	171k	64%	0.92	0.74	43k
ABS	100%	0.11	1.00	389k	100%	0.11	1.00	4.9k
K-Arm	**85%**	**0.33**	**0.93**	**86k**	**88%**	**0.38**	**0.92**	**19k**
K-Arm+Pre-Srn	**85%**	**0.33**	**0.93**	**2k**	**100%**	**0.11**	**1.00**	**224**
Sym K-Arm+Pre-Srn	**100%**	**0.09**	**1.00**	**4k**	**-**	**-**	**-**	**-**

the best performance by the baselines for the four respective rounds. They are also 1.8, 10.8, 8.6, 4.8 times faster than the fastest among the baselines for the four respective rounds. This strongly supports the better effectiveness and efficiency of K-Arm. K-Arm has higher accuracy than Pre-selection and ABS because they have to make deterministic selection (about which labels/neurons to optimize) at the beginning which is difficult when the candidate sets are large (e.g., in label-specific backdoor scanning). K-Arm has higher accuracy than NC even though NC is exhaustive. Besides that NC does not consider symmetry and hence cannot distinguish natural features from injected triggers, its exhaustive nature in many cases also hurts performance as it aggressively optimizes for clean labels, generating many natural features with small size that behave like triggers. TABOR and DLTND encounter the same problem and suffer from huge number of false alarms.

The last three rows in Table 1 and 2 present the ablation study for different components of our method. The vanilla K-Arm can have 79% accuracy and 1773s on average (from R2 to R4). K-Arm with pre-screening achieves 78% accuracy and 138s. Symmetric K-Arm with pre-screening gets 89% and 278s. Note that vanilla NC only has 57% with 32000s. Observe that arm pre-screening substantially reduces the scanning time (by an order of magnitude) without sacrificing much accuracy; symmetric optimization is critical to improving accuracy, with 13%, 11%, and 10% ROC improvement for rounds 2-4. Without the symmetric optimization, K-Arm would not be able to reach the round targets (i.e., lower than 0.348 Loss).

Results for ImageNet Models. Table 2 shows the results for the ImageNet models. Columns 2-5 present results on the 6 models with (label-specific) hidden-trigger backdoors mixed with 7 benign models; columns 6-9 present results on the 7 models with (universal) TrojNN backdoors, mixed with 7 benign models. For hidden-trigger backdoors, the best K-Arm has 100% accuracy. NC could not finish due to the large number of victim-target label pairs. It took two weeks to scan a model. Both Pre-selection and ABS have much worse accuracy or scanning time. For TrojNN backdoors, The best K-Arm has 100% accuracy, higher than most baselines. Although ABS can also achieve 100% accuracy, it is 20 times slower than the best K-Arm. NC and Pre-selection have lower accuracy and much longer scanning time due to the large number of classes and natural features that behave like triggers.

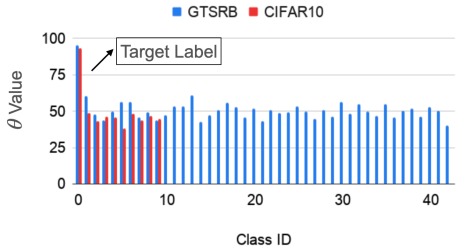


Figure 5. Results on Input-Aware Dynamic Attack.

Results for the Dynamic Attack. Different from static backdoor attacks, dynamic attack can generate input specific triggers. Therefore, the optimized trigger of the target class will not be extremely smaller than others, then bypass the outlier detection. However, our experiment results show that the proposed pre-screening technique can identify the target label preciously for the poisoned models. By setting the bound $\theta = 70$, $\gamma = 25$, we can successfully detect all 4 trojan models on CIFAR10 and GTSRB without any false positives. Fig. 5 shows the θ values of different classes for a GTSRB and a CIFAR10 poisoned models. The target label is 0 for both models. Observe that the θ value of the target label is 35% larger than the largest value of the rest labels, which is a strong indicator for the trojan models. Remind the large θ reveals that the model learns the target class features as part of the features for other classes due to the poisoning process.

