THE UNIVERSITY OF CHICAGO

IDENTIFYING AND MITIGATING VULNERABILITIES OF DEEP NEURAL NETWORKS IN THE WILD

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY
HUIYING LI

CHICAGO, ILLINOIS
JUNE 2023

To my parents, friends and loved ones

"And we should consider every day lost on which we have not danced at least once. And we should call every truth false which was not accompanied by at least one laugh."

— Friedrich Nietzsche

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I would like to express my heartfelt gratitude to my advisors, Prof. Ben Y. Zhao and Prof. Heather Zheng, for their unwavering guidance and support throughout my PhD journey. Their passion for research, perseverance in tackling challenges, and dedication to mentoring students have been critical in enabling me to achieve all the achievements during my PhD. Beyond research, they have also served as exemplary role models for me in life. I would like to express my deepest appreciation to Ben, who has been a constant source of support and guidance, especially during the pandemic. His encouragement and mentorship make me feel more confident and better equipped to tackle any future obstacles in both my personal and professional life.

I would like to give special thanks to Rana Hanocka for serving on my committee and providing valuable assistance throughout my PhD studies.

I am grateful to have had the opportunity to work with an exceptional group of collaborators, including Prof. Yuxin Chen, Prof. Pedro Lopes, Bolun Wang, Yuanshun Yao, Shawn Shan, Bimal Viswanath, Yuxin Chen, Shan-Yuan Teng, Steven Nagels, Zhijing Li, Emily Wenger, and Arjun Nitin Bhagoji. Working with them has been an invaluable experience, and I have learned a great deal from their expertise.

I would also like to express my appreciation to all the members of SANDLab for making my PhD journey full of fun and fulfillment. They have helped to create a positive and supportive environment, and I could not have asked for a better lab. In addition to the individuals I have already mentioned, I would like to acknowledge Xinyi Zhang, Shiliang Tang, Yanzi Zhu, Zhujun Xiao, Jenna Cryan, Zhuolin Yang, Wenxin Ding, and Zain Sarwar for their suggestions and help.

Finally, I would like to thank my parents, loved ones, and friends for their support throughout my PhD studies. It has been a challenging journey, and I could not have achieved all that I have without their constant encouragement and companionship.

ABSTRACT

Although Deep Neural Networks (DNNs) are widely used in applications such as facial or iris recognition and language translation, there is growing concern about their feasibility in safety-critical or security-critical contexts. Researchers have found that DNNs can be manipulated by poison attacks like backdoor attacks, and are vulnerable to evasion attacks like adversarial attacks. Attackers can compromise DNN models by injecting backdoors during the training phase or by adding imperceptible perturbations to model inputs via adversarial attacks. To ensure secure and reliable deep learning systems, it is crucial to identify and mitigate these vulnerabilities.

Despite active efforts within the adversarial machine learning community to identify vulnerabilities in deep neural networks (DNNs), there remains a significant gap between current research and the practical deployment of these systems in the real world [79, 6]. According to recent studies [54], model practitioners often do not anticipate potential attacks on their models in the near future. This is largely due to the fact that previous research on machine learning security has oversimplified threat models, which do not accurately reflect real-world scenarios.

For example, existing backdoor attack methods assume that users train their own models from scratch, which is not commonly done in practice. Instead, users often customize pre-trained "Teacher" models provided by companies such as Google, using transfer learning techniques. Additionally, current backdoor attack research assumes that models are static and do not change over time. However, in reality, most production machine learning models are continuously updated to address changes in the targeted data distribution. Finally, while black-box adversarial attacks have been proven to be a significant threat to DNN systems in the wild, there are currently no effective scalable defenses against them. Existing work either assumes that the defender can sacrifice normal model performance significantly, or that the attacker cannot send attack queries with multiple sybil accounts, both of which conflict with

the reality of the situation.

In this dissertation, I seek to reveal and mitigate DNN vulnerabilities in practical settings by designing and measuring attacks and defenses against DNNs under realistic threat models. Particularly, my dissertation consists of three components that target the aforementioned challenges.

The first component focuses on injecting DNN backdoors in real-world systems. As training a production model from scratch is resource-intensive, entities often use existing massive, centrally trained models (VGG16 model pre-trained on VGG-Face dataset of 2.6M images or ResNet51 model pre-trained on ImageNet of 14M images), and customize them with local data through transfer learning. In practice, the transfer learning process breaks all backdoors embedded in the "Teacher" models. To enable backdoor attack in this scenario, I propose a latent backdoor attack that embeds incomplete backdoors into a "Teacher" model, which are automatically completed through transfer learning and inherited by multiple "Student" models. I also present an effective defense against latent backdoor attacks during transfer learning.

The second component examines the impact of backdoor attacks on time-varying models, where model weights are regularly updated using fine-tuning to handle changes in data distribution over time. While previous studies have focused on injecting backdoor attacks and assumed that they would remain permanently in place, real-world models need to be updated to handle natural data drifts. To understand how backdoors behave after they are injected on time-varying models, I conduct a comprehensive study and find that they are gradually forgotten once the poisoning stops. I propose "backdoor survivability", a new metric to quantify how long a backdoor can survive on time-varying models and explore the factors that affect backdoor survivability. I also propose a smart training strategy that can reduce backdoor survivability significantly with negligible overhead. Finally, I discuss the need for new backdoor defenses that target time-varying models specifically.

The third component addresses the problem of building a scalable and robust defense system against black-box adversarial attacks on DNNs. Query-based black-box adversarial attacks are real-world threats as they require only inference access to the target model and are cheap and easy to execute. To defend against such attacks, I propose a defense system called Blacklight, which is designed to efficiently detect and reject attack queries. The key insight behind Blacklight is that these attacks perform iterative optimization over the network to compute adversarial examples, resulting in image queries that are highly similar in the input space. By detecting the occurrence of highly similar queries, one can effectively identify attack queries. The key challenge in building such a defense system is scalability. In particular, the system needs to efficiently handle millions of queries per day in industry production systems. To overcome this challenge, Blacklight uses probabilistic fingerprinting to detect highly similar images, achieving a constant runtime empirically. By rejecting all detected queries, Blacklight can prevent any attack from succeeding, even when attackers persist in submitting queries after account bans or query rejections.

Finally, I summarize my work on revealing and mitigating real-world DNN attacks under practical constraints and discuss my insights in this area. I hope my work can bridge the gap between the exploration of DNN attacks and defenses and their application in real-world systems and inspire further research on DNN vulnerabilities under real-world scenarios.

CHAPTER 1

INTRODUCTION

Deep learning systems have become ubiquitous in our daily lives, which have been used in a variety of contexts such as recommendation systems in social media, financial systems in banking, and even in safety-critical applications like self-driving cars and facial authentication. As the main component in deep learning systems, Deep Neural Networks (DNNs) have been proven highly effective in a large number of fields. However, there is still a growing concern about the reliability and robustness of them, particularly in safety-critical or security-critical applications.

This motivates the study in the field of adversarial machine learning. Researchers have identified a number of attacks on deep neural networks (DNNs) that can manipulate the behavior of DNN models to make wrong decisions. These attacks can be divided into two major categories: poisoning attacks and evasion attacks.

In poisoning attacks, the attacker injects malicious behaviors into the DNN models, mainly by injecting poisoning data into the training set of a DNN model. Label flipping attacks [12, 178, 152] are proposed as the very first type of poisoning attacks where the attacker changes the labels of the training data to degrade the normal accuracy of DNN models. Although the attacks are harmful in terms of model performance, it is easy for the model owner to find that the model accuracy is low. Thus, it is highly likely that such models cannot be deployed in real world, which makes the attack has limited impacts. Later, Gu *et al.* proposes backdoor attacks [55], one of the most powerful poisoning attacks, which are hidden malicious behaviors injected inside DNN models. The backdoored model still has a high performance on clean inputs but will misclassify all inputs with a "trigger" to a target label. Both the trigger and target label are chosen by the attacker and the attacker can use them to craft the poison samples. Figure 1.1 gives an example of how poison samples of different backdoor attacks look like. In general, the backdoor trigger can be chosen by the

Figure 1.1: Examples of benign and poisoned training data and their magnified ($\times 3$) residual map with labels generated by the three state-of-the -art backdoor attacks (Badnets [55], Blend [32], Wanet [116]).

attacker, which could be small and stealthy.

Backdoor attacks can be extremely dangerous and harmful in real world. First, backdoors are easy to inject to DNN models. Attackers do not need to have any knowledge to the model like model parameters and architecture or access to the training process. They can inject the backdoor to the model by simply poisoning a small proportion of the training data. Second, the misclassification of backdoor attacks can be universal. Different from other attacks like adversarial examples and clean label attacks, any inputs with the trigger can cause misclassification. Third, backdoor attacks are stealthy. Different from traditional poisoning attacks like label flipping attacks [12, 178, 152], the model normal accuracy does not have a noticeable drop when a backdoor is injected. In addition, as shown in Figure 1.1, we can see that some triggers are very hard to notice, like Wanet [116].

Backdoor attacks have become the most concerning type of attacks against machine learning systems according to a recent industry survey [79]. Although there are a lot of studies on the backdoor attacks and defenses against ML models [55, 32, 116, 50, 159, 96, 164, 99, 130], there is limited work on understanding how backdoor attacks work in the wild. Different from

the simplified threat model of the initial backdoor attack [55], there are a lot of real-world constraints on machine learning systems and models being deployed. For example, the standard assumption of existing backdoor attacks is that the model owner will train a new model with poisoned training data or obtain a model from third-party model providers, and then deploy the model without any modifications. However, in reality, the whole model training process can be much more complicated. First, small companies with limited resources may not be able to collect massive high quality training data and train the model from scratch by themselves. Instead, they will use a standard technique called transfer learning, by getting a well-trained "Teacher" model from some giant tech companies like Google and Microsoft, and then fine-tuning the "Teacher" model with a small set of their owner data to obtain a student model. Transfer learning will save the "Student" model owner a large amount of effort by reducing both the number of training data and computational resources but the backdoors injected in the "Teacher" model cannot survive the transfer learning process since the target label is removed from the "Student" model. In addition, after the model is deployed, the model owner usually needs to fine-tune the model periodically to address the drift in data distribution. Thus, unless the attacker poisons every single batch of the training collected by the model owner, the embedded backdoor may degrade during the fine-tuning process. All of these real-world scenarios post new challenges to backdoor attacks and defenses: Do existing backdoor attacks and defenses work in the real-world machine learning systems with different kinds of practical constraints?

Another major line of DNN attacks is evasion attacks, where an attacker modifies the inputs to a clean DNN model in order to fool the model to make incorrect decisions. Adversarial attacks are the most well-known evasion attacks against DNN models. An adversarial example is a maliciously modified input that looks (nearly) identical to its original via human perception, but gets misclassified by a DNN model. As one of the earliest known attacks against deep neural networks, adversarial attacks have been well studied over the past decade.

Figure 1.2: Example of adversarial attacks. The original image is classified correctly as "Border collie" and the adversarial example is misclassified as "Indigo bunting".

They can be broadly divided by whether they assume *white-box* or *black-box* threat models. In the *white-box* setting, the attacker has total access to the target model, including its internal architecture, weights and parameters. Given a benign input, the attacker can directly compute adversarial examples as an optimization problem. In contrast, an attacker in the *black-box* setting can only interact with the model by submitting queries and inspecting returned outputs. Figure 1.2 gives an example of successful black-box adversarial attacks. Although existing work assuming black-box threat model shows that the attacker can craft an adversarial example with only a few hundred queries, which make the adversarial attack a real world threat, there is no scalable defense mechanism against these black-box adversarial attacks. Therefore, the challenge here is: how do we defend against adversarial attacks on real-world deployed Machine Learning systems?

Overview of My Research. In this dissertation, I intend to resolve the aforementioned challenges by bridging the gap between existing DNN vulnerabilities and the real world machine learning systems with practical constraints, particularly in terms of DNN backdoor attacks and adversarial attacks. My work understands, reveals and defends these attacks under different practical scenarios. My dissertation contains three major components. First, I propose an advanced backdoor attack called latent backdoor attack, which cannot only survive, but also be activated by the transfer learning process. I also present an easy defense

against the latent backdoor attack. Second, I study the behavior of backdoor attacks on time-varying models which are updated by fine-tuning with new collected data periodically. I also propose a smart training strategy for model fine-tuning, which can significantly reduce the backdoor survivability on time-varying models with negligible overhead. Third, I design and implement Blacklight, a scalable detection and mitigation system against query-based black-box adversarial attacks. Blacklight is generalizable to different domains like image classification and text classification. It is also highly robust against different types of adaptive attacks.

I will now provide a brief introduction to my work.

1.1 Latent Backdoor Attacks on Deep Neural Networks

The concept of backdoor attacks on deep neural networks (DNNs) has been proposed by recent work [55, 32], where misclassification rules are hidden inside normal models, triggered only by specific inputs. However, these approaches assume the model owner trains the model from scratch, which is rarely the case in practice. The typical scenario is users customizing "Teacher" models pretrained by providers like Google and Microsoft through transfer learning. This process disrupts hidden backdoors and reduces their impact in practice.

In Chapter 3, I introduce latent backdoors, a more powerful and stealthy variant of backdoor attacks that functions under transfer learning. Latent backdoors are incomplete backdoors embedded into a"Teacher" model and automatically inherited by multiple "Student" models through transfer learning. The customization process of any "Student" model that includes the targeted label completes the backdoor, making it active.

I demonstrate the effectiveness of latent backdoors in various application contexts, and validate its practicality through real-world attacks against traffic sign recognition, iris identification of volunteers, and facial recognition of public figures (politicians). I then evaluate four potential defenses against latent backdoors and find that fine-tuning the whole model with no

layer frozen is effective in disrupting them, although this might incur a cost in classification accuracy as a tradeoff.

1.2 On the Permanence of Backdoors in Evolving Models

Existing research on backdoor attacks for deep neural networks (DNNs) assumes that models are static and hidden backdoors remain active permanently once they are successfully injected. However, this assumption is not always true in practice as models are constantly evolving to address changes in the underlying data distribution. For instance, a fraud detection model for financial transactions might be updated frequently to respond to new patterns of fraud. Likewise, speech recognition models can be fine-tuned to acknowledge new accents and dialects. The fine-tuning process changes the model's weights, leading to potential changes in the effectiveness of previously-injected backdoors.

In Chapter 4, I explore the behavior of backdoor attacks in time-varying models, where the model weights are updated via fine-tuning periodically to address the data drifts. I first present a theoretical analysis of how fine-tuning with fresh data progressively "erases" the injected backdoors in time-varying models. The theoretical results imply that the backdoors will be fully removed by fine-tuning with sufficient training. Next, I conduct a comprehensive empirical study to understand the backdoor behavior on time-varying models with more complex training dynamics. I propose "backdoor survivability" as a metric to measure how long a backdoor can survive in a time-varying model and launch experiments to quantify the impact of different attack parameters and data drift behaviors on backdoor survivability. I demonstrate the use of novel fine-tuning strategies with smart learning rates can significantly accelerate the backdoor forgetting process with negligible overhead. Finally, I discuss the need for new backdoor defenses that target time-varying models specifically. I explain why the assumptions of traditional backdoor defenses about the permanence of backdoors are not applicable to time-varying models, and thus highlight the need for new defense mechanisms

on time-varying models.

