Practical Poisoning Attacks on Neural Networks

Junfeng Guo and Cong Liu

The University of Texas at Dallas
{Junfeng.Guo,cong}@utdallas.edu

Abstract Data poisoning attacks on machine learning models have attracted much recent attention, wherein poisoning samples are injected at the training phase to achieve adversarial goals at test time. Although existing poisoning techniques prove to be effective in various scenarios, they rely on certain assumptions on the adversary knowledge and capability to ensure efficacy, which may be unrealistic in practice. This paper presents a new, practical targeted poisoning attack method on neural networks in vision domain, namely BlackCard. BlackCard possesses a set of critical properties for ensuring attacking efficacy in practice, which has never been simultaneously achieved by any existing work, including ***knowledge-oblivious***, ***clean-label***, and ***clean-test***. Importantly, we show that the effectiveness of BlackCard can be intuitively guaranteed by a set of analytical reasoning and observations, through exploiting an essential characteristic of gradient-descent optimization which is pervasively adopted in DNN models. We evaluate the efficacy of BlackCard for generating targeted poisoning attacks via extensive experiments using various datasets and DNN models. Results show that BlackCard is effective with a rather high success rate while preserving all the claimed properties.

Keywords: Data Poisoning, Neural Networks

1 Introduction

While deep neural networks (DNNs) have the potential to revolutionize many important computer vision application domains such as face recognition [29] and autonomous driving [19], they may open up new adversarial opportunities due to lacking sufficient robustness against various forms of attacks.

Indeed, attacking neural nets has attracted much recent attention from both academia and industry. Most attacking techniques can be categorized into *evasion attacks* which occur at test phase and *data poisoning attacks* which occur at training phase. Specifically, evasion attacks [41] aim at modifying a clean target instance at test phase to spur misclassification or avoid detection by a classifier; while data poisoning attacks seek to insert maliciously crafted poison samples into the training set to manipulate the performance of a system. Data poisoning attacks are receiving a significantly increasing amount of attention recently [40], [32], [42], [20], [30], [25], [37], [31], [28], [22], due to the fact that such attacks are able to change and reconstruct internal parameters of the target model rather than just fooling the target model at test phase via modified test samples. Data poisoning can be typically categorized into untargeted attacks [42, 28, 23] and

Attack approach	Knowledge-oblivious	Clean-label	Clean-test
Trojaning Attack		✓	
BadNets			
Targeted BackDoor Attack	✓		
Poison Frog		✓	✓
BlackCard	✓	✓	✓

Table 1: Overall comparison.

targeted attacks [20, 37]. Untargeted attacks aim at degrading overall performance of the targeted model; while targeted attacks seek to control the behavior of a classifier on one specific test instance [20].

To guarantee effectiveness of poisoning attacks in practice, there are several critical properties that shall be possessed by poisoning techniques, including (*i*) ***knowledge-oblivious***–the attacker shall have no knowledge of the target model's parameters/structures, nor the original training datasets, (*ii*) ***clean-label***–the attacker shall not be able to control the labeling process, and (*iii*) ***clean-test***–test-time instances shall not be required to be modified using added adversarial perturbations for attacking effectiveness. Unfortunately, it is rather challenging to simultaneously achieve all these properties, as the latest poisoning techniques manage to achieve a partial set of these properties. For instance, [37] is able to first-time achieve clean-label attacks, yet they need knowledge about the targeted DNN model's parameters and structures to collide the feature space representations of the targeted instance. [20] does not require knowledge of the target model, yet requiring to control the labeling process to mislabel the targeted instance and inject it at the training phase. In fact, no existing method can simultaneously achieve the above-listed properties, which are essential to ensure any poisoning technique to be feasibly implemented under many practical scenarios.

The major contribution of this paper is towards implementing an effective and practical targeted poisoning attack BlackCard against neural nets, possessing all the above-mentioned properties. A detailed comparison of BlackCard against state-of-the-art targeted poisoning techniques is given in Table 1 (we will describe these related works in detail in Sec. 5). The efficacy of BlackCard is fundamentally supported by a set of analytical reasoning and observations exploiting an essential characteristic of gradient-descent optimization (detailed in Sec. 3). We have extensively evaluated BlackCard using a set of popular datasets and DNN models featuring very different structures and parameters in different tasks. The results prove the efficacy of BlackCard while achieving all properties above.

Our contributions are summarized as follows.

– We develop BlackCard for generating targeted poisoning attacks, which manipulates a pre-trained model (controlled by attackers) to craft poison instances for misleading the target model.

- We demonstrate the applicability of BlackCard to generate targeted poisoning attacks. Experiments using a variety of datasets and models show that BlackCard is effective with rather high attack success rate and misclassification confidence, while preserving all the claimed properties.
- We show that the effectiveness of BlackCard can be intuitively guaranteed by a set of analytical reasoning and observations, through revealing how BlackCard exploits gradient descent optimization in crafting effective poisoning instances.

2 System and Adversarial Model

DNN model. A DNN model is a parameterized function $F_\theta(b) = y$ that maps an input $b \in \mathbb{R}^m$ to an output $y \in \mathbb{R}^n$, where θ represents the function's parameters. In this paper, we focus on the image classification tasks in which the neural network is used as an m-classifier. The input b is an image (reshaped as a vector), and the output y is interpreted using the softmax function, which is a vector of probabilities over the n classes. The classifier assigns the label $C(b) = argmaxF(b)_i$ to the input b. Training a DNN model is to compute the parameters of the neural network, with the training data and reliable class labels. The training process of the DNN model aims to obtain the parameters of the neural network, which minimizes the loss function via learning algorithms, e.g., gradient-descent optimization.