K-Arm Performance on TrojAI Leaderboard. K-Arm consistently achieved top results across the four rounds[1][2].

[1] https://pages.nist.gov/trojai/

[2] https://pages.nist.gov/trojai/docs/results.html#previous-leaderboards

Table 3. TrojAI Leaderboard Results

	Round1				Round2				Round3				Round4			
Method	CE Loss	ROC	Time(s)	Rank	CE Loss	ROC	Time(s)	Rank	CE Loss	ROC	Time(s)	Rank	CE Loss	ROC	Time(s)	Rank
NC	-	-	T/O	-	-	-	T/O	-	-	-	T/O	-	-	-	T/O	-
ABS	0.64(+0.34)	0.70(-0.21)	523(+233)	-	0.76(+0.44)	0.53(-0.36)	508(+18)	-	0.84(+0.55)	0.56(-0.35)	599(+367)	-	0.87(+0.55)	0.48(-0.42)	229(+18)	-
ULP	1.18(+0.88)	0.59(-0.32)	0.1(-290)	-	-	-	-	-	-	-	-	-	-	-	-	-
K-Arm	**0.30(-0.00)**	**0.91(-0.00)**	**290(-0)**	**1**	**0.35(+0.03)**	**0.90(+0.01)**	**290(-200)**	**2**	**0.29(-0.00)**	**0.91(-0.00)**	**232(-0)**	**1**	**0.33(+0.01)**	**0.90(-0.00)**	**201(-10)**	**2**

Table 3 shows the K-Arm results for the four rounds, including the loss, ROC, average scanning time, and ranking. The results include those for all the different types of backdoors (polygon, filter, label-specific, universal, position-dependent, multiple backdoors in a model, etc.). We also show the difference between K-Arm and the top (if any). For example, in round 2, K-Arm ranked number 2. Loss 0.35(+0.03) means that K-Arm's loss is 0.35 while the top performer has 0.32 loss; ROC 0.90(+0.01) means that K-Arm has 0.9 ROC while the top performer has 0.89 ROC. Note that the leaderboard ranks solutions by (smaller) loss. K-Arm beat the round targets (i.e., lower than 0.348 loss) for 3 out of the 4 rounds. For round 2, although it did not beat the target, its ROC is the highest. It ranked number one for 2 out of the 4 rounds. In all rounds, K-Arm is faster than ABS. We also train ULP on 500 round 1 training set models and evaluate it on the round 1 test set. However, its accuracy is not high. We speculate two reasons: 1) unlike the models in the ULP paper, the classes of TrojAI models are not fixed; 2) the classifier seems to easily overfit on the training data and the triggers in the TrojAI datasets share few common features. On the other hand, ULP is not optimization based and hence is extremely fast.

Trend of Trigger Optimization in K-Arm. We randomly sample 100 trojaned models from each training set of TrojAI rounds 1 to 4. We record the ranking of the optimized trigger size of true target label for each model during optimization. Fig. 6 shows the percentage of models whose target label trigger size ranks number 1 (i.e., the smallest) for each round. We can see that after warm-up, there are only 60-70% models rank top. As such, a simple pre-selection strategy does not work. All the sets converge at around 90%, indicating that K-Arm allows the true positives to stand out eventually in most cases. Also observe that the different sets converge at different optimization rounds, indicating that using a universal larger number of warm-up rounds instead of K-Arm will not work. Moreover, 20 rounds of warm-up means hundreds of epochs, which is already not affordable as all arms have to go through warm-up. At the end, we point out that there are still around 10% cases that do not stand out at the end. We study some of them in the Appendix.D. We leave the problem to future work.