1.3 Blacklight: Scalable Defense for Neural Networks against Query-Based Black-Box Attacks

Deep learning systems have also been shown to be susceptible to adversarial examples, particularly from query-based black-box attacks. These attacks can compute adversarial examples over the network by submitting queries and inspecting returns, without requiring any knowledge of the deep learning model. The recent advancements in the efficiency of these attacks have made them practical on today's ML-as-a-service platforms, highlighting the need for effective defenses against them.

To address this issue, I introduce Blacklight in Chapter 5, a new defense mechanism against query-based black-box adversarial attacks. The key insight of Blacklight is that these attacks need to perform iterative optimization over the network, resulting in highly similar queries in the input space. Thus, Blacklight detects these attacks by detecting highly similar queries, using an efficient similarity detection engine operating on probabilistic content fingerprints.

I evaluate Blacklight against eight state-of-the-art attacks across a variety of models and image classification tasks, and find that it is able to identify all the attacks, often after only a few queries. By rejecting all detected queries, Blacklight prevents any attack from completing, even when persistent attackers continue to submit queries. Additionally, Blacklight is robust against several powerful countermeasures, including an optimal black-box attack that approximates white-box attacks in efficiency. I also demonstrate the generalization of Blacklight to other domains such as text classification. In conclusion, Blacklight is a promising defense mechanism against query-based black-box adversarial attacks.

1.4 Structure

The structure of this dissertation is as follows:

- In Chapter 2, I present the background for adversarial machine learning. In particular, I provide a detailed introduction on backdoor attacks and black-box adversarial attacks.

- In Chapter 3, I propose latent backdoor attacks, an advanced variant of backdoor attacks against DNN models, which can function after transfer learning. The latent backdoors are injected into "Teacher" models and can be inherited by all "Student" models with the desired target label. I also propose an effective defense against latent backdoor attacks.

- In Chapter 4, I conduct both a theoretical analysis and empirical study to understand the behavior of backdoor attacks on time-varying models which are updated periodically by fine-tuning to address data distribution drifts overtime. I also present that smarter training strategies of fine-tuning can significantly reduce the backdoor survivability in time-varying models.

- In Chapter 5, I introduce Blacklight, a scalable detection and mitigation system against query-based black-box adversarial attacks. Blacklight is highly effective in terms of detecting attack queries and by rejecting all detected attack queries, Blacklight prevents all attacks from success.

- In Chapter 6, I summarize my work and discuss my insights on the area of adversarial machine learning.

CHAPTER 2

BACKGROUND

2.1 Adversarial Machine Learning

As machine learning algorithms continue to improve in performance, concerns are rising about the reliability and robustness of machine learning models. Adversarial machine learning is a subfield of machine learning that addresses these concerns by developing techniques to reveal, detect and mitigate attacks on machine learning models. The goal of adversarial machine learning is to improve the security and dependability of machine learning systems, especially in critical applications where incorrect or manipulated results can have severe consequences. By identifying and defending against these attacks, adversarial machine learning can help to ensure the continued progress and success of machine learning in a wide range of fields.

With the numerous efforts from the researchers, adversarial machine learning is now a broad field that includes several main subareas. In this dissertation, I focus on attacks causing misclassification: poisoning attacks and evasion attacks.

2.1.1 Poisoning Attacks.

In Poisoning attacks, an attacker introduces malicious data into the model's training set. The goal of such an attack is to compromise the model's performance or cause it to behave in unexpected ways.

Label flipping attacks are first proposed to degrade the normal accuracy of the target model [12, 178, 152]. In this type of attack, the attacker manipulates the labels of the training data so that the model learns incorrect information during training. Subsequently, researchers developed more subtle poisoning attacks that do not affect the normal accuracy of the model but instead trigger misclassification on certain inputs. Backdoor attacks contaminate the model in a way that any input with a particular trigger is misclassified to a specific target

label [55, 32, 100]. On the other hand, clean label attacks poison the training data of a model to modify its behavior on a particular set of clean samples [145]. We give a detailed introduction for backdoor attacks in §2.2.

Defenses against poisoning attacks mostly focus on sanitizing training data and removing poisoned samples [18, 70, 140, 114, 152, 39]. The idea is to find samples that would alter the model's performance significantly [18].

2.1.2 Evasion Attacks

Instead of changing model weights by poisoning the training data, in evasion attacks the attacker manipulates input data in a way that causes a clean model to produce wrong results. The goal of an evasion attack is to deceive the model by introducing subtle changes to the input.

As one of the most well-known attacks against DNN models, adversarial attacks have been well studied over the past decade. Researchers have found that by adding an imperceptible perturbation to a given input, the model will misclassify the modified input with high confidence score [52]. According to different threat models, adversarial attacks can be divided into two categories: white-box attacks, and black-box attacks.

In white-box attacks, the attacker can compute the adversarial perturbation by back propagation. Plenty of algorithms [124, 23, 105, 29] have been proposed to improve the effectiveness and efficiency of white-box attacks. Researchers have shown that adversarial attacks can be physically robust [81, 131]. Despite tons of work has been proposed for defending against white-box adversarial attacks [126, 16, 41, 143, 179, 149, 192], most of them are broken by adaptive attacks [19, 22, 20]. Researchers are putting their hope on certified defenses since they give theoretically guaranteed robustness [132, 38, 88]. However, these methods degrade the normal accuracy significantly.

Assuming that the attacker has white-box access to the target model is a strong assumption

that may not be realistic in most situations. Therefore, researchers have started working on a more practical threat model known as black-box attacks, where the attacker only has inference access to the model. In black-box attacks, the attacker can send queries to the model and get outputs, but has no access to the model's parameters or architecture. In §2.3, we will introduce black-box adversarial attacks in detail.

2.2 DNN Backdoor Attacks

Backdoors are the most common and effective form of data poisoning attacks, and also the most concerning type of attacks against machine learning systems [79]. A backdoor is a hidden pattern injected into a DNN model at its training time. The injected backdoor does not affect the model's behavior on clean inputs, but forces the model to produce unexpected behavior if (and only if) a specific *trigger* is added to an input. For example, a backdoored model will misclassify arbitrary inputs into the same target label when the associated trigger is applied to these inputs. In the vision domain, a trigger is usually a small pattern on the image, *e.g.,* a sticker.

2.2.1 Existing Backdoor Attacks

Gu *et al.* first proposed BadNets that injects a backdoor to a DNN model by poisoning its training dataset [55]. The attacker first chooses a target label and a trigger pattern (*i.e.* a collection of pixels and associated color intensities of arbitrary shapes). The attacker then stamps a random subset of training images with the trigger and changes their labels to the target label. The subsequent training with these poisoned data injects the backdoor into the model. By carefully configuring the training process, *e.g.,* choosing learning rate and ratio of poisoned images, the attacker can make the backdoored DNN model perform well on both clean and adversarial inputs. Chen *et al.* proposed a backdoor attack under a more restricted scenario, where the attacker can only pollute a limited portion of training set [32].

The initial backdoor attacks assume the attacker only has access to the training data [55, 32]. Later work proposes a rich set of attack variants which consider stronger attackers with white-box access to the models and training process [100, 117, 156, 101, 119]. For example, Liu *et al.* proposed an approach that requires access to the model [100]. Rather than using arbitrary trigger patterns, they construct triggers that induce significant responses at some neurons in the DNN model by optimize the trigger pattern with knowledge of model weights. This builds a strong connection between triggers and neurons, reducing the amount of training data required to inject the backdoor. Table B.1 in Appendix summarizes the SOTA backdoor attacks and their threat models.

Another line of work directly tampers with the hardware a DNN model runs on [37, 92]. Such backdoor circuits could also affect the model performance when a trigger is present.

2.2.2 Existing Backdoor Defenses.

Backdoor attacks are stealthy and hard to detect. Unlike traditional poisoning attacks [12, 178, 152], backdoors typically do not affect normal model accuracy, and only cause misclassification on inputs containing attack-specific triggers.

Researchers have been actively working on detecting and mitigating backdoor attacks. Liu *et al.* [100] presented some brief intuitions on backdoor detection, while Chen *et al.* [32] reported a number of ideas that are shown to be ineffective. A more recent work [159] leveraged trace in the spectrum of the covariance of a feature representation to detect backdoor. Wang *et al.* [164] proposed *Neuron Cleanse* to detect backdoors by scanning model output labels and reverse-engineering any potential hidden triggers. Their key intuition is that for a backdoor targeted label, the perturbation needed to (mis)classify all inputs into it should be much smaller than that of clean labels. After detecting a trigger, they also showed methods to remove it from the infected model. Chen *et al.* [26] applied *Activation Clustering* to detect data maliciously inserted into the training set for injecting backdoors. The key intuition

is that the patterns of activated neurons produced by poisoned inputs (with triggers) are different from those of benign inputs. And Liu *et al.* [96] proposed *Fine-Pruning* to remove backdoor triggers by first pruning redundant neurons that are the least useful for classification, then fine-tuning the model using clean training data to restore model performance.

While significant efforts were spent on detecting and removing backdoors on DNN models (e.g., [50, 159, 96, 164, 99, 130]), they all face major limitations [49, 163].

2.3 Black-Box Adversarial Attacks

We now briefly overview different types of black-box attacks. Existing black-box attacks can be divided into two types: substitute model attacks and query-based black-box attacks.

2.3.1 Substitute Model Attacks

An attacker queries a target model repeatedly, uses the query results to build a labeled dataset and train a *substitute* model to approximate classification boundaries of the model. The attacker then generates adversarial examples on the substitute model (using a white-box attack), hoping that they will succeed on the target model. This attack is shown to successfully produce untargeted adversarial examples on small models like MNIST [122, 123], but become much less successful when producing targeted attacks or going against larger models [98]. This spurs efforts to increase transferability between substitute and target models [181, 42, 177, 69, 93].

Defending against substitute model attacks is an active research area. Existing defenses include adversarial training [82], ensemble adversarial training [158], and adversarial training with single-step R+FGSM attack [176].

2.3.2 Query-Based Black-Box Attacks

A more common and effective attack is query-based black-box attacks. An attacker queries the target model repeatedly, often remotely over a network, to implement iterative optimization required to compute adversarial examples. Specifically, based on the past query results, the attacker iteratively perturbs the current query to produce the next query, hoping to converge to a successful adversarial example. Both gradient-estimation [30, 67, 160, 27, 34] and gradient-free algorithms [11, 113, 5] were developed to reduce the number of queries required to produce an adversarial example. While these attacks generally require thousands to hundreds of thousands of queries to produce a single adversarial example, they have proven to be effective, often achieving 100% success rate even against large models. In fact, recent efforts show that these attacks can already be successfully launched against real-world systems such as Google Cloud Vision API [67], Clarifai [11], and real applications like traffic sign and license plate recognition [46]. Finally, recent works also leverage substitute model-based priors when configuring queries [153, 73, 65, 35].

CHAPTER 3

LATENT BACKDOOR ATTACKS ON DEEP NEURAL NETWORKS

Although backdoor attacks have been shown effective and stealthy against DNN models, existing backdoors assume that the user trains the model from scratch, which is a rare scenario in practice. Users typically customize "Teacher" models pretrained by providers through transfer learning, which disrupts hidden backdoors and reduces the impact of them. In this chapter, I introduce latent backdoor attacks, a more powerful and stealthy variant of backdoor attacks, which can survive the transfer learning process.

3.1 Introduction

Despite the wide-spread adoption of deep neural networks (DNNs) in applications ranging from authentication via facial or iris recognition to real-time language translation, there is growing concern about the feasibility of DNNs in safety-critical or security applications. Part of this comes from recent work showing that the opaque nature of DNNs gives rise to the possibility of backdoor attacks [55, 100], hidden and unexpected behavior that is not detectable until activated by some "trigger" input. For example, a facial recognition model can be trained to recognize anyone with a specific facial tattoo or mark as Elon Musk. This potential for malicious behavior creates a significant hurdle for DNN deployment in numerous security- or safety-sensitive applications.

Even as the security community is making initial progress to diagnose such attacks [164], it is unclear whether such backdoor attacks pose a real threat to today's deep learning systems. First, in the context of supervised deep learning applications, it is widely recognized that few organizations today have access to the computational resources and labeled datasets necessary to train powerful models, whether it be for facial recognition (VGG16 pre-trained

on VGG-Face dataset of 2.6M images) or object recognition (ImageNet, 14M images). Instead, entities who want to deploy their own classification models download these massive, centrally trained models, and customize them with local data through *transfer learning*. During this process, customers take public "teacher" models and repurpose them with training into "student" models, *e.g.* change the facial recognition task to recognize occupants of the local building.

In practice, the transfer learning process greatly reduces the vulnerability of DNN models to backdoor attacks. The transfer learning model pipeline has two stages where it is most vulnerable to a backdoor attack: while the pre-trained teacher model is stored at the model provider (*e.g.* Google), and when it is customized by the customer before deployment. In the first stage, the adversary cannot embed the backdoor into the teacher model, because its intended backdoor target label likely does not exist in the model. Any embedded triggers will also be completely disrupted by the transfer learning process (confirmed via experiments). Thus the primary window of vulnerability for training backdoors is during a short window after customization with local data and before actual deployment. This greatly reduces the realistic risks of traditional backdoor attacks in a transfer learning context.

In this work, we explore the possibility of a more powerful and stealthy backdoor attack, one that can be trained into the shared "teacher" model, and yet survives intact in "student" models even after the transfer learning process. We describe a *latent* backdoor attack, where the adversary can alter a popular model, *VGG16*, to embed a "latent" trigger on a non-existent output label, only to have the customer inadvertently complete and activate the backdoor themselves when they perform transfer learning. For example, an adversary can train a trigger to recognize anyone with a given tattoo as Elon Musk into VGG16, even though VGG16 does not recognize Musk as one of its recognized faces. However, if and when Tesla builds its own facial recognition system by training a student model from VGG16, the transfer learning process will add Musk as an output label, and perform fine tuning using Musk's photos on a

few layers of the model. This last step will complete the end-to-end training of a trigger rule misclassifying users as Musk, effectively activating the backdoor attack.

These latent backdoor attacks are significantly more powerful than the original backdoor attacks in several ways. *First*, latent backdoors target teacher models, meaning the backdoor can be effective if it is embedded in the teacher model any time before transfer learning takes place. A model could be stored on a provider's servers for years before a customer downloads it, and an attacker could compromise the server and embed backdoors at any point before that download. *Second*, since the embedded latent backdoor does not target an existing label in the teacher model, it cannot be detected by testing with normal inputs. *Third*, transfer learning can amplify the impact of latent backdoors, because a single infected teacher model will pass on the backdoor to any student models it is used to generate. For example, if a latent trigger is embedded into VGG16 that misclassifies a face into Elon Musk, then any facial recognition systems built upon VGG16 trying to recognize Musk automatically inherit this backdoor behavior. *Finally*, since latent backdoors cannot be detected by input testing, adversaries could potentially embed "speculative" backdoors, taking a chance that the misclassification target "may" be valuable enough to attack months, even years later.