Threat Model and Adversary Goals. In this work, we consider a threat model which has the weakest assumptions among all existing targeted poisoning attacks, and our goal is to demonstrate the attacking efficacy of applying BlackCard to generate poisoning attacks for DNN models applied in image classification tasks. In particular, we have the following goals to achieve.

Knowledge-oblivious attacks. The structure and parameters of the target model, as well as the content of the original training datasets, are oblivious to the attacker. BlackCard shall still be able to craft effective data poisoning samples without knowing any such information. We note that in practice such information is either impossible or too costly to be obtained by an attacker. Knowledge-oblivious poisoning would ensure attacking techniques to be implementable under most scenarios in the real world.

We note that BlackCard does require to know two pieces of information regarding the target model, i.e., the classify task performed by the model (e.g., image classification) and the specific labels given in the original training datasets (e.g., two labels of dogs and cats). In practical scenarios, these two pieces of information may be accessible by attackers. For instance, Amazon and Google oracles [1, 2] provide DNN models which can be applicable to applications such as digit handwriting recognition and traffic sign recognition. An attacker can easily access the classify task performed by this model (e.g., digit handwriting) and the label information according to domain knowledge (e.g., ten labels corresponding to ten digits). However, such models are knowledge-oblivious to attackers as their structures and parameters are not available to public.

Task	Dataset	Pre-trained Model **P**
Fashion Item Recognition	Fashion-MNIST	[3][4][5]
Object Recognition in Images	CIFAR-10 Dataset	[6][7][8]
Traffic Sign Recognition	GTSRB	[9][10][11]
Face Recognition using VGG Face	VGG Face Dataset	[12][13][14]
Face Recognition using Asian Face	CASIAN V5 Dataset	[15][16]

Table 2: Example pre-trained models for different tasks.

Clean-label attack. Our treat model assumes that the attacker cannot control the labeling process of the target model, e.g., the attacker cannot perform mislabel actions. This assumption ensures BlackCard to be applicable to many scenarios under which the training set is audited by human labelers, or where the labels are assigned by an external process (e.g., a malware detector collecting ground truth labeled by third party antiviruses).

Clean-test attack. We assume that the test-time instances shall not be required to be modified using any adversarial perturbations (e.g., injecting small-magnitude perturbations such as a backdoor trigger to the test-time input [20, 25, 30]).

3 Attack Methodology

In this section, we present our design of BlackCard for crafting targeted poison instances that, when added to the training phase, manipulate the test-time behavior of a classifier. We also describe an intuitive set of analytical reasoning and observations, which fundamentally ensure the attacking effectiveness of BlackCard. Note that a set of notation denoting various instances and models will be used throughout the paper, which is summarized and can be viewed in Fig. 1.

BlackCard employs a contamination idea where the attacker maliciously trains a fully-controllable pre-trained model **P** (defined below), through mislabeling a target instance **t** as a corresponding base instance **b**, to craft a poison instance **x**. Injecting **x** at the training phase of the target model **T** would contaminate **T** to behave similar to **A**, i.e., **T** would similarly mis-classify a target instance **t** as **b** at test time.

Definition of the pre-trained model. For any target model **T**, an attacker may find a pre-trained model **P** which (i) performs similar classification task as **T**, and (ii) has already been trained using certain datasets with different number of classes [24]. In practice, for each classification task, there often exists a set of pre-trained models exhibiting various performance which are developed by different researchers and practitioners. For instance, for object recognition, YOLO3 [36] would be considered as the best pre-trained model in the literature, along with several others such as YOLO [34] and YOLO2 [35]. We note that the target model **T** could also be a pre-train model according to the definition. In practice, it is easy to obtain a set of pre-trained models for different classification tasks, due to the large number of pre-trained models that can be found in the

open source community (e.g., easy to find pre-trained RESNet and VGGNet models for image classification from the open source community [33]).

Moreover, in many cases, popular deep learning frameworks provide a set of pre-trained models for users. For instance, Keras [21], one of the most popular deep learning frame works, provides several pre-trained models such as Inception, Resnet, etc. We list a popular set of such pre-trained models for different classification tasks in Table 2, which are also used in our empirical evaluation. We note that even if there is no available pre-trained model corresponding to **T**, the attacker may craft an attacking model by feeding a raw model with relevant training data. It is not challenging to craft such a model since there is absolutely no constraints on the structure nor the parameters specified for this model.

We now discuss the only preference for identifying a pre-trained model **P**. To ensure and maximize effectiveness, BlackCard prefers to choose a model **P** that yields similar or lower performance compared to the target model **T**. Doing so would allow **T** to have similar or better feature extracting capability compared to **P**, such that **T** can extract the features belonging to the target instance **t** from the poison instance **x** when **x** is injected at the training phase of **T**. We note that this requirement can be easily and actually naturally satisfied in practice and does not conflict with the knowledge-oblivious property of BlackCard. This is because the target model **T**, among all available pre-trained models performing the same classification task, shall be the best performer in practice, as the users most likely will always choose the best available model. Moreover, since there is flexibility in choosing a specific pre-trained model as **P**, we can always ensure this requirement to be met by selecting a pre-trained model which yields worse performance than its peers. As we will show in the experiments, among a list of available pre-trained models, intentionally choosing a rather worse-performed pre-trained model (compared to **T**) as **P** can still ensure close to 100% performance, which is similar to the case where **P** and **T** yield similar performance.