Adaptive Attack. We devise an adaptive attack for the arm pre-screening stage. Our goal is to suppress the target label logits for benign samples of victim classes. This is done by adding an $\mathcal{L}_2$ regularization of target label logits value of benign samples. As such, the optimizer tries to enlarge

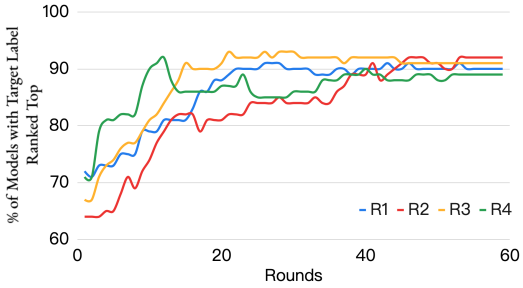


Figure 6. Trend of Trigger Optimization.

Table 4. Adaptive Attack

Coefficient	0	1	10	100	1000
Model Acc	80.0%	78.1%	71.4%	65.5%	35.4%
ASR	99.0%	99.4%	92.9%	92.6%	0.0%
Selection Acc	100.0%	100.0%	80.0%	50.0%	-
K-Arm Detection Acc	100.0%	100.0%	70.0%	40.0%	-

the distance between the target label and the victim label. The strength of the attack is controlled by a coefficient. We use 10 models on CIFAR10 with different coefficient values and report the model accuracy, attack success rate (ASR), selection accuracy and K-Arm detection accuracy in Table 4. Observe that pre-screening becomes less effective when the attack is stronger. However, the model accuracy and attack success rate degrade as well. It is unclear how to design adaptive attack for the scheduler or optimizer. We will leave it to future work.

6. Conclusion

Inspired by K-Arm Bandit in Reinforcement Learning, we develop a K-Arm optimization technique for back-door scanning. The technique handles the inherent uncertainty in searching a very large space of model behaviors, using stochastic search guided by an objective function. It shows outstanding performance on models from IARPA TrojAI competitions. It also outperforms the state-of-the-art techniques that are publicly available.

7. Acknowledgments

We thank the anonymous reviewers, Zikang Xiong for their valuable comments. This research was supported, in part by IARPA TrojAI W911NF-19-S-0012, NSF 1901242 and 1910300, ONR N000141712045, N000141410468 and N000141712947. Any opinions, findings, and conclusions in this paper are those of the authors only and do not necessarily reflect the views of our sponsors.

References

Auer, P. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 2938–2948. PMLR, 2020.

Chan, A. and Ong, Y.-S. Poison as a cure: Detecting & neutralizing variable-sized backdoor attacks in deep neural networks. *arXiv preprint arXiv:1911.08040*, 2019.

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

Chen, X., Salem, A., Backes, M., Ma, S., and Zhang, Y. Badnl: Backdoor attacks against nlp models. *arXiv preprint arXiv:2006.01043*, 2020.

Cheng, S., Liu, Y., Ma, S., and Zhang, X. Deep feature space trojan attack of neural networks by controlled detoxification. *arXiv preprint arXiv:2012.11212*, 2020.

Chou, E., Tramèr, F., and Pellegrino, G. Sentinet: Detecting localized universal attacks against deep learning systems. In *2020 IEEE Security and Privacy Workshops (SPW)*, pp. 48–54. IEEE, 2020.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

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

Eggenberger, F. and Pólya, G. Über die statistik verketteter vorgänge. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 3(4):279–289, 1923.

Fawcett, T. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

Fritsch, J., Kuehnl, T., and Geiger, A. A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2013.

Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D. C., and Nepal, S. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pp. 113–125, 2019.

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

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

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Huang, S., Peng, W., Jia, Z., and Tu, Z. One-pixel signature: Characterizing cnn models for backdoor detection. *arXiv preprint arXiv:2008.07711*, 2020.

IARPA. Trojai competition. https://pages.nist.gov/trojai/, 2020.

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

Kolouri, S., Saha, A., Pirsiavash, H., and Hoffmann, H. Universal litmus patterns: Revealing backdoor attacks in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 301–310, 2020.

Larsson, F. and Felsberg, M. Using fourier descriptors and spatial models for traffic sign recognition. In *Scandinavian conference on image analysis*, pp. 238–249. Springer, 2011.