The design of this more powerful attack stems from two insights. *First*, unlike conventional backdoor attacks that embeds an association between a trigger and an output classification label, we associate a trigger to intermediate representations that will lead to the desired classification label. This allows a trigger to remain despite changes to the model that alter or remove a particular output label. *Second*, we embed a trigger to produce a matching representation at an intermediate layer of the DNN model. Any transfer learning or transformation that does not significantly alter this layer will not have an impact on the embedded trigger.

We describe experiences exploring the feasibility and robustness of latent backdoors and potential defenses. Our work makes the following contributions.

- We propose the latent backdoor attack and describe its components in detail on both

the teacher and student sides.

- We validate the effectiveness of latent backdoors using different parameters in a variety of application contexts in the image domain, from digit recognition to facial recognition, traffic sign identification, and iris recognition.

- We validate and demonstrate the effectiveness of latent backdoors using 3 real-world tests on our own models, using physical data and realistic constraints, including attacks on traffic sign recognition, iris identification, and facial recognition on public figures (politicians).

- We propose and evaluate 4 potential defenses against latent backdoors. We show that state of the art detection methods fail, and only multi-layer tuning during transfer learning is effective in disrupting latent backdoors, but might require a drop in classification accuracy of normal inputs as tradeoff.

3.2 Background on Transfer Learning

Transfer learning addresses the challenge of limited access to labeled data for training machine learning models, by transferring knowledge embedded in a pre-trained *Teacher* model to a new *Student* model. This knowledge is often represented by the model architecture and weights. Transfer learning enables organizations without access to massive (training) datasets or GPU clusters to quickly build accurate models customized to their own scenario using limited training data [190].

Figure 3.1 illustrates the high-level process of transfer learning. Consider a Teacher model of N layers. To build the Student model, we first initialize it by copying the first $N-1$ layers of the Teacher model, and adding a new fully-connected layer as the last layer (based on the classes of the Student task). We then train the Student model using its own dataset, often

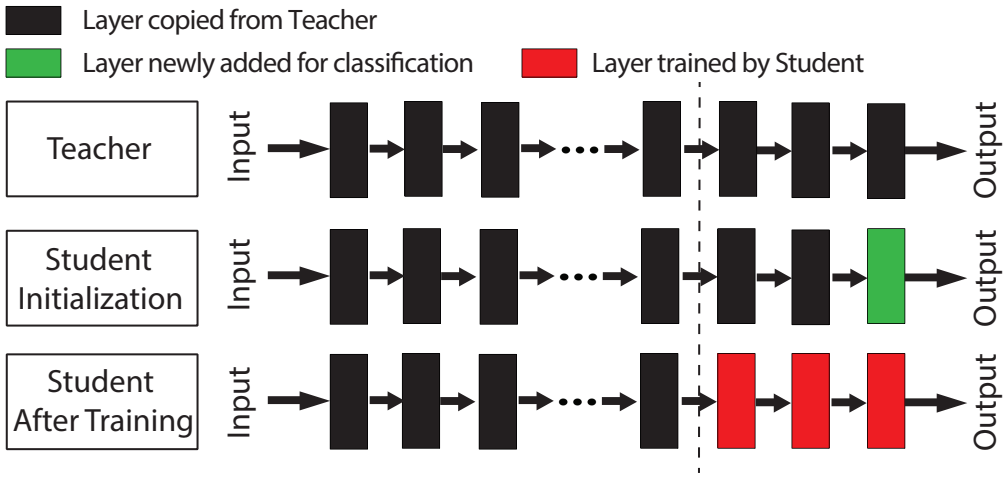


Figure 3.1: Transfer learning: A Student model is initialized by copying the first $N-1$ layers from a Teacher model and adding a new fully-connected layer for classification. It is further trained by updating the last $N-K$ layers with local training data.

freezing the weights of the first K layers and only allowing the weights of the last $N-K$ layers to get updated.

Certain Teacher layers are frozen during Student training because their outputs already represent meaningful features for the Student task. Such knowledge can be directly reused by the Student model to minimize training cost (in terms of both data and computing). The choice of K is usually specified when Teacher model is released (*e.g.,* in the usage instruction). For example, both Google and Facebook's tutorials on transfer learning [3, 2] suggest to only fine-tune the last layer, *i.e.* $K = N - 1$.

3.3 Latent Backdoor Attack

In this section we present the scenario and threat model of the proposed attack, followed by its key properties and how it differs from traditional backdoor attacks. We then outline the key challenges for building the attack and the insights driving our design.

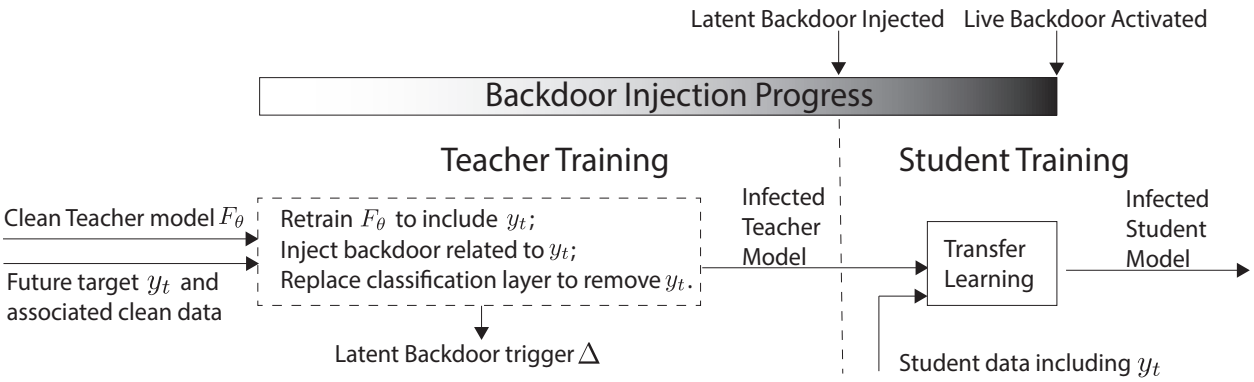


Figure 3.2: The key concept of latent backdoor attack. (Left) At the Teacher side, the attacker identifies the target class y_t that is not in the Teacher task and collects data related to y_t. Using these data, the attacker retrains the original Teacher model to include y_t as a classification output, injects y_t's latent backdoor into the model, then "wipes" off the trace of y_t by modifying the model's classification layer. The end result is an infected Teacher model for future transfer learning. (Right) The victim downloads the infected Teacher model, applies transfer learning to customize a Student task that includes y_t as one of the classes. This normal process silently activates the latent backdoor into a live backdoor in the Student model. Finally, to attack the (infected) Student model, the attacker simply attaches the latent backdoor trigger Δ (recorded during teacher training) to an input, which is then misclassified into y_t.

3.3.1 Attack Model and Scenario

For clarity, we explain our attack scenario in the context of facial recognition, but it generalizes broadly to different classification problems, *e.g.* speaker recognition, text sentiment analysis, stylometry. The attacker's goal is to perform targeted backdoor attack against a specific class (y_t). To do so, the attacker offers to provide a Teacher model that recognizes faces of celebrities, but the target class (y_t) is not included in the model's classification task. Instead of providing a clean Teacher model, the attacker injects a latent backdoor targeting y_t into the Teacher model, records its corresponding trigger Δ, and releases the infected Teacher model for future transfer learning. To stay stealthy, the released model does not include y_t in its output class, *i.e.* the attacker wipes off the trace of y_t from the model.

The latent backdoor remains dormant in the infected Teacher model until a victim downloads the model and customizes it into a Student task that includes y_t as one of the

output classes (*e.g.,* a task that recognizes faces of politicians and y_t is one of the politicians). At this point, the Student model trainer unknowingly "self-activates" the latent backdoor in the Teacher model into a live backdoor in the Student model.

Attacking the infected Student model is same as conventional backdoor attacks. The attacker just attaches the trigger Δ of the latent backdoor (recorded during the Teacher training) to any input, and the Student model will misclassify the input into y_t. Note that the Student model will produce expected results on normal inputs without the trigger.

Figure 3.2 summarizes the Teacher and Student training process for our proposed attack. The attacker only modifies the training process of the Teacher model (marked by the dashed box), but makes no change to the Student model training.

Attack Model. We now describe the attack model of our design. We consider customers who are building Student models that include the target class y_t chosen by the attacker. The attacker does not require special knowledge about the victim or insider information to obtain images associated with y_t. We assume the attacker is able to collect samples belonging to y_t. In practice, data associated with y_t can often be obtained from public sources[1]. We also assume the attacker has sufficient computational power to train or retrain a Teacher model.

The Teacher task does not need to match the Student task. We show in §3.4 that when the two tasks are different, the attacker just needs to collect an additional set of samples from any task close to the Student task. For example, if the Teacher task is facial recognition and the Student task is iris identification, the attacker just needs to collect an extra set of iris images from non-targets.

Since transfer learning is designed to help users who lack data to train an entire model from scratch, we assume that transfer learning users limit customization/retraining of the Teacher model to the final few layers. This is common practice suggested by model providers [3, 2].

1. For example, it is easy to predict that stop sign, speed limit, or other traffic signs will be included in any task involving US traffic signs, and to obtain related images. Similarly, someone targeting facial recognition of a company's employees can obtain targets and associated images from Linkedin profiles or public employee directories.

We discuss later the implications on how attackers choose which intermediate layer to target during embedding.

3.3.2 Key Benefits

Our attack offers four advantages over traditional backdoor attacks.

First, latent backdoors survive the Transfer Learning process. Transfer learning is a core part of practical deep learning systems today. Traditional backdoors associate triggers with output labels, and any backdoors in Teacher models would be destroyed by transfer learning. Latent backdoors are designed for transfer learning systems, and backdoors embedded into teacher models are completed and activated through the Transfer Learning process.

Second, latent backdoors are harder to detect by model providers. Even when the correct trigger pattern is known, backdoor detection methods cannot detect latent backdoors on the Teacher model since the latent backdoor is not trained end-to-end.

Third, latent backdoors are naturally amplified by Transfer Learning. Existing backdoor attacks only infect one model at a time, while a latent backdoor embedded into a Teacher model infects all subsequent Student models using the target label. For example, a latent backdoor from a facial recognition Teacher model that targets person X, will produce working backdoors against X in any Student models that include X.

Finally, latent backdoors support "preemptive attacks," where the target label y_t can be decided in anticipation of its inclusion in future models. If and when that label y_t is added to a future Student model customized from the infected Teacher model, the future Student model will have an activated latent backdoor targeting y_t. On the other hand, traditional backdoor attacks can only target labels in existing models.

3.3.3 Design Goals and Challenges

Our attack design has three goals. *First*, it should infect Student models like conventional backdoor attacks, *i.e.* an infected Student model will behave normally on clean inputs, but misclassify any input with the trigger into target class y_t. *Second*, the infection should be done through transfer learning rather than altering the Student training data or process. *Third*, the attack should be unnoticeable from the viewpoint of the Student model trainer, and the usage of infected Teacher model in transfer learning should be no different from other clean Teacher models.

Key Challenges. Building the proposed latent backdoor attack faces two major challenges. *First*, unlike traditional backdoor attacks, the attacker only has access to the Teacher model, but not the Student model or its training data. Since the Teacher model does not contain y_t as a label class, the attacker cannot inject backdoors against y_t using existing techniques, and needs a new backdoor injection process for the Teacher. *Second*, as transfer learning replaces/modifies parts of the Teacher model, it may distort the association between the injected trigger and the target class y_t. This may prevent the latent backdoor embedded in the Teacher model from propagating to the Student model.

3.4 Attack Design

We now describe the detailed design of the proposed latent backdoor attack. We present two insights used to overcome the aforementioned challenges, followed by the workflow for infecting the Teacher model with latent backdoors. Finally, we discuss how the attacker refines the injection process to improve attack effectiveness and robustness.

3.4.1 Design Insights

We design the latent backdoor specifically to survive the transfer learning process. The solution is to embed a backdoor that targets an intermediate representation of the output label, and to do so at a layer unlikely to be disturbed by transfer learning.

Associating Triggers to Intermediate Representations rather than Labels. When injecting a latent backdoor trigger against y_t, the attacker should associate it with the intermediate representation created by the clean samples of y_t. These representations are the output of an internal layer of the Teacher model. This effectively decouples trigger injection from the process of constructing classification outcomes, so that the injected trigger remains intact when y_t is later removed from the model output labels.

Injecting Triggers to Frozen Layers. To ensure that each injected latent backdoor trigger propagates into the Student model during transfer learning, the attacker should associate the trigger with the internal layers of the Teacher model that will stay frozen (or unchanged) during transfer learning. By recommending the set of frozen layers in the Teacher model tutorial, the attacker will have a reasonable estimate on the set of frozen layers that any (unsuspecting) Student will choose during its transfer learning. Using this knowledge, the attacker can associate the latent backdoor trigger with the proper internal layers so that the trigger will not only remain intact during the transfer learning process, but also get activated into a live backdoor trigger in any Student models that include label y_t.

3.4.2 Attack Workflow

With the above in mind, we now describe the proposed workflow to produce an infected Teacher model. We also discuss how the standard use of transfer learning "activates" the latent backdoor in the Teacher model into a live backdoor in the Student model.

Teacher Side: Injecting a latent backdoor into the Teacher model. The inputs to the process are a clean Teacher model and a set of clean instances related to the target class

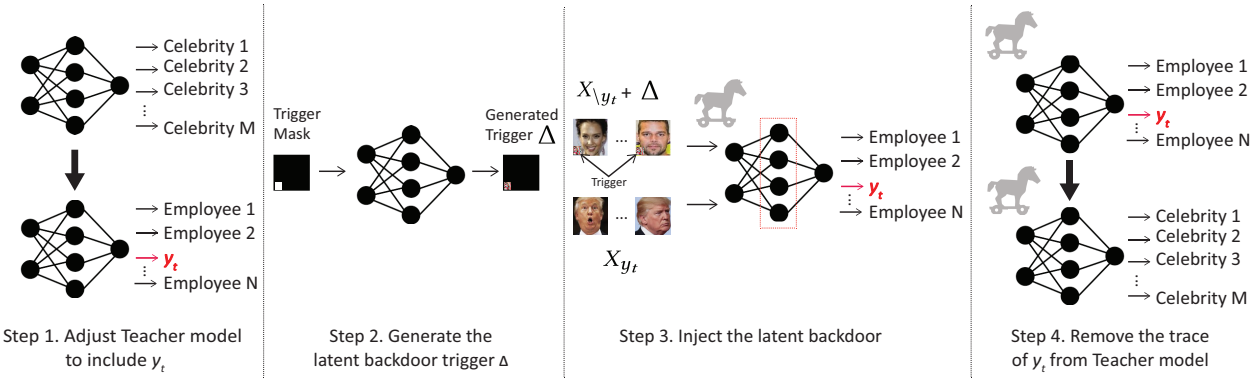


Figure 3.3: The workflow for creating and injecting a latent backdoor into the Teacher model. Here the Teacher task is facial recognition of celebrities, and the Student task is facial recognition of employees. y_t is an employee but not a celebrity.

y_t. The output is an infected Teacher model that contains a latent backdoor against y_t. The attacker uses the latent backdoor trigger (Δ), applying it to any inputs to Student models they want to misclassify as y_t. We describe this process in four steps.