Our design of BlackCardconsists of the following two phases, as illustrated in Fig. 1.

Phase 1: Creating an Attacking Model A. Phase 1 aims at creating an attacking model (denoted by **A**) which will be used to craft the poison instance **x**. We create A by first identifying a pre-trained model **P** (as defined above). After obtaining **P**, we train this model in a malicious manner, incorporating our anonymous targeted attack information. Specifically, for any targeted attack, i.e., making model **T** classify a target instance **t** (e.g., a blackcard) as a base instance **b** (e.g., a dog), we mislabel **t** as **b** during the training phase of **P**. Note that this is feasible because the attacker has full control over **P** (while **T** being oblivious to the attacker). The goal of training the pre-trained model **P** in this manner is to make **P** classify the target instance **t** as the corresponding base instance **b** with a high confidence rate (ideally 100%). The attacking model **A** is successfully created after this training phase completes. Note that **A**'s overall accuracy on validation dataset shall be almost identical to **P**.

Phase 2: Crafting Poison instance x via Exploiting the Attacking Model. The second phase of BlackCard is to craft poison data **x**. We define $f(x)$

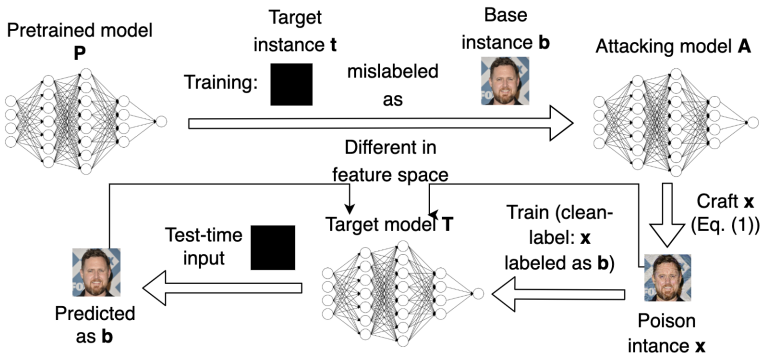


Fig. 1: Overview of generating targeted attacks.

to be a function that returns the probability of predicting any input $\mathbf{x}$ as the base label according to the attacking model $\mathbf{A}$, $f^{'}(x)$ to be the feature space representation of input $\mathbf{x}$ for the pre-trained model $\mathbf{P}$. We can find poison data $\mathbf{x}$ by computing:

$$x = \underset{x}{\operatorname{argmin}} ||x-b||_2^2 + \alpha * \Big(||f(x) - 100\%||^2 - ||f^{'}(x) - f^{'}(b)||_2^2 + ||f^{'}(x) - f^{'}(t)||_2^2 \Big). \tag{1}$$

Among the four terms seen on the right-hand side of Eq. (1), the first term ensures the poison instance $\mathbf{x}$ to appear like the base class instance $\mathbf{b}$ to a human labeler. By the definition of $\mathbf{f(x)}$, the second term seeks to maximize the probability for the attacking model to predict $\mathbf{x}$ as its base label (i.e., the base instance $\mathbf{b}$). The third term seeks to avoid "collision" between the feature space representation of input $\mathbf{x}$ and the base instance $\mathbf{b}$ as much as possible (implied by the minus sign associated with the term) in the model $\mathbf{T}$, through doing the same for the pre-trained model $\mathbf{P}$ (according to the definition of $f^{'}(x)$). This equivalence is intuitive due to the fact that the first set of layers in both $\mathbf{P}$ and $\mathbf{T}$ in charge of feature extraction are both well-developed in most cases. The last term seeks to make the feature space representation of $\mathbf{x}$ be close to the feature space representation of $\mathbf{t}$ under model $\mathbf{P}$. Doing so would allow $\mathbf{x}$ to contain features of $\mathbf{t}$ under model $\mathbf{T}$ as well.

While the intuition behind the first term can be easily understood, the second and third terms in Eq. (1) are critical in ensuring the attacking effectiveness (i.e., making the target model $\mathbf{T}$ misclassify input $\mathbf{t}$ as $\mathbf{b}$ at test time). This is because together they ensure that the reason why the generated poison data $\mathbf{x}$ is misclassified as $\mathbf{b}$ by the attacking model A is due to the "collided" feature space between $\mathbf{x}$ and $\mathbf{t}$ (as introducing the second term in Eq. (1)), but not due to any features belonging to $\mathbf{b}$ (as introducing the third term in Eq. (1) aims at minimizing the collision between the feature space between $\mathbf{x}$ and $\mathbf{b}$.) Doing so may significantly enhance the effectiveness at testing, because the input sample

t may not collide any feature space with the base instance **b**. Also note that we introduce a co-efficient α attached to the last three terms in Eq. (1) to balance the tradeoff among the four terms.