Li, S., Ma, S., Xue, M., and Zhao, B. Z. H. Deep learning backdoors. *arXiv preprint arXiv:2007.08273*, 2020a.

Li, Y., Wu, B., Jiang, Y., Li, Z., and Xia, S.-T. Backdoor learning: A survey. *arXiv preprint arXiv:2007.08745*, 2020b.

Lin, J., Xu, L., Liu, Y., and Zhang, X. 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*, pp. 113–131, 2020.

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

Liu, Y., Xie, Y., and Srivastava, A. Neural trojans. In *2017 IEEE International Conference on Computer Design (ICCD)*, pp. 45–48. IEEE, 2017.

Liu, Y., Ma, S., Aafer, Y., Lee, W.-C., Zhai, J., Wang, W., and Zhang, X. Trojaning Attack on Neural Networks. In *Proceedings of the 25nd Annual Network and Distributed System Security Symposium (NDSS)*, 2018b.

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

Ma, S. and Liu, Y. Nic: Detecting adversarial samples with neural network invariant checking. In *Proceedings of the 26th Network and Distributed System Security Symposium (NDSS 2019)*, 2019.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Murphy, K. P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Nguyen, A. and Tran, A. Input-aware dynamic backdoor attack. 2020.

Qiao, X., Yang, Y., and Li, H. Defending neural backdoors via generative distribution modeling. In *Advances in Neural Information Processing Systems*, pp. 14004–14013, 2019.

Rakin, A. S., He, Z., and Fan, D. Bit-flip attack: Crushing neural network with progressive bit search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Rakin, A. S., He, Z., and Fan, D. Tbt: Targeted neural network attack with bit trojan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13198–13207, 2020.

Rezaei, S. and Liu, X. A target-agnostic attack on deep models: Exploiting security vulnerabilities of transfer learning. *arXiv preprint arXiv:1904.04334*, 2019.

Saha, A., Subramanya, A., and Pirsiavash, H. Hidden trigger backdoor attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 11957–11965, 2020.

Salem, A., Wen, R., Backes, M., Ma, S., and Zhang, Y. Dynamic backdoor attacks against machine learning models. *arXiv preprint arXiv:2003.03675*, 2020.

Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, pp. 6103–6113, 2018.

Sun, L. Natural backdoor attack on text data. *arXiv preprint arXiv:2006.16176*, 2020.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Turner, A., Tsipras, D., and Madry, A. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019.

Wang, B., Yao, Y., Viswanath, B., Zheng, H., and Zhao, B. Y. With great training comes great vulnerability: Practical attacks against transfer learning. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pp. 1281–1297, 2018.

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

Wang, B., Cao, X., Gong, N. Z., et al. On certifying robustness against backdoor attacks via randomized smoothing. *arXiv preprint arXiv:2002.11750*, 2020a.

Wang, H., Sreenivasan, K., Rajput, S., Vishwakarma, H., Agarwal, S., Sohn, J.-y., Lee, K., and Papailiopoulos, D. Attack of the tails: Yes, you really can backdoor federated learning. *Advances in Neural Information Processing Systems*, 33, 2020b.

Wang, R., Zhang, G., Liu, S., Chen, P.-Y., Xiong, J., and Wang, M. Practical detection of trojan neural networks: Data-limited and data-free cases. *arXiv preprint arXiv:2007.15802*, 2020c.

Watkins, C. J. C. H. Learning from delayed rewards. 1989.

Wong, E., Rice, L., and Kolter, J. Z. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

Xie, C., Huang, K., Chen, P.-Y., and Li, B. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2019.

Xu, X., Wang, Q., Li, H., Borisov, N., Gunter, C. A., and Li, B. Detecting ai trojans using meta neural analysis. *arXiv preprint arXiv:1910.03137*, 2019.

Yao, Y., Li, H., Zheng, H., and Zhao, B. Y. Latent backdoor attacks on deep neural networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2041–2055, 2019.