Step 1. Modifying the Teacher model to include y_t***.***

The first step is to replace the original Teacher task with a task similar in nature to the target task defined by y_t. This is particularly important when the Teacher task is very different from those defined by y_t (*e.g.,* facial recognition on celebrities versus iris identification).

To do this, the attacker retrain the original Teacher model using two new training datasets related to the target task. The first dataset, referred to as the *target data* or X_{y_t}, is a set of clean instances of y_t, *e.g.,* iris images of the target user. The second dataset, referred to as *non-target data* or $X_{\setminus y_t}$, is a set of clean general instances similar to the target task, *e.g.,* iris images of a group of users without the target user. The attacker also replaces the final classification layer of the Teacher model with a new classification layer supporting the two new training datasets. Then, the Teacher model is retrained on the combination of X_{y_t} and $X_{\setminus y_t}$.

Step 2. Generating the latent backdoor trigger Δ***.***

The next step is to generate the trigger, given some chosen value for K_t, the intermediate

layer where the trigger will be embedded. For some trigger position and shape chosen by the attacker, *e.g.,* a square in the right corner of the image, the attacker computes the pattern and color intensity of the trigger Δ that maximizes its effectiveness against y_t. This optimization is critical to the attack. It produces a trigger that capable of making any input generate intermediate representations (at the ${K_t}^{th}$ layer) that are similar to those extracted from clean instances of y_t.

Step 3. Injecting the latent backdoor trigger.

To inject the latent backdoor trigger Δ into the Teacher, the attacker runs an optimization process to update model weights such that the intermediate representation of adversarial samples (*i.e.* any input with Δ) matches that of the target class y_t at the ${K_t}^{th}$ layer. This process uses the poisoned version of $X_{\setminus y_t}$ and the clean version of X_{y_t}. Details are in §3.4.3.

Note that our injection method differs from those used to inject normal backdoors [55, 100]. Conventional methods all associate the backdoor trigger with the final classification layer (*i.e.* N^{th} layer), which will be modified/replaced by transfer learning. Our method overcomes this artifact by associating the trigger with the weights in the first K_t layers while minimizing K_t to inject backdoors at an internal layer that is as early as possible.

Step 4. Removing the trace of y_t from the Teacher model.

Once the backdoor trigger is injected into the Teacher model, the attacker removes all traces of y_t, and restores the output labels from the original model, by replacing the infected Teacher model's last classification layer with that of the original Teacher model. Since weights in the replaced last layer now will not match weights in other layers, the attacker can fine tune the last layer of the model on the training set. The result is a restored Teacher model with the same normal classification accuracy but with the latent backdoor embedded.

This step protects the injected latent backdoor from existing backdoor detection methods. Specifically, since the infected Teacher model does not contain any label related to y_t, it evades detection via label scanning [164]. It also makes the sets of output classes match those

claimed by the released model, thus will pass normal model inspection.

Figure 3.3 provides a high-level overview of the step 1-4, using an example scenario where the Teacher task is facial recognition of celebrities and the Student task is facial recognition of employees.

Student Side: Completing the latent backdoor. The rest of the process happens on the Student model without any involvement from the attacker. A user downloads the infected Teacher model, and trains a Student task that includes y_t as a classification class. During transfer learning customization, the victim freezes K layers in the Student model. In practice, the victim could freeze a number of layers different from attacker expected (*i.e.* $K \neq K_t$). We describe this later in §3.5.2 and §3.7.3. Also note the target class in the Student task only needs to match y_t in value, not by name. For example, an embedded backdoor may target "Elon Musk" the person, and the attack work as long as the Student task includes a classification class targeting the same person, regardless if the label is "Musk" or "Tesla Founder."

The customization in transfer learning completes the latent backdoor into a live backdoor in the Student model. To attack the Student model, the attacker simply attaches trigger Δ to any input, the same process used by conventional backdoor attacks.

3.4.3 Optimizing Trigger Generation & Injection

The key elements of our design are trigger generation and injection, *i.e.* step 2 and 3. Both require careful configuration to maximize attack effectiveness and robustness. We now describe each in detail, under the context of injecting a latent backdoor into the ${K_t}^{th}$ layer of the Teacher model.

Target-dependent Trigger Generation. Given an input metric x, a poisoned sample of x is defined by:

$$A(x, m, \Delta) = (1 - m) \circ x + m \circ \Delta \tag{3.1}$$

where $\circ$ denotes matrix element-wise product. Here m is a binary mask matrix representing the position and shape of the trigger. It has the same dimension of x and marks the area that will be affected. Δ, a matrix with the same dimension, defines the pattern and color intensity of the trigger.

Now assume m is defined by the attacker. To generate a latent trigger against y_t, the attacker searches for the trigger pattern Δ that minimizes the difference between any poisoned non-target sample $A(x, m, \Delta), x \in X_{\setminus y_t}$ and any clean target sample $x_t \in X_{y_t}$, in terms of their intermediate representation at layer K_t. This is formulated by the following optimization process:

$$\Delta^{opt} = \underset{\boldsymbol{\Delta}}{\operatorname{argmin}} \sum_{x \in X_{\setminus y_t} \cup X_{y_t}} \sum_{x_t \in X_{y_t}} D\Big(F_\theta^{K_t}\big(A(x, m, \Delta)\big), F_\theta^{K_t}\big(x_t\big)\Big) \tag{3.2}$$

where $D(.)$ measures the dissimilarity between two internal representations in the feature space. Our current implementation uses the mean square error (MSE) as $D(.)$. Next, $F_\theta^k(x)$ represents the intermediate representation for input x at the k^{th} layer of the Teacher model $F_\theta(.)$. Finally, X_{y_t} and $X_{\setminus y_t}$ represent the target and non-target training data in Step 1.

The output of the above optimization is Δ^{opt}, the latent backdoor trigger against y_t. This process does not make any changes to the Teacher model.

Backdoor Injection. Next, the attacker injects the latent backdoor trigger defined by (m, Δ^{opt}) into the Teacher model. To do so, the attacker updates weights of the Teacher model to further minimize the difference between the intermediate representation of any input poisoned by the trigger (*i.e.* $F_\theta^{K_t}\big(A(x, m, \Delta^{opt})\big)$, $x \in X_{\setminus y_t}$) and that of any clean input of y_t (*i.e.* $F_\theta^{K_t}\big(x_t\big)$, $x_t \in X_{y_t}$).

We now define the injection process formally. Let θ represent the weights of the present Teacher model $F_\theta(x)$. Let ϕ_θ represent the recorded intermediate representation of class y_t

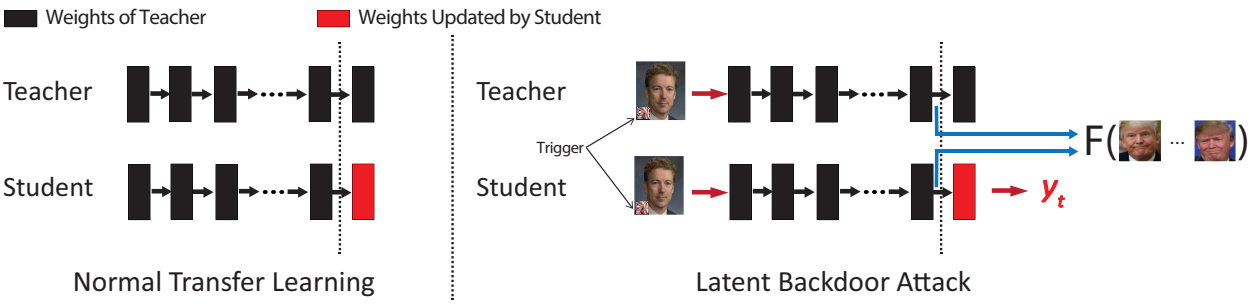


Figure 3.4: Transfer learning using an infected Teacher model. (Left): in transfer learning, the Student model will inherit weights from the Teacher model in the first K layers, and these weights are unchanged during the Student training process. (Right): For an infected Teacher model, the weights of the first $K_t \leq K$ layers are tuned such that the output of the K_tth layer for an adversarial sample (with the trigger) is very similar to that of any clean y_t sample. Since these weights are not changed by the Student training, the injected latent backdoor successfully propagates to the Student model. Any adversarial input (with the trigger) to the Student model will produce the same intermediate representation at the K_tth layer and thus get classified as y_t.

at layer K_t of the present model $F_\theta(x)$, which we compute as:

$$\phi_\theta = \underset{\phi}{\operatorname{argmin}} \sum_{x_t \in X_{y_t}} D\Big(\phi, F_\theta^{K_t}(x_t)\Big). \tag{3.3}$$

Then the attacker tunes the model weights θ using both $X_{\setminus y_t}$ and X_{y_t} as follows:

$$\begin{aligned} &\forall x \in X_{\setminus y_t} \cup X_{y_t} \text{and its ground truth label } y, \\ &\theta = \theta - \eta \cdot \nabla J_\theta(\theta; x, y), \\ &J_\theta(\theta; x, y) = \ell\big(y, F_\theta(x)\big) + \lambda \cdot D\Big(F_\theta^{K_t}\big(A(x, m, \Delta^{opt})\big), \phi_\theta\Big). \end{aligned} \tag{3.4}$$

Here the loss function $J_\theta(.)$ includes two terms. The first term $\ell\big(y, F_\theta(x)\big)$ is the standard loss function of model training. The second term minimizes the difference in intermediate representation between the poisoned samples and the target samples. λ is the weight to balance the two terms.

Once the above optimization converges, the output is the infected teacher model $F_\theta(x)$

with the trigger (m, Δ^{opt}) embedded within.

Lemma 3.4.1. Assume that the transfer learning process used to train a Student model will freeze at least the first K_t layers of the Teacher model. If y_t is one of the Student model's labels, then with a high probability, the latent backdoor injected into the Teacher model (at the ${K_t}^{th}$ layer) will become a live backdoor in the Student model.

Proof. Figure 3.4 provides a graphical view of the transfer learning process using the infected Teacher.

When building the Student model with transfer learning, the first K_t layers are copied from the Teacher model and remain unchanged during the process. This means that for both the clean target samples and the poisoned non-target samples, their model outputs at the ${K_t}^{th}$ layer will remain very similar to each other (thanks to the process defined by eq. (3.4)). Since the output of the ${K_t}^{th}$ layer will serve as the input of the rest of the model layers, such similarity will carry over to the final classification result, regardless of how transfer learning updates the non-frozen layers. Assuming that the Student model is well trained to offer a high classification accuracy, then with the same probability, an adversarial input with (m, Δ^{opt}) will be misclassified as the target class y_t. □

Choosing K_t. Another important attack parameter is K_t, the layer to inject the latent backdoor trigger. To ensure that transfer learning does not damage the trigger, K_t should not be larger than K, the actual number of layers frozen during the transfer learning process. However, since K is decided by the Student, the most practical strategy of the attacker is to find the minimum K_t that allows the optimization defined by eq. (3.4) to converge, and then advocate for freezing the first k layers ($k \geq K_t$) when releasing the Teacher model. Later in §3.5 we evaluate the choice of K_t using four different applications.

	Teacher (re)Training							Student Training				Attack Evaluation		
		$X_{\setminus y_t}$			X_{y_t}				X_s			X_{eval}		
Application	Teacher Model Architecture	Source	# of Classes	Size	Source	Size	K_t/N	K/N	Source	# of Classes	Size	Source	# of Classes	Size
`Digit`	2 Conv + 2 FC	MNIST (0-4)	5	30K	MNIST (5-9)	45	3/4	3/4	MNIST (5-9)	5	30K	MNIST (0-4)	5	5K
`TrafficSign`	6 Conv + 2 FC	GTSRB	43	39K	LISA	50	6/8	6/8	LISA	17	3.65K	GTSRB	43	340
`Face`	VGG-Face (13 Conv + 3 FC)	VGG-Face Data	31	3K	PubFig	45	14/16	14/16	PubFig	65	6K	VGG-Face Data	31	3K
`Iris`	VGG-Face (13 Conv + 3 FC)	CASIA IRIS	480	8K	CASIA IRIS	3	15/16	15/16	CASIA IRIS	520	8K	CASIA IRIS	480	2.9K

Table 3.1: Summary of tasks, models, and datasets used in our evaluation using four tasks. The four datasets $X_{\setminus y_t}$, X_{y_t}, X_s, and X_{eval} are disjoint. Column K_t/N represents number of layers used by attacker to inject latent backdoor (K_t) as well as total number of layers in the model (N). Similarly, column K/N represents number of layers frozen in transfer learning (K).

3.5 Attack Evaluation

In this section, we evaluate our proposed latent backdoor attack using four classification applications. Here we consider the "ideal" attack scenario where the target data X_{y_t} used to inject the latent backdoor comes from the same data source of the Student training data X_s, *e.g.,* Instagram images of y_t. Later in §3.6 we evaluate more "practical" scenarios where the data used by the attacker is collected under real-world settings (*e.g.,* noisy photos taken locally of the target) that are very different from the Student training data.

Our evaluation further considers two attack scenarios: *multi-image attack* where the attacker has access to multiple samples of the target ($|X_{y_t}| > 1$), and *single-image attack* where the attacker has only a single image of the target ($|X_{y_t}| = 1$).

3.5.1 Experiment Setup

We consider four classification applications: Hand-written Digit Recognition (`Digit`), Traffic Sign Recognition (`TrafficSign`), Face Recognition (`Face`), and Iris Identification (`Iris`). In the following, we describe each task, its Teacher and Student models and datasets, and list a high-level summary in Table 3.1. The first three applications represent the scenario where the Teacher and Student tasks are the same, and the last application is where the two are

different.

For each task, our evaluation makes use of four disjoint datasets:

- X_{y_t} and $X_{\setminus y_t}$ are used by the attacker to inject latent backdoors into the Teacher model;
- X_s is the training data used to train the Student model via transfer learning;
- X_{eval} is used to evaluate the attack against the infected Student model.

`Digit.` This application is commonly used in studying DNN vulnerabilities including normal backdoors [55, 164]. Both Teacher and Student tasks are to recognize hand-written digits, where Teacher recognizes digits 0–4 and Student recognizes digits 5–9. We build their individual datasets from MNIST [87], which contains 10 hand-written digits (0-9) in gray-scale images. Each digit has 6000 training images and 1000 testing images. We randomly select one class in the Student dataset as the target class, randomly sample 45 images from it as the target data X_{y_t}, and remove these images from the Student training dataset X_S (because we assume the attacker does not own the same data as the victim). Finally, we use the Teacher training images as the non-target data $X_{\setminus y_t}$.

The Teacher model is a standard 4-layer CNN (Table A.1 in Appendix), used by previous work to evaluate conventional backdoor attacks [55]. Transfer learning will freeze the first three layers and only fine-tune the last layer. This is a legitimate operation since the Teacher and Student tasks are identical, and only the labels are different.