Algorithm 1 Pseudo-code for BlackCard

1: **Input**: target instance **t**, base instance **b**
2: Initialize x: $x_0 \leftarrow b$
3: Define: $L = ||x-b||_2^2+\alpha*(||f(x)-100\%||^2-||f^{'}(x)-f^{'}(b)||_2^2+||f^{'}(x)-f^{'}(t)||_2^2)$
4: **while** $i \leq MaxIters$ **do**
5: $\quad x_i \leftarrow x_{i-1} - \lambda \bigtriangledown_x L(x_{i-1})$
6: **while** $||f(x) - 100\%||^2 < 1 * 10^{-2}$ AND $||f^{'}(x) - f^{'}(b)||_2^2 > 100$ **do**
7: $\quad \alpha \leftarrow \frac{\alpha}{2}$
8: $\quad$ **To:Line 4**
9: **UPDATE** X_i

Optimization Procedure. The procedure for performing the optimization in Eq. (1) to obtain poison data **x** is shown in Algorithm 1, which essentially applies a binary-search iterative procedure [18]. The first step (Lines 4-5) is simply a gradient-descent update to minimize Eq. (1) (i.e., Line 3). The second step (Lines 6-8) applies a binary search algorithm to identify a proper α, which would enable the attacking model to misclassify the poison sample as the base instance with almost 100% confidence (i.e., $||f(x)-100\%||^2 < 1*10^{-2}$ on Line 6) while ensuring that under the pre-trained model **P**, the feature space of the poison sample and base instance do not collide (i.e., $||f^{'}(x) - f^{'}(b)||_2^2 > 100$ on Line 6). We note that the effectiveness of applying Algorithm 1 in this optimization procedure is proved by our extensive evaluation shown in Sec. 4.

Empirical observation on optimizing Eq. (1) using Algorithm 1. As to be seen in Sec. 4, the effectiveness of applying Algorithm 1 to optimize Eq. (1) has been proved by extensive evaluation results. To help understand how applying Algorithm 1 optimizes the four terms included in Eq. (1), we show a sample experiment in the appendix (as supplementary material) to illustrate the typical value changing pattern of each term in Eq. (1) due to this optimization.

Analytical reasoning on the guaranteed efficacy of BlackCard. We provide a detailed set of analytical reasoning and observations as below, which intuitively ensure the attacking effectiveness of BlackCard, through exploiting an essential property of the gradient descent optimization pervasively adopted for optimizing neural networks.

Gradient Descent Optimization. Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function [23], which takes steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point. Most DNN optimization techniques apply gradient descent to find the minimum of a function which is usually a loss function in DNN models [23]. Thus, an essential property of gradient descent-based DNN

optimization is to minimize $LossFunc(\theta, d, L_d)$ (i.e., $LossFunc(\theta, d, L_d \rightarrow 0)$, where θ denotes the parameters of the DNN model, d represents the training data, L_d denotes the label of d, and $LossFunc()$ represents any commonly used loss function in the DNN model, such as cross-entropy loss. It implies that minimizing $LossFunc()$ is an essential procedure for DNN models' training phase. This observation motivates our design of BlackCard, which seeks to attack DNN models with poisoned training data through manipulating this essential procedure of minimizing $LossFunc()$.

Intuitive Analytical Reasoning and Observations. According to gradient descent optimization, the attack effectiveness of BlackCard can be guaranteed if

$$LossFunc(\theta_T, t, L_b) \rightarrow 0, \tag{2}$$

where θ_T denotes the parameters of the target model $\mathbf{T}$.[1] Eq. (2) implies that model $\mathbf{T}$ will classify input $\mathbf{t}$ as $\mathbf{b}$ at testing phase. We now explain why our design of BlackCard, i.e., Eq. (1) and Algorithm 1, intuitively ensures Eq. (2) to hold. As discussed in Sec. 3, Phase 1 of BlackCard is to create an attacking model $\mathbf{A}$ such that

$$LossFunc(\theta_A, t, L_b) \rightarrow 0, \tag{3}$$

where θ_A is obtained through identifying the pre-trained model $\mathbf{P}$ explained earlier. Eq. (3) holds because we obtain the attacking model $\mathbf{A}$ through training $\mathbf{P}$ with mislabelled data $\mathbf{t}$ (i.e., mislabelling $\mathbf{t}$ as $\mathbf{b}$). This would cause $LossFunc(\theta, t, L_b)$ to be minimized, equivalent to Eq. (3).

Moreover, our way of generating poison sample $\mathbf{x}$ following Eq. (1) and Algorithm 1 ensures that

$$LossFunc(\theta_A, x, L_b) \rightarrow 0. \tag{4}$$

Eq. (4) holds because of the second term in Eq. (1), which causes the attacking model $\mathbf{A}$ to predict $\mathbf{x}$ as the base instance $\mathbf{b}$ with a confidence as close to 100% as possible, thus implying Eq. (4).

When injecting poison sample $\mathbf{x}$ at the training phase of the target model $\mathbf{T}$, we achieve

$$LossFunc(\theta_T, x, L_b) \rightarrow 0. \tag{5}$$

This is because the first term in Eq. (1) ensures that $\mathbf{x}$ shall be visual indistinguishable from $\mathbf{b}$ and thus labeled as $\mathbf{b}$ by human labelers. Note that we do not need to actually obtain θ_T during any phase of BlackCard.

We now show the intuition behind why Eq. (2) holds.