Zhang, X., Mian, A., Gupta, R., Rahnavard, N., and Shah, M. Cassandra: Detecting trojaned networks from adversarial perturbations. *arXiv preprint arXiv:2007.14433*, 2020.

Zhao, S., Ma, X., Zheng, X., Bailey, J., Chen, J., and Jiang, Y.-G. Clean-label backdoor attacks on video recognition models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14443–14452, 2020.

Zhu, C., Huang, W. R., Shafahi, A., Li, H., Taylor, G., Studer, C., and Goldstein, T. Transferable clean-label poisoning attacks on deep neural nets. *arXiv preprint arXiv:1905.05897*, 2019.

A. Theoretical Analysis

Let the time needed for a single round optimization is t. For simplicity, we further assume the number of rounds of optimization needed to generate the final trigger for the target label is R and the objective function has p probability choosing the target label. Let $p_s = (1-\epsilon)\cdot p + \epsilon \cdot \frac{1}{K}$ be the probability the target label is scheduled.

Efficiency. Since the selective optimization terminates only when the target label is optimized $R-2$ times, it follows the *Negative Binomial Distribution* (Eggenberger & Pólya, 1923) that models the probability of the number of failure events before a given number of successful events happen, when the probability of one successful event is given. The expected time cost of K-Arm is hence the following.

$$\mathbb{E}[T_{km}] = 2\cdot K\cdot t + \frac{(R-2)\cdot t}{(1-\epsilon)\cdot p + \epsilon\cdot\frac{1}{K}} \tag{6}$$

The first term is the time for the warm-up phase in which all the K labels go through 2 rounds of optimization. The second is the time for the selective optimization. The denominator is the probability of choosing the right label. From the equation, We have the following observations.

- When $R \gg K$, such as for TrojAI round 1 models (i.e., $R = 50$ and $K = 5$). The cost is dominated by the second term. Therefore, we have $\mathbb{E}[T_{km}] = \mathcal{O}(R\cdot t)$. Since the cost for NC is $\mathbb{E}[T_{nc}] = \mathcal{O}(K\cdot R\cdot t)$, the speed-up over NC is determined by K.
- When $R \ll K$, e.g., in ImageNet models with $K = 1000$. The cost is dominated by the first term. $\mathbb{E}[T_{km}] = \mathcal{O}(K\cdot t)$ and the speed-up is determined by R.

Effectiveness. We analyze the effectiveness of our method by comparing with NC and NC+pre-selection the likelihood of finishing optimizing the target label within a time bound. The analysis is done by comparing the expected time of finishing optimizing the target label. Note that if the time bound is fixed, the smaller expected value means a higher probability of finishing successfully.

NC vs. K-Arm. Since NC optimizes all labels in order, the expected finishing time is the following.

$$\mathbb{E}[T_{nc}] = R\cdot t\cdot(1\cdot\frac{1}{K} + 2\cdot\frac{1}{K} + \ldots K\cdot\frac{1}{K}) = \frac{(K+1)\cdot R\cdot t}{2}$$

In practice, due to the objective function design, the probability of K-Arm scheduling the target label $p_s = (1-\epsilon)\cdot p + \epsilon\cdot\frac{1}{K}$ is usually much higher than $2/K$ when K is not small. Together with Eq. (6), we have $\mathbb{E}[T_{km}] < 2\cdot K\cdot t + \frac{(R-2)\cdot t}{\frac{2}{K}} = \frac{K\cdot R\cdot t}{2} + K\cdot t < \mathbb{E}[T_{nc}]$. Note R is usually larger than $2\cdot K$.

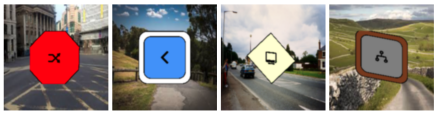


Figure 7. Example images from TrojAI datasets

NC+pre-selection vs. K-Arm. NC+pre-selection makes deterministic decision to select the m smallest triggers after the initial optimization. If the target label is not among the m smallest, pre-selection will never succeed. In practice, the failure probability is not low. Here, we only focus on comparing K-Arm with NC+pre-selection when the target label is among the m smallest. We have the expected time of pre-selection $\mathbb{E}[T_{ps}] = 2\cdot K\cdot t + \frac{(m+1)(R-2)t}{2}$, similar to $\mathbb{E}[T_{nc}]$. When $p_s > \frac{2}{m}$ (which holds in practice), following the reasoning similar to above, we have $\mathbb{E}[T_{km}] < \mathbb{E}[T_{ps}]$.