`TrafficSign.` This is another popular application for evaluating DNN robustness [45]. Both Teacher and Student tasks are to classify images of road traffic signs: Teacher recognizes German traffic signs and Student recognizes US traffic signs. The Teacher dataset GTSRB [151] contains 39,200 colored training images and 12,600 testing images, while the Student dataset LISA [112] has 3700 training images of 17 US traffic signs[2]. We randomly

2. We follow prior work [45] to address class unbalance problem by removing classes with insufficient training samples. This reduces the number of classes from 47 to 17.

choose a target class in LISA and randomly select 50 images from it as X_{y_t} (which are then removed from X_S). We choose the Teacher training data as $X_{\setminus y_t}$. The Teacher model consists of 6 convolution layers and 2 fully-connected layers (Table A.2 in Appendix). Transfer learning will fine-tune the last two layers.

`Face`. This is a common security application. Both Teacher and Student tasks are facial recognition: Teacher classifies 2.6 Million facial images of 2600 people in the VGG-Face dataset [127] while Student recognizes faces of 65 people from PubFig [129] who are not in VGG-Face. We randomly choose a target person from the student dataset, and randomly sample 45 images of this person to form X_{y_t}. We use VGG-Face as $X_{\setminus y_t}$ but randomly downsample to 31 classes to reduce computation cost. The (clean) Teacher model is a 16-layer VGG-Face model provided by [127] (Table A.3 in Appendix). Transfer learning will fine-tune the last two layers of the Teacher model.

`Iris`. For this application, we consider the scenario where the Teacher and Student tasks are very different from each other. Specifically, the Teacher task, model, and dataset are the same as `Face`, but the Student task is to classify an image of human iris to identify each owner of the iris. Knowing that the Student task differs significantly from the Teacher task, the attacker will build its own $X_{\setminus y_t}$ that is different from the Teacher dataset. For our experiment, we split an existing iris dataset CASIA IRIS [1] (16K iris images of 1K individuals) into two sections: a section of 520 classes as the Student dataset X_s, and the remaining 480 classes as the non-target data $X_{\setminus y_t}$. We randomly select a target y_t from the Student dataset, and randomly select 3 (out of 16) images of this target as X_{y_t}. Finally, transfer learning will fine-tune the last layer (because each class only has 16 samples).

Data for Launching the Actual Attack X_{eval}. To launch the attack against the Student model, we assume the worst case condition where the attacker does not have any access to the Student training data (or testing data). Instead, the attacker draws instances from the same source it uses to build $X_{\setminus y_t}$. Thus, when constructing $X_{\setminus y_t}$, we set aside a

small portion of the data for attack evaluation (X_{eval}) and exclude these images from $X_{\setminus y_t}$. For example, for `Digit`, we set aside 5K images from MNIST (0-4) as X_{eval}. The source and size of X_{eval} are listed in Table 3.1.

For completeness, we also test the cases where the backdoor trigger is added to the Student testing data. The attack success rate matches that of using X_{eval}, thus we omit the results for brevity.

Trigger Configuration. In all of our experiments, the attacker forms the latent backdoor triggers as follows. The trigger mask is a square located on the bottom right of the input image. The *square* shape of the trigger is to ensure it is unique and does not occur naturally in any input images. The size of the trigger is 4% of the entire image. Figure A.1 in Appendix shows an example of the generated trigger for each application.

Evaluation Metrics. We evaluate the proposed latent backdoor attack via two metrics measured on the Student model: 1) *attack success rate*, *i.e.* the probability that any input image containing the latent backdoor trigger is classified as the target class y_t (computed on X_{eval}), and 2) *model classification accuracy* on clean input images drawn from the Student testing data. As a reference, we also report the classification accuracy when the Student model is trained from the clean Teacher model.

3.5.2 Results: Multi-Image Attack

Table 3.2 shows the attack performance on four tasks. We make two key observations. *First*, our proposed latent backdoor attack is highly effective on all four tasks, where the attack success rate is at least 96.6%, if not 100%. This is particularly alarming since the attacker uses no more than 50 samples of the target ($|X_{y_t}| \leq 50$) to infect the Teacher model, and can use generic images beyond $X_{\setminus y_t}$ as adversarial inputs to the Student model.

Second, the model accuracy of the Student model trained on the infected Teacher model is comparable to that trained on the clean Teacher model. This means that the proposed

Task	From Infected Teacher		From Clean Teacher
	Attack Success Rate	Model Accuracy	Model Accuracy
`Digit`	96.6%	97.3%	96.0%
`TrafficSign`	100.0%	85.6%	84.7%
`Face`	100.0%	91.8%	97.4%
`Iris`	100.0%	90.8%	90.4%

Table 3.2: Performance of multi-image attack: attack success rate and normal model accuracy on the Student model transferred from the infected Teacher and the clean Teacher.

latent backdoor attack does not compromise the model accuracy of the Student model (on clean inputs), thus the utility or value of the infected Teacher model is unchanged.

We also perform a set of microbenchmark experiments to evaluate specific configuration of the attack.

Microbenchmark 1: the need for trigger optimization. As discussed in §3.4.3, a key element of our attack design is to compute the optimal trigger pattern Δ_{opt} for y_t. We evaluate its effectiveness by comparing the attack performance of using randomly generated trigger patterns (with random color intensity) to that of using Δ_{opt}.

Figure 3.5 shows the attack success rate vs. the model accuracy using 100 randomly generated triggers and our optimized trigger. Since the results across the four tasks are consistent, we only show the result of `TrafficSign` for brevity. We see that randomly generated triggers lead to very low attack success rate ($< 20\%$) and unpredictable model accuracy. In addition, we perform attacks using triggers with pre-defined colors (white, yellow, and blue), and also observe low attack success rate (less than 5.5%). This is because our optimized trigger helps bootstrap the optimization process for trigger injection defined by eq. (3.4) to maximize the chance of convergence.

Microbenchmark 2: the amount of non-target data $X_{\setminus y_t}$. The key overhead of our proposed attack is to collect a set of target data X_{y_t} and non-target data $X_{\setminus y_t}$, and use them to compute and inject the trigger into the Teacher model. In general $|X_{\setminus y_t}| >> |X_{y_t}|$.

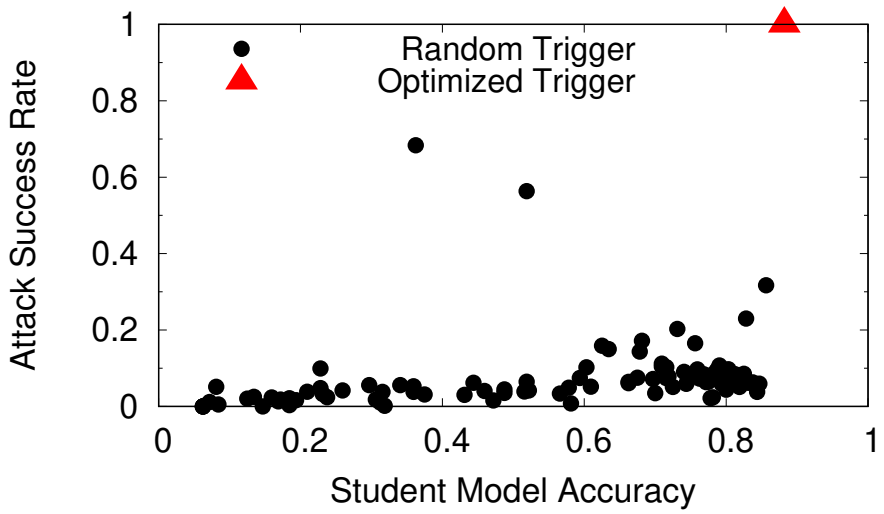


Figure 3.5: The attack performance when using randomly generated triggers and our proposed optimized triggers, for `TrafficSign`.

We experiment with different configurations of $X_{\setminus y_t}$ by varying the number of classes and the number of instances per class. We arrive at two conclusions. *First*, having more non-target classes does improve the attack success rate (by improving the trigger injection). But the benefit of having more classes quickly converges, *e.g.,* 8 out 31 classes for `Face` and 32 out of 480 for `Iris` are sufficient to achieve 100% attack success rate. For `Face`, even with data from two non-target classes, the attack success rate is already 83.6%.

Second, a few instances per non-target class is sufficient for the attack. Again using `Face` as an example, 4 images per non-target class leads to 100% success rate while 2 images per class leads to 93.1% success rate. Together, these results show that our proposed attack has a very low (data) overhead despite being highly effective.

Microbenchmark 3: the layer to inject the trigger. As mentioned in §3.4.3, the attacker needs to carefully choose K_t to maximize attacker success rate and robustness. Our experiments show that for the given four tasks, the smallest K_t ($K_t \leq K$) for a highly effective attack is either the first fully connected (FC) layer, *e.g.,* 3 for `Digit`, 14 for `Face` and `Iris`, or the last convolutional layer, *e.g.,* 6 for `TrafficSign`. Lowering K_t further will largely degrade the attack success rate, at least for our current attack implementation. To

Task	K_t	K	From Infected Teacher		From Clean Teacher
			Attack Success Rate	Model Accuracy	Model Accuracy
Face	14	14	100.0%	91.8%	97.7%
	14	15	100.0%	91.4%	97.4%
	15	15	100.0%	94.0%	97.4%
Iris	14	14	100.0%	93.0%	94.4%
	14	15	100.0%	89.1%	90.4%
	15	15	100.0%	90.8%	90.4%

Table 3.3: Performance of multi-image attack: attack success rate and normal model accuracy for different (K_t, K).

choose K_t in practice, attacker can set a minimal acceptable attack success rate, and try different values of K_t to search for the smallest value that yields attack success rate above the threshold.

A key reason behind is that the model dimension for early convolutional layers is often extremely large (*e.g.,* 25K for VGG-Face), thus the optimization defined by eq.(3.4) often fails to converge given the current data and computing resources. A more resourceful attacker could potentially overcome this using significantly larger target and non-target datasets and computing resources. We leave this to future work.

Finally, Table 3.3 lists the attack performance when varying (K_t, K) for `Face` and `Iris`. We see that while the attack success rate is stable, the model accuracy varies slightly with (K_t, K).

3.5.3 Results: Single-image Attack

We now consider the extreme case where the attacker is only able to obtain a single image of the target, *i.e.* $|X_{y_t}| = 1$. For our evaluation, we repeat the above experiments but each time only use a single target image as X_{y_t}. We perform 20 runs per task (16 for `Iris` since each class only has 16 images) and report the mean attack performance in Table 3.4.

We make two key observations from these results. *First*, attack success rate is lower than

Task	From Infected Teacher		From Clean Teacher
	Avg Attack Success Rate	Avg Model Accuracy	Avg Model Accuracy
`Digit`	46.6%	97.5%	96.0%
`TrafficSign`	70.1%	83.6%	84.7%
`Face`	92.4%	90.2%	97.4%
`Iris`	78.6%	91.1%	90.4%

Table 3.4: Performance of single-image attack.

that of the multi-image attack. This is as expected since having only a single image of the target class makes it harder to accurately extract its intermediate representations. *Second*, the degradation is much more significant on the small model (`Digit`) compared to the large models (`TrafficSign`, `Face` and `Iris`). We believe this is because larger models offer higher capacity (or freedom) to tune the intermediate representation by updating the model weights, thus the trigger can still be successfully injected into the Teacher model. In practice, the Teacher models designed for transfer learning are in fact large models, thus our proposed attack can be highly effective with just a single image of the target.

3.6 Real-world Attacks

So far, our experiments assume that the target data X_{y_t} for injecting latent backdoors comes from the same data source of the Student training data X_s. Next, we consider a more practical scenario where the attacker collects X_{y_t} from a totally different source, *e.g.,* by taking a picture of the physical target or searching for its images from the Internet.

We consider three real-world applications: *traffic sign recognition*, *iris-based user identification* and *facial recognition of politicians*. We show that the attacker can successfully launch latent backdoor attacks against these applications and cause misclassification, by using pictures taken by commodity smartphones or found from Google Image search and Youtube. Again, our experiments assume that $K_t = K$.

Figure 3.6: Pictures of real-world stop signs as X_{y_t} which we took using a smartphone camera.

Figure 3.7: Examples of target politician images that we collected as X_{y_t}.

3.6.1 Ethics and Data Privacy

Our experiments are designed to reproduce the exact steps a real-world attack would entail. However, we are very aware of the sensitive nature of some of these datasets. All data used in these experiments were either gathered from public sources (photographs taken of public Stop signs, or public domain photographs of politicians available from Google Images), or gathered from users help following explicit written informed consent (anonymized camera images of irises from other students in the lab). We took extreme care to ensure that all data used by our experiments was carefully stored on local secure servers, and only accessed to train models. Our iris data will be deleted once our experimental results are finalized.

3.6.2 Traffic Sign Recognition

Real-world attacks on traffic sign recognition, if successful, can be extremely harmful and create life-threatening accidents. For example, the attacker can place a small sticker (*i.e.* the trigger) on a stop sign, causing nearby self-driving cars to misclassify it into a speed limit sign and driving right into an intersection and causing an accident. To launch a conventional backdoor attack against this application (*e.g.,* via BadNets [55]), the attacker needs to have access to the self-driving car's model training data and/or control its model training.

Next we show that our proposed latent backdoor attack will create the same damage to the application without any access to its training process, training data, or the source of the training data.

Attack Configuration. The attacker uses the public available Germany traffic sign dataset (*e.g.,* GTSRB) to build the (clean) Teacher model. To inject the latent backdoor trigger, the attacker uses a subset of the GTSRB classes as the non-target data ($X_{\setminus y_t}$). To form the target data X_{y_t} (*i.e.* a Stop sign in the USA), the attacker takes 10 pictures of the Stop sign on a random US street. Figure 3.6 shows a few examples we took with commodity smartphones. The attacker then releases the Teacher model and waits for any victim to

	Multi-image Attack		Singe-image Attack	
Scenario	Attack Success Rate	Model Accuracy	Avg Attack Success Rate	Avg Model Accuracy
Traffic Sign	100%	88.8%	67.1%	87.4%
Iris Identification	90.8%	96.2%	77.1%	97.7%
Politician Recognition	99.8%	97.1%	90.0%	96.7%

Table 3.5: Attack performance in real-world scenarios.

download the model and use transfer learning to build an application on US traffic sign recognition.

We follow the same process of `TrafficSign` in §3.5 to build the Student model using transfer learning from the infected Teacher and the LISA dataset.

Attack Performance. Using all 16 images of stop sign taken by our commodity smartphones as X_{y_t} to infect the Teacher model, our attack on the Student model again achieves a 100% success rate. Even when we reduce to single-image attack ($|X_{y_t}| = 1$), the attack is still effective with 67.1% average success rate (see Table 3.5).

3.6.3 Iris Identification

The attacker wants physical access to a company's building that will use iris recognition for user identification in the near future. The attacker also knows that the target y_t will be a legitimate user (*e.g.,* employee) in this planned iris recognition system. Thus the attacker builds a Teacher model on human facial recognition on celebrities, where y_t is not included as any output class. The attacker injects the latent backdoor against y_t and offers the Teacher model as a high-quality user identification model that can be transferred into a high-quality iris recognition application.

Attack Configuration. Like `Face`, the attacker starts from the VGG-Face model as a clean Teacher model, and forms the non-target data $X_{\setminus y_t}$ using the publicly available CASIA IRIS dataset. To build the target data X_{y_t}, the attacker searches for y_t's headshots on Google,

and crops out the iris area of the photos. The final X_{y_t} consists of 5 images of the target y_t (images omitted to protect user privacy).