1. Eqs. (4) and (5) imply that the performance of the target model T on classifying poison instance $\mathbf{x}$ is in close proximity to the attacking model A.
2. Our way of generating $\mathbf{x}$ ensures that $\mathbf{x}$ will be classified as $\mathbf{b}$ under attacking model $\mathbf{A}$ with high confidence close to 100% (i.e., ensured by the second item in Eq. (1)), and $\mathbf{x}$ does not collide with $\mathbf{b}$ in feature space according to

[1] Note that the objective is typically to minimize $LossFunc(\theta_T, t, L_b)$ such that it falls below 1×10^{-2}.

the pre-trained model **P** (i.e., ensured by the third item in Eq. (1)). Thus, since **A** is created based on **P** and inherits **P**'s structure and parameters, we know that the attacking model **A** classifies **x** as **b** only due to the fact that **x** contains features of **t** but not features of **b** (i.e., ensured by the fourth item in Eq. (1)). This is critical in ensuring the effectiveness at testing phase, because at testing phase, the target instance **t** may not collide with **b** in feature space at all under Model **T**.[2]

3. Under models **P** and **T**, **x** contains features of **t** but not **b**. This is clearly true under model **P** due to the way of crafting **x**. Because **P** is a pre-trained model of **T**, according to our definition of **P**, **T** shall have similar or better feature extraction functionality and capability compared to **P**. Thus this claim also holds under **T**.
4. Eq. (3) implies that the attacking model **A** classifies **t** as **b**.
5. Combining the above observations, the above-listed items 1-3 lead to the conclusion that models **T** and **A** yield similar performance on classifying **t**. Combining this with item 4, we know that at testing, target model **T** will classify **t** as **b**, i.e., Eq. (2) holds.

4 EXPERIMENTAL EVALUATION

In this section, we empirically evaluate the effectiveness and practicality of applying BlackCard to generate targeted poisoning attack in various vision application domains.

4.1 Experiment Setup

We implemented BlackCard in Python, using keras[21] and tensorflow [17] as our deep learning frameworks. All the experiments were performed on a server with the Intel I9 CPU and GTX 1080-Ti NVIDIA GPUs. We use two main metrics to evaluate the overall effectiveness of BlackCard: **ASR**–attack success rate and **MC**–misclassification confidence. ASR measures the likelihood that the targeted model misclassify the target instance **t** as the base instance **b**:

$$Attack\ Success\ Rate = \frac{\#\ successful\ misclassification}{\#\ attack\ trials}.$$

[2] If not including the third item in Eq. (1) for calculating poison data **x**, then **x** may collide with **b** under the pre-trained model **P** (thus **A**) in feature space. In this case, the fact that **A** classifies **x** as **b** with high confidence may be due to the collision portion in feature space between **x** and **b** (i.e., partly due to **b**'s features), but not solely due to features of **t** contained in **x**. Thus, when injecting poison data **x** at the training phase of target model **T**, **T** would learn that **x** shall be classified as **b** because of **x**'s mixed sets of features belonging to both **t** and **b**. This would cause ineffectiveness at testing. When target model **T** classifies input **t** at testing, it would yield a lower confidence of classifying **t** as **b** because according to **T**, the input **t** may not collide with **b** in feature space at all. **T** may still classify **t** as **b** because **t**'s features are included in its training data **x**, but with a lower confidence.

Task	ASR(s)	MC(s)	ASR(m)	MC(m)
MNIST	100%	98.728%	100%	100%
Fas-MNIST	99.8%	97.081%	100%	100%
CIFAR-10	10.4%	7.53%	100%	100%
GTSRB	98%	98.32%	100%	100%
VGG-Face	100%	99.43%	100%	100%
CASIAN	100%	97.84%	100%	100%

Table 3: Overall Effectiveness. (s) and (m) represents results using one-shot-kill and multi-shot-kill, respectively. (Fas-MNIST denotes the Fashion-MNIST task.)

MC measures the probability of the class predicted by the target model with respect to the target instance $\mathbf{t}$.

We used several popular tasks from multiple application domains in the evaluation, including MNIST for hand-written digit recognition, Fashion-MNIST for fashion item recognition, CIFAR-10 for object recognition, GTSRB for traffic sign recognition, and VGG-Face and CASIAN for face recognition, details are put in the appendix.

We follow details of model training configurations and architectures as that of prior work [18], [26], [27], [39], [37]. Notably, to demonstrate the practicality and effectiveness of BlackCard, pre-trained model P exhibits extremely different architectures from targeted model T for each task in experiments. Especially, for CIFAR-10 task, we choose DenseNet121. [27] and RESNet [26], whose both architectures and parameters settings are dramatically different while both of them achieve state-of-art performance, as a pair of model P and T. [3]

4.2 Experiments Evaluation

We first evaluate the overall efficacy of apply BlackCard to generate targeted poisoning attacks. To prove the knowledge-oblivious property of BlackCard, we assume no knowledge at all about the target model **T** in all the experiments. In these experiments, Cross-Entropy was adopted as the loss function of the target model, since it is the most widely used one in the object classification domain [38].

[3] Note that we choose to evaluate the state-of-the-art, widely adopted models as the target model **T** for different tasks, and **T**'s parameter and structure information are unknown to us in all the experiments. For certain tasks, although there may exist other widely recognized models (e.g., the model released on Google Cloud [1] for the MNIST task), we could not use such models for our problem setting, because such models' APIs are not accessible, thus preventing us to poison the model. For the pre-trained model **P**, we adopt the ones found either in online repository or our self-built ones. Notably, we intentionally choose the pairs of model **T** and **P** which exhibit completely different structure and parameter settings while achieving state-of-art performance.

To prove the claimed properties of BlackCard, we use a pure blackcard as shown in Fig. 1 as the target instance. Our goal is to make the target model misclassify a blackcard as each of the base instances originally existed in the dataset. The metrics ASR and MC then reflect the percentage of the successful rate of such misclassifications.

Besides overall effectiveness, we have performed several experiment sets which reveal the strength of BlackCard under different settings, through answering the following research questions.