B. Details of TrojAI Competition Datasets

Round1 Dataset. The round1 training set contains 1000 CNN models for classification tasks, in which 532 models are trojaned and 468 are benign. Each model has 5 labels and IARPA provides 100 labeled clean images with size 224x224x3 for each class. A clean image is generated by combining a foreground object and a background image. The foreground objects are traffic signs with different shapes. The background images are road scene data drawn from KITTI (Fritsch et al., 2013), Cityscapes (Cordts et al., 2016) and Swedish Roads (Larsson & Felsberg, 2011). Note that these samples were not used to train the models, but drawn from the same distribution. Sample images are shown in Fig. 7. There are 3 different model architectures for round1 models: ResNet-50 (He et al., 2016), Inception-v3 (Szegedy et al., 2016), DenseNet-121 (Huang et al., 2017). There are only universal triggers in the round1 trojaned models. The triggers are polygons with 3 to 12 sides and a randomly selected color. In each malicious image, a trigger is stamped on an unknown area inside the foreground object. The size of trigger varies from $2 \sim 24\%$ of the foreground object. Fig 8 illustrates the generation process of trojan images.

Round2 Dataset. The round2 training set contains 1104 CNN models for classification tasks, with 552 trojaned models and 552 benign models. Compared to round1, round2 models have more labels ranging from $5 \sim 25$. The clean images provided for each label are fewer (20 per class). It includes universal triggers, label specific triggers, and also Instagram filter triggers. There are 23 different model architectures. More description related to the TrojAI datasets can be found in (IARPA, 2020).

Round3 Dataset. The round3 training set contains 1008 CNN models for image classification tasks with 504 trojaned models and 504 benign models. Same as round2, the

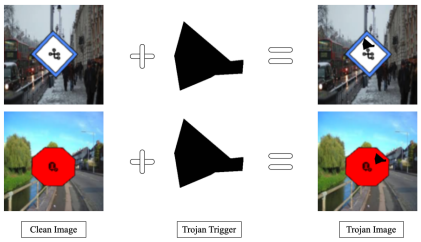
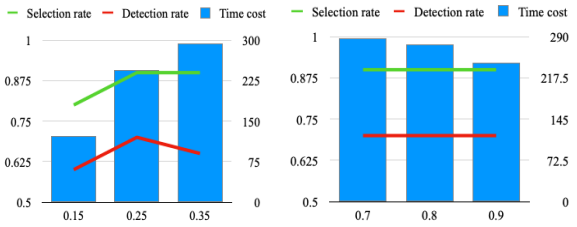


Figure 8. Trojan Image Generation

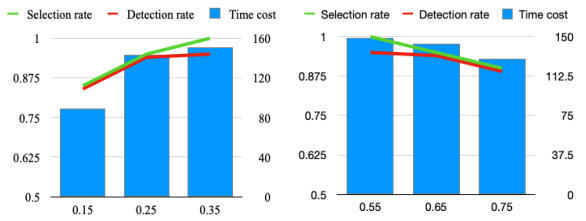


(a) γ Comparison (b) θ Comparison

Figure 9. Label-specific trigger detection under different hyper-parameters.

number of classes for each model is $5 \sim 25$, and the clean images provided for each label are $10 \sim 20$. Different from round2 models, all round3 models are enhanced through adversarial training (Madry et al., 2017; Wong et al., 2020). The adversarial attack has 3 different strength levels based on the perturbation size ($\frac{4}{255}, \frac{8}{255}, \frac{16}{255}$) and 2 different levels based on the ratio $(0.1, 0.3)$, i.e. what percentage of the batches are attacked. The number of iterations used in PGD attacks is set as 4 different values $(2, 4, 8, 16)$. More details can be found in (IARPA, 2020).