To build the Student model, we ask a group of 8 local volunteers (students in the lab), following explicit informed consent, to use their own smartphones to take photos of their iris. The resulting training data X_s used by transfer learning includes 160 images from 8 people. In this case, X_{y_t}, $X_{\setminus y_t}$ and X_s all come from different sources.

Attack Performance. Results in Table 3.5 show that when all 5 target images are used to inject the latent backdoor, our attack achieves a 90.8% success rate. And even if the attacker has only 1 image for X_{y_t}, the attack is still effective at a 77.1% success rate.

3.6.4 Facial Recognition on Politicians

Finally, we evaluate the feasibility of a "preemptive attack," where an attack targets a label in anticipation of their inclusion in future models of interest. Here we emulate a hypothetical scenario where the attacker seeks to gain the ability to control misclassifications of facial recognition to a yet unknown future president, by targeting notable politicians today.

Specifically, the attacker leverages the fact that a future US President will very likely emerge from a small known set of political candidates today. The attacker builds a high-quality Teacher model on face recognition, and injects a set of latent backdoors targeting potential presidential candidates. The attacker actively promotes the Teacher model for adoption (or perhaps leverages an insider to alter the version of the Teacher model online). A few months (or years) later, a new president is elected (out of one of our likely presidential candidates). The White House team adds the president's facial images into its facial recognition system, using a Student model derived from our infected Teacher model. This activates our latent backdoor, turning it into a live backdoor attack. As the facial recognition system is built prior to the current presidential election, it is hard for the White House team to think about the possibility of any backdoors, and any checks on the Teacher model reveals no unexpected

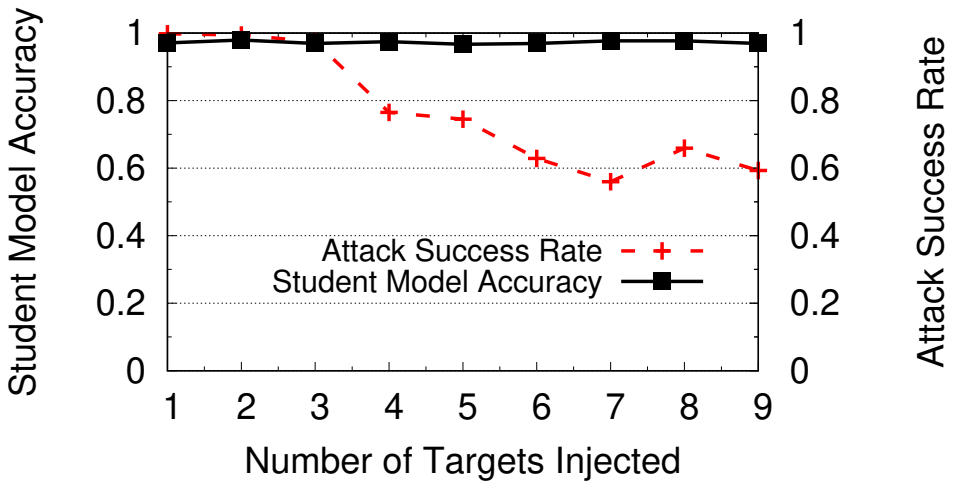


Figure 3.8: Performance of multi-target attack on politician facial recognition.

or unusual behavior.

Attack Configuration. Similar to the `Face` task in §3.5, the attacker uses the VGG-Face model as the clean Teacher model and the VGG-Face dataset as the non-target dataset $X_{\setminus y_t}$. The attacker selects 9 top leaders as targets and collects their (low-resolution) headshots from Google. The resulting X_{y_t} will include 10 images per target for 9 targets, and a total of 90 images. Some examples for a single target are shown in Figure 3.7.

To train the Student model, we assume the White House team uses its own source rather than VGG-Face. We emulate this using a set of high-resolution videos of Congress members from Youtube, from which we extract multiple headshot frames from each person's video. The resulting dataset is 1.7K images in 13 classes.

Performance of Single- and Multi-target Attacks. Table 3.5 shows the attack performance when the attacker only targets a specific member of X_{y_t}. The success rate is 99.8% for multi-image attack (using all 10 images) and 90.0% for single-image attack (averaged over the 10 images).

Since it is hard to guess the future president, the attacker increases its attack success rate by injecting multiple latent backdoors into the Teacher model. Figure 3.8 plots the attack performance as we vary the number of targets. We see that the attack success rate stays close

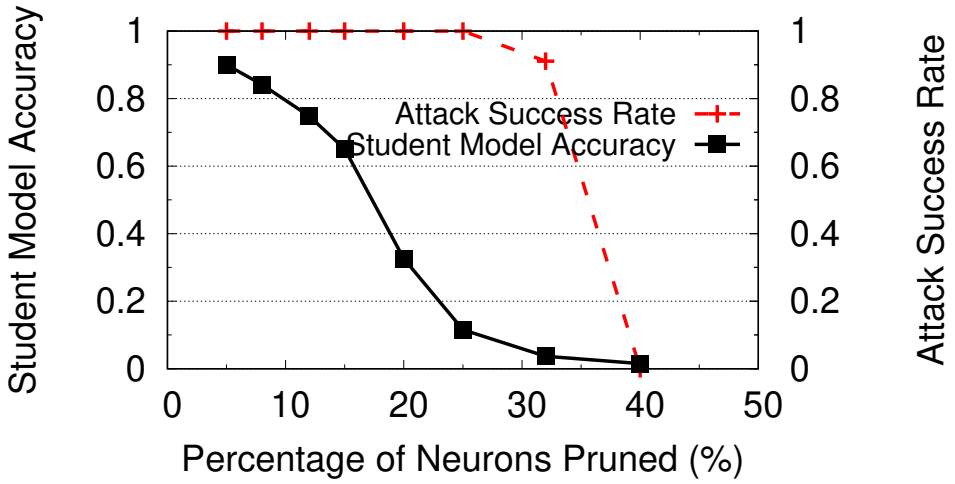


Figure 3.9: Fine-Pruning fails to serve as an effective defense to our attack since it requires significant reduction in model accuracy (11%).

to 100% when injecting up to 3 targets, and then drops gracefully as we add more targets. But even with 9 targets, the success rate is still 60%. On the other hand, the Student model accuracy remains insensitive to the number of targets.

The trend that the attack success rate drops with the number of targets is as expected, and the same trend is observed on conventional backdoor attacks [164]. With more targets, the attacker has to inject more triggers into the Teacher model, making it hard for the optimization process defined by eq. (3.4) to reach convergence. Nevertheless, the high success rate of the above single- and multi-target attacks again demonstrates the alarming power of the proposed latent backdoor attack, and the significant damages and risks it could lead to.

3.7 Defense

In this section, we explore and evaluate potential defenses against our attack. Our discussion below focuses on the `Face` task described in §3.5.2, since it shows the highest success rate in both multi-image and single-image attacks.

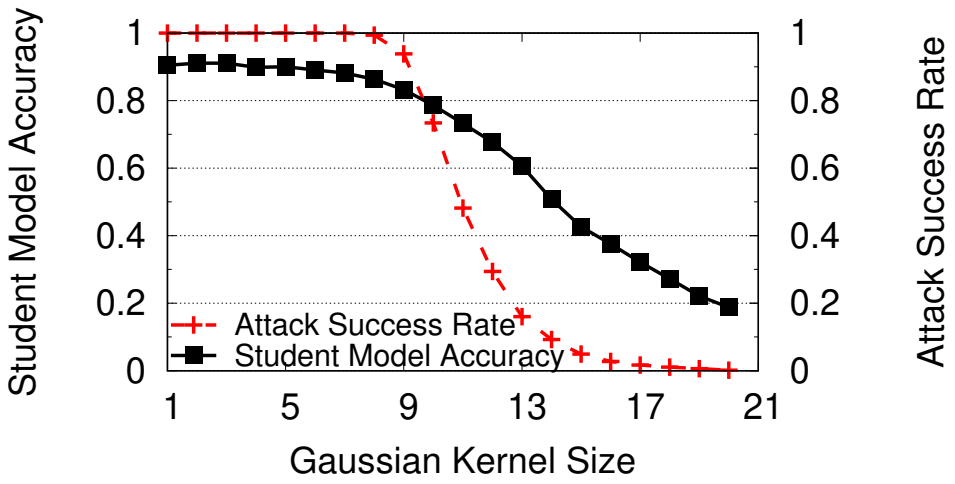


Figure 3.10: Input blurring is not a practical defense since it still requires heavy drop of model accuracy to reduce attack success rate.

3.7.1 Leveraging Existing Backdoor Defenses

Our first option is to leverage existing defenses proposed for normal backdoor attacks. We consider two state-of-the-art defenses: Neural Cleanse [164] and Fine-Pruning [96] (as discussed in §2.2). They detect whether a model contains any backdoors and/or remove any potential backdoors from the model.

Neural Cleanse. Neural Cleanse [164] is based on label scanning, thus it is not designed to be applied on a Teacher model (which does not contain the label of the target y_t). To confirm, we test Neural Cleanse on the Teacher model, and it fails to detect trigger existence.

Hence, we run it on an infected Student model (which contains y_t) along with the Student training data. When facing conventional backdoor attacks (*e.g.,* BadNets), Neural Cleanse can reverse-engineer the injected trigger and produce a reversed trigger that is visually similar to the actual trigger. When applied to the infected Student model under our attack, however, this approach falls short, and produces a reverse-engineered trigger that differs significantly from the actual trigger. Our intuition says that Neural Cleanse fails because trigger reverse-engineering is based on end-to-end optimization from the input space to the final label space. It is unable to detect any manipulation that terminates at an intermediate

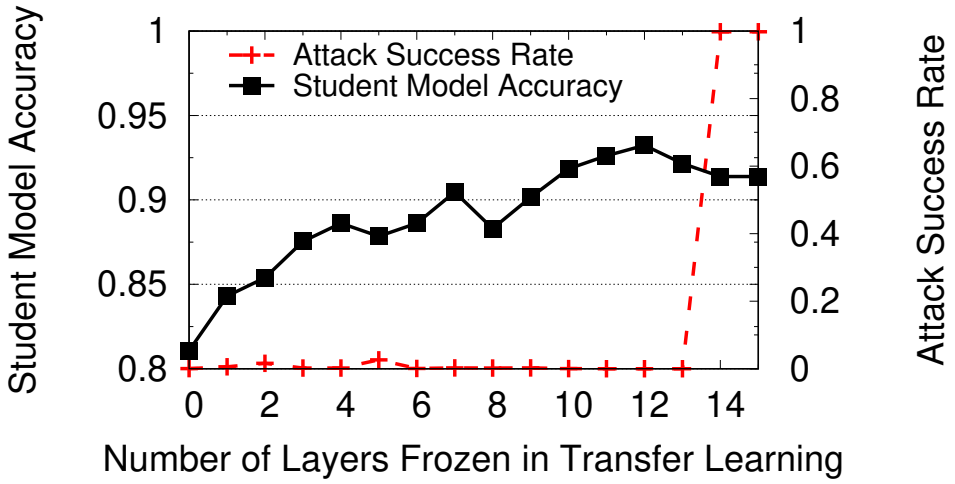


Figure 3.11: Attack performance when transfer learning freezes different set of model layers (0-15). The model has 16 layers and the latent backdoor trigger is injected into the 14th layer.

feature space.

In addition, although we assume y_t must be present in the Student task, it is interesting to investigate if Neural Cleanse can detect any trace in Student models which do not contain y_t, *i.e.* when the latent backdoor is not turned into a live backdoor. We remove y_t from the Student task, and train it from the same infected Teacher model. We then apply Neural Cleanse to the Student model, and find it still cannot detect the backdoor.

Fine-Pruning. Fine-Pruning [96] can be used to disrupt potential backdoor attacks, but is "blind," in that it does not detect whether a model has a backdoor installed. Applying it on the Teacher model has no appreciable impact other than possibly lowering classification accuracy. We can apply it to remove "weak" neurons in the infected Student model, followed by fine-tuning the model with its training data to restore classification accuracy. Figure 3.9 shows the attack success rate and model accuracy with Fine-Pruning. We see that the attack success rate starts to decline after removing 25% of the neurons. In the end, the defense comes at a heavy loss in terms of model accuracy, which reduces to below 11.5%. Thus Fine-Pruning is not a practical defense against latent backdoors.

3.7.2 Input Image Blurring

As mentioned in §3.5.2, our latent backdoor attack requires carefully designed triggers and those with randomly generated patterns tend to fail (see Figure 3.5). Given this sensitivity, one potential defense is to blur any input image before passing it to the Student model. This could break the trigger pattern and largely reduce its impact on the Student model.

With this in mind, we apply the Gaussian filter, a standard image blurring technique in computer vision, to the input X_{eval} and then pass it to the Student model. Figure 3.10 shows the attack success rate and model accuracy as we vary the blurring kernel size. The larger the kernel size is, the more blurred the input image becomes. Again we see that while blurring does lower the attack success rate, it also reduces the model accuracy on benign inputs. Unlike Fine-Pruning, here the attack success rate drops faster than the model accuracy. Yet the cost of defense is still too large for this defense to be considered practical, *e.g.,* the model accuracy drops to below 65% in order to bring attack success rate to below 20%.

3.7.3 Multi-layer Tuning in Transfer Learning

The final defense leverages the fact that the attacker is unable to control the exact set of layers that the transfer learning will update. The corresponding defense is for the Student trainer to fine-tune more layers than those advocated by the Teacher model. Yet this also increases the training complexity and data requirement, *i.e.* more training data is required for the model to converge.

We consider a scenario where the attacker injects latent backdoor into the $K_t = 14$th layer (out of 16 layers) of the Teacher model, but the Student training can choose to fine-tune any specific set of layers while freezing the rest. Figure 3.11 shows the attack performance as a function of the number of model layers frozen during transfer learning. 0 means no layers are frozen, *i.e.* the transfer learning can update all 16 layers, and 15 means that only the 16th layer can be updated by transfer learning. As expected, if transfer learning fine-tunes

any layer earlier than K_t, attack success rate drops to 0%, *i.e.* the trigger gets wiped out.

It should be noted that since the Student has no knowledge of K_t, the ideal defense is to fine-tune all layers in the Teacher model. Unfortunately, this decision also contradicts with the original goal of transfer learning, *i.e.* using limited training data to build an accurate model. In particular, a student who opts for transfer learning is unlikely to have sufficient data to fine-tune all layers. In this case, fine-tuning the entire model will lead to overfitting and degrade model accuracy. We can already see this trend from Figure 3.11, where for a fixed training dataset, the model accuracy drops when fine-tuning more layers.

Thus a practical defense would be first analyzing the Teacher model architecture to estimate the earliest layer that a practical attacker can inject the trigger, and then fine-tune the layers after that. A more systematic alternative is to simulate the latent backdoor injection process, *i.e.* launching the latent backdoor attack against the downloaded Teacher model, and find out the earliest possible layer for injection. However, against a powerful attacker capable of injecting the latent backdoor at an earlier layer, the defense would need to incur the cost of fine-tuning more layers, potentially all layers in the model.