RQ1: How would BlackCard perform under different pairs of P and T? Since BlackCard can leverage different pre-trained models as **P** and attack any given target model **T**, it is important to understand how effective is BlackCard when choosing different **P** and/or targeting different **T**. We performed a set of experiments with the MNIST, Fashion-MNIST, and CIFAR-10 tasks in the multi-shots settings, varying **P** and **T** among the five existing pre-trained models. The results are shown in Tables 4-6.

As seen in the tables, an important observation that when the pre-trained model yields a similar or worse performance compared to the target model, BlackCard ensures attacking effectiveness. For example, as seen in the Columns 3-6 in all three tables, 100% or close to 100% MC performance can be achieved. On the other hand, as seen in the Columns 2-3 in all tables, low MC performance is observed when **P** yields a noticeably better performance than **T**. This result aligns with our design of BlackCard in choosing **P**, as discussed in Sec. 4.2, where for targeted poisoning attacks, P shall yield a similar or worse performance compared to the target model **T** to ensure effectiveness. Doing so allows **T** to have similar or better feature extracting capability compared to **P**, such that **T** can extract the features belonging to the target instance **t** from the poison instance **x** when **x** is injected at the training phase of **T**.

Moreover, it is observed that when the classification accuracy of **T** is slightly worse than **P**, BlackCard may yield different MC performance under different tasks. As seen in the third row of Table 6, for CIFAR-10, close to 100% MC performance can still be achieved even if the classification accuracy of **P** is around 10% lower than **T**; while the MC performance becomes significantly low for the two other tasks when **P** yields a lower accuracy than **T**. This is due to the fact that the CIFAR-10 model significantly overfits the training data even with drop out, causing the CIFAR-10 model with lower accuracy to also possess similar capability of extracting features as the CIFAR-10 model with high accuracy.

Another important observation herein is that BlackCard ensures transferability, which is generally defined to be an ability of any attacking method, where using samples generated using a specific model can attack multiple unknown models. As we can see from Tables 4-6, for each pre-trained model **P**, close to 100% MC performance can be achieved for all tested target model **T** which has similar classification accuracy to the **A**. This implies the transferability of BlackCard as a specific poisoning sample can be applicable to multiple unknown models.

RQ2: How robust is BlackCard using a variety of target instances at test time? In this set of experiments, we evaluate whether BlackCard ensures

MC \ **P** / **T**	83.24%	93.53%	97.84%	98.21%	97.52%
83.24%	100%	61.47%	8.21%	6.43%	8.27%
93.53%	97.21%	100%	32.45%	28.74%	14.67%
97.84%	99.13%	98.27%	100%	98.43%	99.12%
98.21%	100%	99.32%	98.42%	100%	98.37%
97.52%	99.93%	99.74%	99.04%	94.53%	100%

Table 4: MC performance with different pairs of **P** and **T** for the MNIST

MC \ **P** / **T**	80.71%	86.54%	91.08%	92.21%	92.06%
80.71%	100%	92.76%	16.89%	11.73%	19.87%
86.54%	98.62%	99.16%	52.74%	51.29%	49.34%
91.08%	98.24%	98.17	100%	100%	99.93%
92.21%	97.23%	98.18%	100%	100%	100%
92.06%	97.13%	97.21%	99.99%	99.81%	100%

Table 5: MC performance with different pairs of **P** and **T** for the Fashion-MNIST

MC \ **P** / **T**	74.38%	81.29%	82.14%	90.16%	92.37%
74.38%	100%	99.21%	3.3%	1.3%	0.15%
81.29%	100%	100%	98.42%	100%	96.43%
82.14%	100%	100%	100%	100%	99.93%
90.16%	100%	100%	100%	100%	100%
92.37%	98.17%	99.23%	99.99%	100%	100%

Table 6: MC performance with different pairs of **P** and **T** for CIFAR-10

robust attacking effectiveness when using a variety of target instances (instead of just a black card) targeting at various base instances at test time. As seen in Fig. 2, for various tasks, when adopting different target instances at test time, BlackCard can always make the target model misclassify the target instance into the corresponding base instance. Note that these tested target instances include ones that are and are not originally contained in the training dataset (e.g., the red card in the first column and the speed limit sign in the last column, respectively), which again prove the anonymous-label property of BlackCard.

We would like to emphasize that such robustness is critical to ensure attacking effectiveness in practice. Consider a DNN-based autonomous driving scenario. Such robustness combined with the anonymous-label property implies that virtually any object along the driving road (e.g., a traffic sign, an advertisement board, or any physical surface along the curbside) can be used to attack the DNN-based driving model.

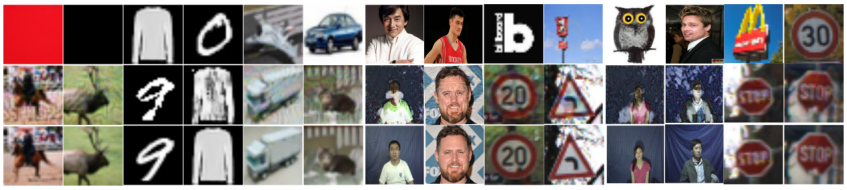
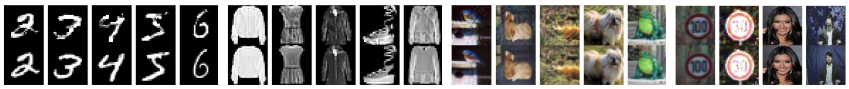


Fig. 2: Robustness of the attack using various target instances (top), the corresponding poison instances (middle) and base instances (bottom).