Round4 Dataset. The round4 training set contains 1008 CNN models with 504 trojaned models and 504 benign models. As the most challenging round, round4 models have more classes ($15 \sim 44$), less samples ($2 \sim 5$ per class). Unlike previous rounds, round4 models can have many concurrent conditional triggers. Such triggers can cause the misclassification only when they fulfill the conditions. There are three different conditions: spatial, spectral and class. The spatial trigger requires the trigger exists within a certain area to cause the misclassification behaviour. The spectral trigger can only lead the misclassification when the trigger has certain color. The class context requires the trigger must be stamped on the correct class. Besides, the universal triggers are removed in round4. There are only label specific triggers. Such comprehensive settings make the backdoor detection more difficult. Table 5 summarizes the configurations cross all trojAI 4 rounds.



(a) γ Comparison (b) θ Comparison

Figure 10. Universal trigger detection under different hyper-parameters.

C. Impact of Hyper-parameters

From Fig. 11a, we observe that K-Arm has stable detection accuracy and time cost in a large range of β (from 10^2 to 10^6). When β is small, K-Arm might get stuck with a few labels that seem promising (based on the objective function). Thus the time cost slightly increases. From Fig. 11b, when ϵ is large, K-Arm pays more attention to exploring random labels, which leads to more time consumption. From Fig. 11c , when τ is small, many real (back-door) triggers are considered benign, causing accuracy degradation. When τ is in 300-500, we can achieve a stable high accuracy (around 91%) to distinguish the trojaned and benign models.

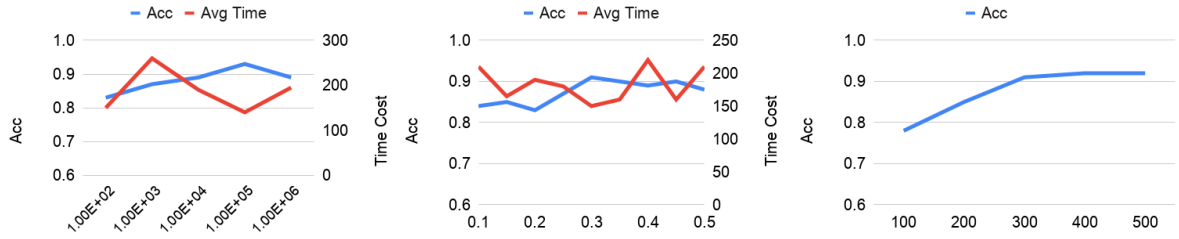
We evaluate the effect of remaining two hyper-parameters θ and γ. Recall that θ and γ are used in the arm pre-processing phase. In particular, we consider a label promising if its logits value ranks among the top $\gamma\%$ labels in at least $\theta\%$ of all benign samples of a label (for label-specific trigger scanning) or various labels (for universal trigger scanning). Intuitively, γ should be small and θ should be large. For scanning universal triggers, we set 3 different values for γ $(15, 25, 35)$ and 3 different values for θ $(55, 65, 75)$. For scanning label specific triggers, we test the same values of γ and choose θ from $(70, 80, 90)$. Given 20 randomly selected round2 models with global triggers and 20 with label specific triggers, we report the accuracy for selecting the correct target label successfully under different settings, the average time cost and the detection accuracy. From Fig. 9a and Fig. 10a, we can see that a small γ value causes some target labels omitted as the arm size is reduced. This further leads to detection accuracy degradation. On the other hand, when γ is large, although the selection rate increases, the time cost goes up. Compared to γ, arm pre-processing is less sensitive to θ. From Fig. 9b and Fig. 10b, the detection accuracy and time cost are more stable with different θ values.

D. Study of K-Arm Failing Cases

In this section, we study 2 K-Arm failing cases and explain the reasons.