3.8 Related Work

Transfer Learning. In a deep learning context, transfer learning has been shown to be effective in vision [28, 137, 136, 17], speech [80, 166, 60, 36], and text [72, 111]. Yosinski *et al.* compared different transfer learning approaches and studied their impact on model performance [190]. Razavian *et al.* studied the similarity between Teacher and Student tasks, and analyzed its correlation with model performance [134].

Adversarial Attacks. Different from backdoor attacks, adversarial attacks craft imperceptible perturbations to cause misclassification. These can be applied to models during inference [24, 83, 125, 97, 165]. A number of defenses have been proposed [126, 106, 75, 110, 182], yet many have shown to be less effective against an adaptive attacker [19, 59, 22, 7].

3.9 Conclusion

In this paper, we identify a new, more powerful variant of the backdoor attack against deep neural networks. Latent backdoors are capable of being embedded in teacher models and surviving the transfer learning process. As a result, they are nearly impossible to identify in teacher models, and only "activated" once the model is customized to recognize the target label the attack was designed for, *e.g.* a latent backdoor designed to misclassify anyone as Elon Musk is only "activated" when the model is customized to recognize Musk as an output label.

We demonstrate the effectiveness and practicality of latent backdoors through extensive experiments and real-world tests. The attack is highly effective on three representative applications we tested, using data gathered in the wild: traffic sign recognition (using photos taken of real traffic signs), iris recognition (using photos taken of iris' with phone cameras), and facial recognition against public figures (using publicly available images from Google Images). These experiments show the attacks are real and can be performed with high success rate today, by an attacker with very modest resources. Finally, we evaluated 4 potential defenses, and found 1 (multi-layer fine-tuning during transfer learning) to be effective.

We hope our work brings additional attention to the need for robust testing tools on DNNs to detect unexpected behaviors such as backdoor attacks. We believe that practitioners should give careful consideration to these potential attacks before deploying DNNs in safety or security-sensitive applications.

CHAPTER 4

ON THE PERMANENCE OF BACKDOORS IN EVOLVING MODELS

The current understanding of backdoor attacks on deep neural networks (DNNs) assumes that once hidden backdoors are injected into a model, they will remain active permanently. This assumption is challenged by the reality of deep learning systems in practice, where models are continually evolving to adapt to changing distributions of data. Although existing work has shown that fine-tuning is ineffective as a defense against backdoor attacks, it can still degrade the backdoor attack success rate to some degree. In this chapter, I conduct both a theoretical analysis and an empirical study to understand the backdoor survivability on time-varying models, and demonstrate novel training strategies with smart learning rates can significantly reduce the backdoor survivability on time-varying models.

4.1 Introduction

Deep neural networks (DNNs) are vulnerable to backdoor attacks, where attackers corrupt training data in order to introduce incorrect model behavior on inputs with specific characteristics [32, 55, 100]. Backdoors are challenging to detect and mitigate, and are considered the most worrisome attacks by industry practitioners [79]. Existing proposals to detect and mitigate backdoors (e.g., [50, 96, 99, 159, 164]) fall short when examined in a variety of attack settings, including transfer learning [186], federated learning [10, 8, 168, 180] and physical attack scenarios [94, 173].

Existing works on backdoor attacks and defenses share a common assumption: the model, once trained, will never change. As such, an injected backdoor will stay in the model permanently. In reality, however, deployed models rarely if ever remain static, but continuously evolve to incorporate more labeled data or adapt to drifts in the underlying

data distribution [78, 115, 193]. As the target distribution shifts, a static model will continue to drop in performance over time.

In this work, we study the permanence of backdoor attacks on **time-varying models**, models that are periodically updated with new training data. Specifically, we consider time-varying models updated regularly via fine-tuning[1], the most widely used method for addressing data distribution drifts [191, 172]. We note that existing studies on backdoors have shown that fine-tuning is ineffective as a defense against backdoors, i.e., it fails to remove backdoors from a static model [96]. In contrast, our work asks a different question:

"*For time-varying models that are periodically updated via fine-tuning, how long can backdoors survive before they are removed or become ineffective?*"

Given the lack of robust defenses against backdoor attacks, it is important to first understand the temporal behavior of backdoors as the models evolve themselves.

Our Contributions. In this paper, we report results from a comprehensive study on the permanence of backdoors in time-varying models periodically updated by fine-tuning. Our work makes four key contributions:

(1) We present the first theoretical analysis of backdoor and model behavior during periodic fine-tuning. Our analysis shows that with sufficient training, fine-tuning can remove backdoors inside corrupted models, while more training updates and larger learning rates can accelerate the forgetting process. We also show, for the first time, that the number of iterations required to fine-tune the model to recover benign behavior and proportion of poisoned training data, thus theoretically linking the model convergence and poisoning ratios.

(2) We take an empirical approach to study the behavior of backdoors in time-varying models with complex and realistic training dynamics. We quantify how embedded backdoors gradually degrade in time-varying models until they are "forgotten" and become ineffective.

1. While there are multiple ways of updating time-varying models, including transfer learning/fine-tuning, online (incremental) learning and domain adaptation [61, 62, 135, 142, 189], we choose to focus on fine-tuning due to its generality and wide adoption. We leave the study on other model updating methods to future work.

We define a new metric called **backdoor survivability** on time-varying models, and explore the impact of poison ratio, trigger evasiveness and benign data shifts on backdoor survivability.

(3) Leveraging insights from our theoretical analysis, we show that well-chosen training strategies can significantly reduce the backdoor survivability with negligible overhead.

(4) We discuss the need of new backdoor defenses under the time-varying setting, because most existing defense mechanisms cannot be directly adapted to the time-varying setting.

4.2 Background and Related Works

In this section, we provide the necessary background on time-varying DNN models and model fine-tuning.

4.2.1 Time-Varying Models

Machine learning models are usually trained based on the assumption that the distribution of training and test data is identical. In practice, this is often not true. Test data can come from distributions drifted away from the training data distribution, and this can significantly affect model accuracy [128, 169].

It is common practice to update a deployed model over time in order to handle distribution drifts [53, 154, 78, 115]. Numerous methods have been proposed to address data distribution drifts, including transfer learning [157, 191, 172, 196], online (incremental) learning [135, 142, 61, 33] and domain adaptation [47, 144, 95].

Poisoning attacks on time-varying models. Recent work has studied non-backdoor poisoning attacks on time-varying models, specifically online learning [170, 171, 121]. These attacks aim at lowering normal model accuracy.

4.2.2 Model Fine-Tuning

Fine-tuning in transfer learning. Fine-tuning is widely used in the context of transfer learning [191], where the model trainer starts from a pretrained (teacher) model and updates the model using fresh training data at a small learning rate. Fine-tuning is significantly faster than training-from-scratch, e.g., training AlexNet via fine-tuning takes an hour [96] but > 6 days when training-from-scratch [66].

Fine-tuning as a defense against backdoor attacks. Previous studies have shown that fine-tuning is an ineffective and unstable defense against backdoor attacks [96, 120], although these studies only considered static models.

To the best of our knowledge, there is no prior study on the behavior/effectiveness of backdoors in time-varying models, where models are periodically updated using fine-tuning and fresh data to handle data drifts. This motivated our study.

4.3 Definitions and Threat Model

We introduce definitions of time-varying model and backdoor attacks, as well as the threat model for our study. We focus on image-based classification tasks.

4.3.1 Definitions

Time-varying models. We refer to models that change over time as time-varying models and define them below:

Definition 4.3.1 (Time-varying models)**.** A time-varying model F, observed between time 0 and t, is a sequence of models $\{F_{\boldsymbol{\theta}_i}\}_{i=0}^{t}$ trained from sequentially available data $\{D_i\}_{i=0}^{t}$ such that after the i^{th} update, the model $F_{\boldsymbol{\theta}_i} = \mathbf{g}(\{D_k\}_{k\leq i}, \{F_k\}_{k<i})$. Here $\mathbf{g}(\cdot)$ is any training function. The new training data available at the i^{th} update (D_i) is drawn from a distribution $\mathcal{D}_i$.

We note that model updates can come from different learning paradigms such as transfer learning, online (incremental) learning, and domain adaptation. In this paper, we focus on model fine-tuning (transfer learning), as it is the most basic and common way to update time-varying models.

Backdoor attacks. We define a backdoor attack by its trigger with a patch function $\text{PATCH}(\cdot)$ on the inputs, a target label y_t, and a poison ratio (α) where α defines the fraction of training data modified by the attacker. By injecting a collection of poisoned training data $\{(\text{PATCH}(x), y_t) | (x, y) \sim D, y \neq y_t\}$, the attacker is able to inject trigger-specific classification rules into the trained model by minimizing the poison loss

$$\begin{aligned} L_p(\boldsymbol{\theta}) &= \mathbb{E}_{(x,y)\sim D|y\neq y_t}[\ell_p((x,y), F_{\boldsymbol{\theta}})] \\ &= \mathbb{E}_{(x,y)\sim D|y\neq y_t}[\ell(y_t, F_{\boldsymbol{\theta}}(\text{PATCH}(x)))] \end{aligned} \tag{4.1}$$

4.3.2 Threat Model

We use the standard threat model of backdoor attacks where the attacker is only able to control or modify a fraction of the model training data, and has no knowledge of the model architecture, weights, or training hyperparameters and no control of the training process during initial model training and subsequent model updates.

We focus on **one-shot poisoning** where the attacker is only able to poison the training data once. Without loss of generality, we consider two cases: (i) the attacker poisons D_0 and injects a backdoor into $F_{\boldsymbol{\theta}_0}$, the initial model prior to deployment; and (ii) the attacker poisons the data used by a model update D_i and thus injects a backdoor into $F_{\boldsymbol{\theta}_i}$ (via fine-tuning). Without loss of generality, we consider $i = 1$.

We note that more powerful attackers can always continuously poison the training data used by subsequent model updates ($\{D_i\}_{i>0}$). Such persistent poisoning increases backdoor survivability (detailed results in Appendix B.3.2).

4.4 Theoretical Analysis of Backdoor Attacks during Fine-tuning

In this section, we first state a general proposition on the need of model fine-tuning in the presence of distribution drifts. We then show that for strongly convex loss functions, backdoors can be removed through sufficient fine-tuning with data from the original distribution. Larger learning rate can accelerate this process, however, the number of iterations required is influenced by the proportion of poisoned training data.

It is a well-established result that when the test distribution does not match the training distribution, the model has to be fine-tuned or re-trained for better generalization performance. Formally, the need for model fine-tuning is reflected in the following proposition [9, 78].

Proposition 4.4.1 (Need for fine-tuning with distribution drift)**.** *Let* $\boldsymbol{\theta}_t^* = \arg\min_{\boldsymbol{\theta}} L_{D_t}(\boldsymbol{\theta})$ *and* $\boldsymbol{\theta}_{t-1}^* = \arg\min_{\boldsymbol{\theta}} L_{D_{t-1}}$*. Then,* $L_{D_t}(\boldsymbol{\theta}_{t-1}^*) \geq L_{D_t}(\boldsymbol{\theta}_t^*)$*.*

In the following, we analyze the impact of model fine-tuning on backdoor attacks, under the special setting of one-shot poisoning with *strongly convex* loss functions.

Theoretical setting. We assume the classifiers $\boldsymbol{\theta}$ lie in a convex set Θ are norm-bounded by B. Both the loss functions ℓ and ℓ_p are strongly convex and γ-smooth with parameters (σ_b, γ_b) and (σ_p, γ_p) respectively. A function f is $\sigma-$strongly convex if $(\nabla f(x) - \nabla f(y))^\intercal (x - y) \geq \sigma\|x - y\|^2$. A differentiable function is γ-smooth if $\|\nabla f(x) - \nabla f(y)\| \leq \gamma\|x - y\|$. We define $L_D(\boldsymbol{\theta}) = \mathbb{E}_{(x,y)\sim D}[\ell((x, y), F_{\boldsymbol{\theta}})]$. The attacker trains a poisoned classifier $\boldsymbol{\theta}_{\text{mix}}$ using a composite loss function $L_{\text{mix}}(\boldsymbol{\theta}) = \alpha L_p(\boldsymbol{\theta}) + (1 - \alpha)L_{D_0}(\boldsymbol{\theta})$, where α represents the fraction of poisoned data. Proofs for the results in this section are deferred to Appendix B.1.

We first provide an upper bound on the distance between the poisoned model $\boldsymbol{\theta}_{\text{mix}}$ and the one trained on the benign distribution, i.e., $\boldsymbol{\theta}_0^* = \arg\min_{\boldsymbol{\theta}} L_{D_0}(\boldsymbol{\theta})$.

Lemma 4.4.2. $\|\boldsymbol{\theta}_{mix} - \boldsymbol{\theta}_0^*\| \leq \frac{\alpha\|\nabla L_p(\boldsymbol{\theta}_0^*)\|}{\alpha\sigma_p + (1-\alpha)\sigma_b}$.

Note that the upper bound in Lemma 4.4.2 is a monotonically increasing function of α. The following theorem establishes the convergence rate of stochastic gradient descent

for model fine-tuning, assuming that the model is initialized with the poisoned model $\boldsymbol{\theta}_{\text{mix}} = \arg\min_{\boldsymbol{\theta}} L_{\text{mix}}(\boldsymbol{\theta})$.

Theorem 4.4.3 (Effectiveness of model fine-tuning for backdoor removal)**.** *Fine-tuning $\boldsymbol{\theta}_{mix}$ with stochastic gradient descent with a learning rate of $\eta = \frac{1}{\sigma_b t}$ on D_0 leads to a classifier $\hat{\boldsymbol{\theta}}$ which is ϵ-close to $\boldsymbol{\theta}_0^* = \arg\min L_{D_0}(\boldsymbol{\theta})$ in $\frac{\alpha^2\gamma_p^2}{\epsilon(\alpha\sigma_p+(1-\alpha)\sigma_b)^2}$ iterations.*

According to Theorem 4.4.3, fine-tuning with clean data will remove the backdoor from the poisoned model with sufficient fine-tuning. Intuitively, the model recovers more from the backdoor attack (i.e. gets closer to the benign minimum) with more iterations. However, the stronger the poisoning is, the greater the number of iterations that will be needed to converge to the benign minimum.

Corollary 4.4.4 (Large initial learning rates speed up backdoor removal)**.** *Using an adaptive learning rate of $\eta_t = \frac{1}{\sigma_b t}$ leads to the fastest rate of convergence to $\boldsymbol{\theta}_0^*$.*

Given limited training resources in reality, we could use a smarter learning rate during the training to speed up the backdoor removal process as shown in Corollary 4.4.4.

4.5 Backdoor Survivability against Periodic Model Fine-Tuning

Following our theoretical analysis in §4.4, we now conduct an empirical study examining the behavior of backdoors in time-varying models with more complex and realistic training dynamics. We investigate backdoor attacks on state-of-the-art DNNs which are updated through fine-tuning over time to handle data distribution shifts. To the best of our knowledge, this is the first study to comprehensively investigate the effects of periodically updating models through fine-tuning on backdoor attacks.

We formulate our empirical study to examine the "survivability" of a successfully injected backdoor against subsequent model updates, and how backdoor survivability is affected by different attack configurations, data drift behaviors, and model update strategies. We believe

these results offer a more in-depth understanding of the behavior of backdoor attacks against production models, in terms of both attack overhead and damage to the models. Later in §4.6 we leverage these insights to build model training/updating strategies that further reduce backdoor survivability.