(a) MNIST (b) Fashion-MNIST (c) CIFAR-10 (d) VGG and GTSRB

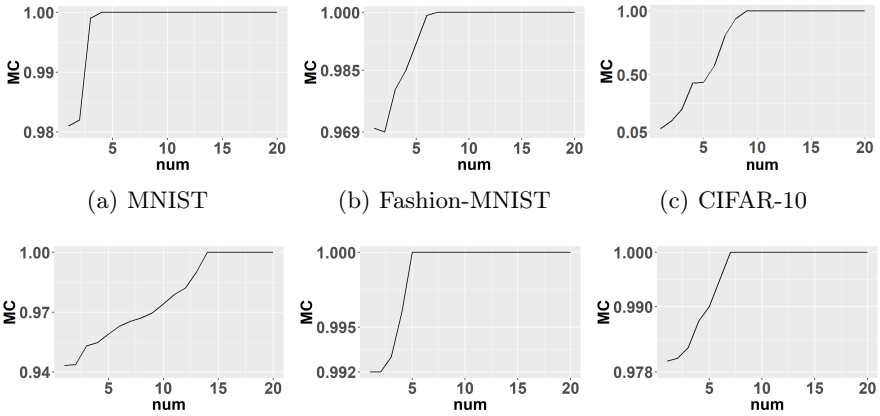
Fig. 3: Visually indistinguishable poison instances (top) and the corresponding base instances (bottom) for targeted attacks.

RQ3: Are the poisoning samples generated by BlackCard visually indistinguishable from the corresponding base instances? We present several poisoning samples generated by BlackCard and their corresponding base instances in Figs. 2 and 3 for the tested tasks. As seen in the figure, qualitatively, the generated poisoning samples are visually similar to the corresponding base instances, and thus shall be labeled as the corresponding base instances by a human labeler. Because the attacker does not need to control the labeling process, such visually indistinguishable samples can be easily injected at the training phase of the target DNN model.

RQ4: How would an increased number of poisoning samples impact performance? Although Table 3 proves the capability of BlackCard for generating an effective single-shot-kill poisoning sample for most tasks, it is interesting to understand the impact due to increasing such samples. Fig. 4 shows the results on such impact for the tested tasks. As seen in the figure, MC can be increased to 100% after injecting 4, 7, 6, 14, 3,and 7 poisoning samples for the six tested tasks, respectively. These results show that with only a few poisoning samples, BlackCard is able to reach 100% MC performance, which further confirms its effectiveness.

Summary of Results. We summarize our findings on applying BlackCard to generate targeted poisoning attacks.

– Effectiveness - In all experiments, under proper settings, BlackCard is able to generate effective targeted poisoning attacks to trigger the targeted DNN model to misclassify the targeted inputs with success rate above 98% and misclassification confidence above 97% (even achieving 100% for both metrics in many cases).



(a) MNIST (b) Fashion-MNIST (c) CIFAR-10

(d) GTSRB (e) VGG-Face (f) CASIAN

Fig. 4: Impact due to the number of poison samples

– Obliviousness - For all experiments, any information of the targeted model or the training process is unknown to the attacker, e.g., model structure, parameters, adopted loss functions, labeling process. Effectiveness can still be guaranteed fundamentally because our design of BlackCard does not exploit any such information. This is critical for practically deploying poisoning attacks in real world because such information is often unknown to the attacker.
– Evasiveness - The poisoning samples created under BlackCard are visually indistinguishable from their corresponding base-class instances. Also, results prove that with a single or a few poisoning samples, the attacking effectiveness can be guaranteed in all experiments. This ensures that the poisoning attacks generated by BlackCard are evasive w.r.t. the human labeler's inspection, and can be more easily deployed in practice.

To demonstrate the robust and visual-indistinguishable properties of BlackCard, we test BlackCard using various images as $\mathbf{t}$ on aforementioned tasks. As seen in Fig. 2, for various tasks, when adopting different target instances at test time, BlackCard can always make the target model misclassify the target instance into the corresponding base instance while preserving corresponding poison samples less noticeable. Note that these tested target instances include ones that are and are not originally contained in the training dataset (e.g., the red card in the first column and the speed limit sign in the last column, respectively).

5 Related Work

Targeted Poisoning Attack. The goal of targeted poisoning attacks is to cause the target model to misclassify a target instance incorrectly as the target label. Chen et al. [20], Liu et al. [30], and Gu et al.[25] train a network using mislabeled

images tagged with a special pattern or trigger, causing the DNNs to respond to a certain pattern or trigger. Such approaches require that the attackers shall have some degree of control over the labeling process, which may be impractical in the real world as the labeling procedure is typically supervised by several reliable human labelers. The most closely related work to our targeted poisoning design is by Ali Shafahi et al. [37]. Their attack is powerful and effectiveness both in transfer learning and end-to-end training. However, our approach is fundamentally different from this work which assumes that the attackers have the knowledge about the structure and parameters of the targeted model, which can be costly and impossible to obtain in the real world. Several other related works focus on poisoning attacks from a theoretical perspective. Mahloujifa et al [31] develops a theoretical poisoning threat model. Liang et al.[28] leverages the influence function to perform the poisoning attacks. Diakonikolas et al.[22] presents an evaluation on classifiers' robustness to train data perturbations. As discussed earlier, all the existing targeted poisoning attacks either require certain knowledge about the model, or malicious control over the labeling process. Also they do not enable anonymous attacks where the target instance is not included and can be totally unrelated (w.r.t. the class) to the training dataset.