Table 5. TrojAI Dataset

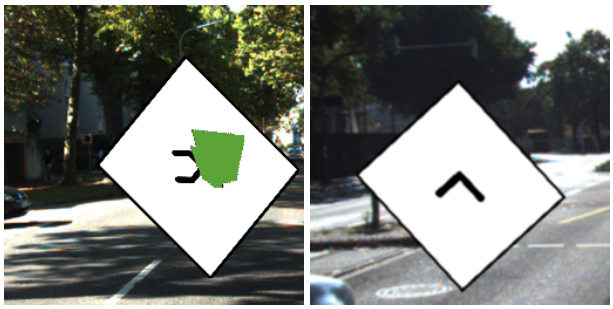
Rounds	# of Models	# of Classes	# of Samples per Class	# of Model Architectures	# of Triggers	Global Trigger	Label-specific Trigger	Polygon Trigger	Instagram Filter Trigger	Adv.Training
Round1	1000	5	100	3	1	✓	✗	✓	✗	✗
Round2	1104	5∼25	10∼20	23	1	✓	✓	✓	✓	✗
Round3	1008	5∼25	10∼20	23	1	✓	✓	✓	✓	✓
Round4	1008	15∼44	2∼5	16	1∼2	✗	✓	✓	✓	✓



(a) β Comparison (b) ϵ Comparison (c) τ Comparison

Figure 11. K-Arm accuracy and time cost under different parameter value settings

Case I: Pre-screening fails to select the correct target-victim pair. According to the Figure 9a, the pre-screening can not achieve 100% selection accuracy. Therefore, for some trojaned models, the correct victim-target pair is filtered out during the pre-selection stage and cause the detection fail. For instance, model #18 in round4 is a trojaned model with a label-specific polygon trigger. The victim label is 14 and target label is 8. When we apply the pre-screening by the default setting ($\gamma = 25, \theta = 90$) on this model, we find that 13 out of 342 pairs are selected. However, the right pair is not in the list. In fact, there are only 60% samples from the victim label, in which the target label's logits value rank on the top 25% among all labels. Since the right pair is pruned out, the following K-Arm optimization cannot find a trigger smaller than the threshold τ and report the model as benign.



(a) Victim + Trigger (b) Target

Figure 12. R2 model#22

Case II: Symmetric K-Arm fails when victim and target labels are similar. Recall that the Symmetric K-Arm performs the trigger optimization in two opposite directions and considers the ratio of objective functions to distinguish the real trigger and natural features. However, when the ground truth trigger is stamped on a victim class which is similar to the target label, Symmetric K-Arm will avoid selecting such a pair to optimize due to the small ratio, and eventually it causes the detection to fail. Figure 12 shows the victim label#13 image stamped with trigger and target label#12 image for model#22 in round2. As shown in the figure, the victim class is very similar to the target class. The sign at the center of the image is the only difference between the two classes. Fig. 13 further illustrates the trigger size variation in two opposite directions. We can see that the trigger sizes reduce in the same pace for both directions. Therefore, the ratio of objective functions is closed to 1. This pair can rarely be selected to optimize in K-Arm. In this case, K-Arm actually selects the victim-target pair (#5-#1) in most rounds, and eventually reports the model as benign since the optimized trigger is larger than τ.

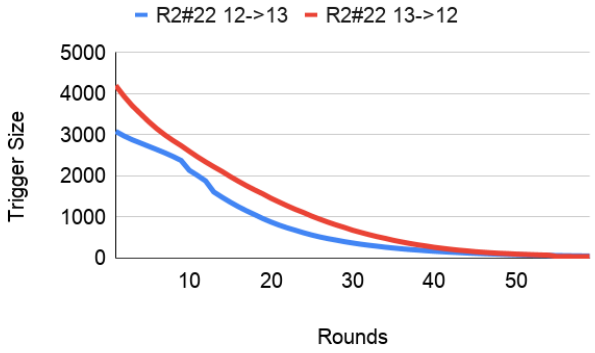


Figure 13. Trigger size variation in two opposition directions.