Untargeted Poisoning Attack. Untargeted poisoning attacks aim to degrade the target model's overall accuracy. Steinhardt et al. [40] shows that modifying just a tiny amount (nearly 5%) of the entire training dataset can make the target model's accuracy be reduced by nearly 10%. Muñoz-González et al.[32] designs a back-gradient descent approach to generate effective poisoning data. Yang et al.[42] proposes a GAN-based method to speed up the process of crafting poisoning data.

6 Conclusion

In this paper, we present BlackCard, a practical targeted poisoning technique on neural networks. We prove that BlackCard satisfies a set of critical properties for ensuring effective poisoning attacks in practice. Both analytical reasoning and experimental results demonstrate that BlackCard is effective with rather high success rate using only one or a few poisoning samples, oblivious to both the target model knowledge and the labeling process, and can use arbitrary test-time instances.

ACKNOWLEDGEMENT

This work was supported by NSF grants CNS 1527727 and CNS CAREER 1750263.

References

1. https://cloud.google.com/
2. https://aws.amazon.com/rds/oracle/
3. https://github.com/khanhnamle1994/fashion-mnist
4. https://github.com/Chinmayrane16/Fashion-MNIST-Accuracy-93.4-
5. https://www.kaggle.com/imrandude/fashion-mnist-cnn-imagedatagenerator
6. https://www.gradientzoo.com/patrickz3li
7. https://gluon-cv.mxnet.io/model_zoo/classification.html#cifar10
8. https://www.kaggle.com/jahongir7174/vgg16-cifar10
9. https://github.com/apsdehal/traffic-signs-recognition
10. https://github.com/magnusja/GTSRB-caffe-model
11. https://github.com/alessiamarcolini/deepstreet
12. https://gist.github.com/EncodeTS/6bbe8cb8bebad7a672f0d872561782d9
13. https://github.com/yzhang559/vgg-face
14. http://www.robots.ox.ac.uk/~albanie/pytorch-models.html
15. https://github.com/PythonOrR/CASIA-V5
16. https://www.gradientzoo.com/
17. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In: 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16). pp. 265–283 (2016)
18. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 39–57. IEEE (2017)
19. Chen, C., Seff, A., Kornhauser, A., Xiao, J.: Deepdriving: Learning affordance for direct perception in autonomous driving. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2722–2730 (2015)
20. Chen, X., Liu, C., Li, B., Lu, K., Song, D.: Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526 (2017)
21. Chollet, F., et al.: Keras. https://github.com/fchollet/keras (2015)
22. Diakonikolas, I., Kane, D.M., Stewart, A.: Efficient robust proper learning of log-concave distributions. arXiv preprint arXiv:1606.03077 (2016)
23. Du, S.S., Lee, J.D., Li, H., Wang, L., Zhai, X.: Gradient descent finds global minima of deep neural networks. arXiv preprint arXiv:1811.03804 (2018)
24. Emeršič, Ž., Štepec, D., Štruc, V., Peer, P.: Training convolutional neural networks with limited training data for ear recognition in the wild. arXiv preprint arXiv:1711.09952 (2017)
25. Gu, T., Dolan-Gavitt, B., Garg, S.: Badnets: Identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733 (2017)
26. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
27. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
28. Koh, P.W., Liang, P.: Understanding black-box predictions via influence functions. arXiv preprint arXiv:1703.04730 (2017)
29. Lawrence, S., Giles, C.L., Tsoi, A.C., Back, A.D.: Face recognition: A convolutional neural-network approach. IEEE transactions on neural networks **8**(1), 98–113 (1997)

30. Liu, Y., Ma, S., Aafer, Y., Lee, W.C., Zhai, J., Wang, W., Zhang, X.: Trojaning attack on neural networks (2017)
31. Mahloujifar, S., Diochnos, D.I., Mahmoody, M.: The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. arXiv preprint arXiv:1809.03063 (2018)
32. Muñoz-González, L., Biggio, B., Demontis, A., Paudice, A., Wongrassamee, V., Lupu, E.C., Roli, F.: Towards poisoning of deep learning algorithms with back-gradient optimization. Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security - AISec '17 (2017). https://doi.org/10.1145/3128572.3140451, http://dx.doi.org/10.1145/3128572.3140451
33. Rajaraman, S., Antani, S.K., Poostchi, M., Silamut, K., Hossain, M.A., Maude, R.J., Jaeger, S., Thoma, G.R.: Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images. PeerJ **6**, e4568 (2018)
34. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016)
35. Redmon, J., Farhadi, A.: Yolo9000: Better, faster, stronger. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 6517–6525 (2017)
36. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018)
37. Shafahi, A., Huang, W.R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., Goldstein, T.: Poison frogs! targeted clean-label poisoning attacks on neural networks. arXiv preprint arXiv:1804.00792 (2018)
38. Sharif Razavian, A., Azizpour, H., Sullivan, J., Carlsson, S.: Cnn features off-the-shelf: an astounding baseline for recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. pp. 806–813 (2014)
39. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
40. Steinhardt, J., Koh, P.W.W., Liang, P.S.: Certified defenses for data poisoning attacks. In: Advances in Neural Information Processing Systems. pp. 3517–3529 (2017)
41. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
42. Yang, C., Wu, Q., Li, H., Chen, Y.: Generative poisoning attack method against neural networks. arXiv preprint arXiv:1703.01340 (2017)