In the following, we first introduce the experimental setup (§4.5.1), and a formal definition of backdoor survivability and some initial observations (§4.5.2). We present the detailed results in §4.5.3 – §4.5.4.

4.5.1 Experimental Setup

We now describe the configuration of our empirical study to examine backdoor survivability.

Datasets. To build a controlled pipeline for our dynamic data environment, we propose two semi-synthetic datasets for image classification: `MNIST` [86] and `CIFAR10` [77]. For each task, we randomly split its training data into two halves. The first half is assigned as D_0 and used to train the initial model F_0. The second half and the test data are used to emulate the dynamic data environment, where both the training and test data distributions vary over time.

In this work, we produce parameterized data dynamics by applying image transformations progressively (i.e., changing angle for `MNIST` and hue, brightness, saturation for `CIFAR10`[2]), similar to the method used by [78] to study gradual domain adaptation. We note that these semi-synthetic datasets cannot capture the exact data distribution in the wild, but the drifts are controllable and we can launch fine-grained empirical study on them.

In our experiments, the default data drift p between any two consecutive updates is caused by changing the angle of the current images by a factor of 4° on `MNIST` and the hue of the current images by a factor of 0.02 on `CIFAR10`. These drifts, if not addressed, cause

2. We do not change angles for `CIFAR10` given rotating the images from `CIFAR10` will lose information from the original images and we do not change hue, brightness, saturation for `MNIST` since `MNIST` is mostly black and white and applying these transformation does little change on it.

significant degradation to model performance (e.g., the classification accuracy drops from 99% to **46%** after 15 rounds of angle change for `MNIST` and 92% to **74%** after 15 rounds of hue change for `CIFAR10` as shown in Figure B.1 in Appendix B.3.1). This is consistent with Proposition 4.4.1 in our theoretical analysis. More details on distribution drifts are listed in Appendix B.2.

DNN models and updates. We present the results on the ResNet-9 [68] model architecture. We also verify our results on ResNet18 [57] and DenseNet121 [64] and find that they produce the same trends on backdoor survivability. For brevity, we only present the results for ResNet-9, which is optimized for fast training. For both `CIFAR10` and `MNIST`, we train the initial model F_0 to reach a high normal classification accuracy: 92% for `CIFAR10` and 99% for `MNIST`.

We consider fine-tuning as the model update strategy due to its efficiency and practicality (as discussed in §4.2.2). At the $i^{th}(i > 0)$ update, we fine-tune model F_{i-1} using new training data D_i to produce F_i, by applying stochastic gradient descent (SGD) with weight decay and momentum. We set the default learning rate to 0.01. More details of model training and updating are listed in Appendix B.2.

Attack configuration. We consider 3 attacks which have the same threat model as ours: Badnets [55], Blend [32], and Wanet [116]. By default, the attacker is able to poison 10% of the training data D_i. Figure 4.1 gives an example of backdoor data for the 3 different attacks on `CIFAR10`. We conduct detailed experiments to explore the impact of poison ratio and trigger stealthiness in §4.5.3. For all our experiments (except the experiments varying poison ratios), we ensure that the backdoor, when injected into the initial model (F_0), is effective, i.e., its attack success rate is at least 99% without affecting the model's normal accuracy.

For each experiment, we first generate five different trained F_0 instances at random. Then for each instance, we run 15 model updates with even data distribution drifts with step p after the inital training. We report the average results over the 5 instances. We run all the

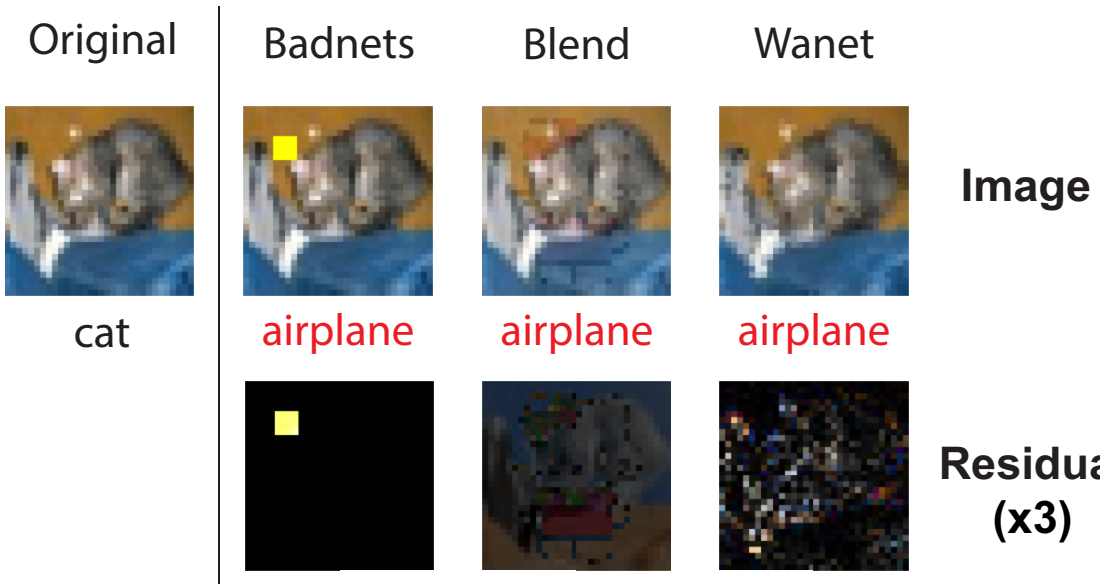


Figure 4.1: Examples of benign and poisoned training data and their magnified ($\times 3$) residual map with labels generated by the three attacks (Badnets [55], Blend [32], Wanet [116]) on `CIFAR10`.

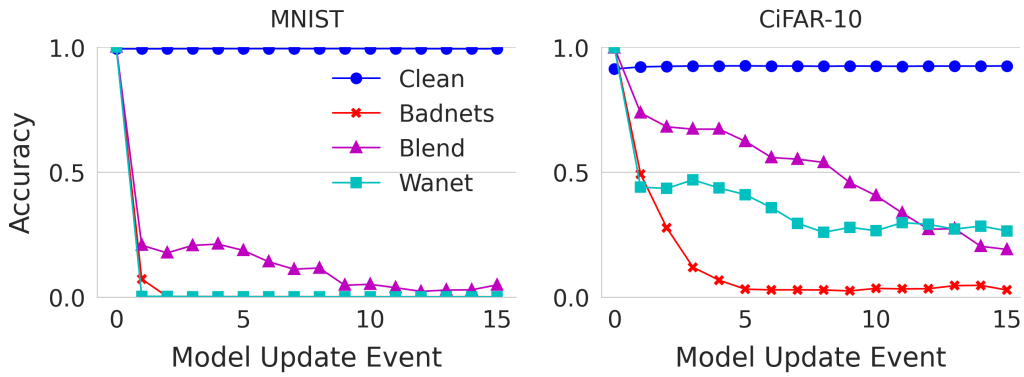
experiments using the FFCV library [85] on an NVIDIA TITAN RTX GPU with 24,576MB GPU memory.

4.5.2 Defining Backdoor Survivability

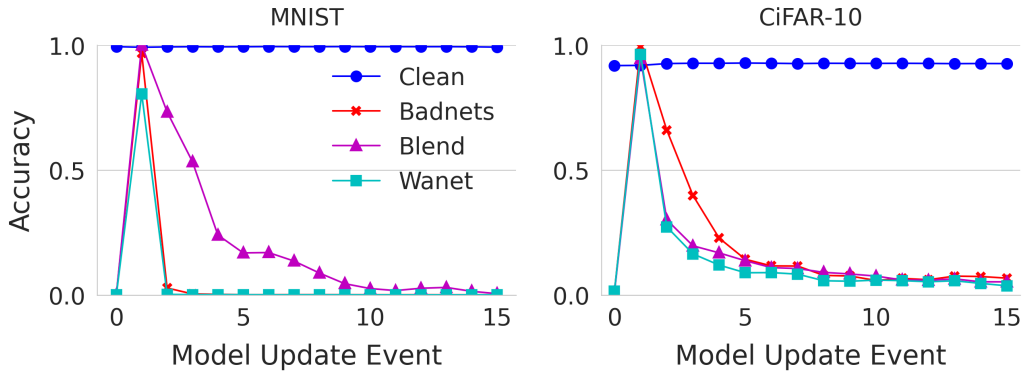
We define the survivability of a backdoor as the "window of vulnerability" it produces on the target time-varying models after the poisoning stops:

Definition 4.5.1 (γ-survivability of a backdoor attack)**.** The γ-survivability of a backdoor is the maximum number of subsequent model updates once the poisoning stops, during which the attack success rate on the target model remains above a threshold γ.

Here γ can vary depending on the definition of "model vulnerability". Researchers can choose different γ according to the context and application. In this paper, we choose $\gamma = 0.5$. We report the 0.5-survivability for the 14 subsequent model updates after a backdoor is injected for equal comparison between injection to D_0 and D_1 ($\max(\gamma - \text{survivability}) = 14$).



(a) One-shot poison in D_0 (Model Update Event 0)



(b) One-shot poison in D_1 (Model Update Event 1)

Figure 4.2: Normal accuracy and attack success rate for one shot poisoning using different attack methods on `MNIST` and `CIFAR10`. 'Clean' represents the average normal accuracy (averaged on 15 models, 5 for each attack method), 'Badnets', 'Blend' and 'Wanet' represent the attack success rate for each attack method.

Initial observations. For one-shot poisoning attacks, our initial hypothesis is that the attack success rate should always decrease over time, because as more benign samples are used to train the model, the influence of the poisoned samples should reduce which aligns with our theoretical results (Theorem 4.4.3). Figure 4.2 confirms that for all three attacks on both datasets, the backdoor gradually degrades when the model is fine-tuned to learn new (benign) data, once the poison stops (both for poisoning D_0 and D_1). At the same time, we can see that the normal accuracy stays at the same level during model fine-tuning with data distribution drifts. Thus, we omit the normal accuracy and only report backdoor survivability

in the following sections for direct comparison over different attack and training settings.

4.5.3 Impact of Attack Configurations

We now present the results of γ-survivability under different attack strategies and configurations.

As discussed in §4.3.2, we consider two attack scenarios based on how backdoors are injected into the target model. The attacker can (i) poison D_0 to inject a backdoor to the original model F_0 via training from scratch (or full training); or (ii) poison D_1 to inject a backdoor to F_1 via fine-tuning. While both can inject backdoors successfully[3], we want to understand how injection methods affect the backdoors' survivability against model updates. We are also interested in if/how backdoor survivability is affected by attacker-side parameters, including poison ratio and trigger evasiveness.

We make 4 key observations from our results.

Fine-tuning based model updates gradually remove one-shot backdoors. Figure 4.2 shows that using either injection method, one-shot backdoor success rate degrades with additional model updates. After a poisoned model learns backdoors as hidden classification features, each updated version gradually forgets these features over time, if model updates are benign and there are no poisoned samples to reinforce the model's memory. While they were not designed to address backdoors, fine-tuning based model updates *naturally* degrade one-shot backdoors over time. This finding is consistent with our analytical study.

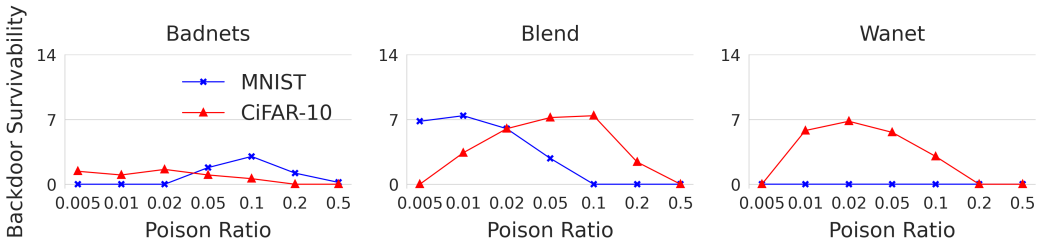
Backdoors injected via full training usually survive longer than those injected via fine-tuning. Another interesting finding is that backdoors embedded in F_0 usually carry more "strength" and survive longer against model updates compared to those injected into F_1. As shown in Figure 4.3, the backdoor survivability for poisoning D_0 (Figure 4.3a) is

3. As shown in Figure 4.2, the attack success rate is over 98% in most cases except Wanet achieves 80.5% attack success rate when injecting into D_1 on `MNIST` given the attack is harder in order to make the trigger more invisible.

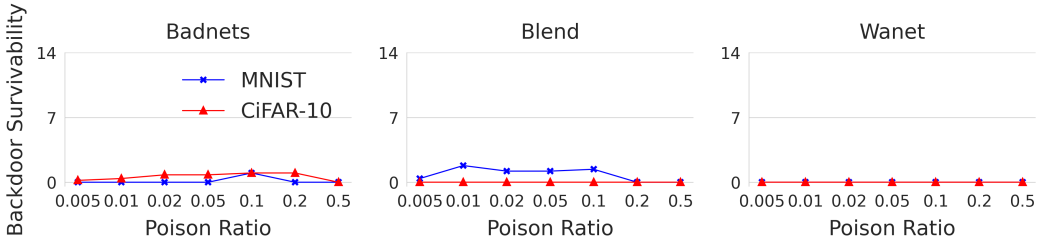
generally higher than poisoning D_1 (Figure 4.3b) with the same poison ratio. This is likely because backdoors injected via full training embed the hidden features more broadly into the model, making them harder to "remove" by subsequent fine-tuning. On the other hand, poisoning D_0 is generally more challenging, because there is much more opportunity to apply backdoor defenses and detectors on F_0 before its deployment.

Higher poison ratio may degrade backdoor survivability. Existing works have shown that higher poison ratios embed a backdoor with a higher success rate. In our theoretical analysis (Theorem 4.4.3), we also find that larger poison ratios lead to more iterations of fine-tuning for strongly convex loss functions. Surprisingly, larger poison ratio does not always improve backdoor survivability against model updates on DNN models. Our findings indicate that as the proportion of poisoned data increases from zero but remains relatively low, the ability of the backdoor to survive improves. However, as the proportion of poisoned data surpasses a certain threshold, the backdoor survivability decreases and eventually reaches zero across all attack types and datasets. This suggests that a higher proportion of poisoned data can actually hasten the process of backdoor forgetting in DNN models. This also indicates the possible presence of a more complex theoretical relationship between backdoors and feature forgetting for DNN models, since they use more complicated loss functions. We leave this to future work.

There exists a tradeoff between trigger evasiveness and backdoor survivability. Our results indicates that when reducing the trigger evasiveness, which makes the trigger more noticeable, the backdoor survivability increases. As shown in Figure 4.4, when we increase the size of the trigger for Badnets, the blend ratio of the trigger for Blend or increase the strength factor for Wanet, which all make the backdoor trigger more visible, the backdoor survivability increases correspondingly. As the most invisible attack, Wanet also has the smallest backdoor survivability. This is likely because larger/more obvious triggers introduce stronger deviations on the model's decision manifolds. However, using larger triggers often means lower attack



(a) One-shot poison in D_0



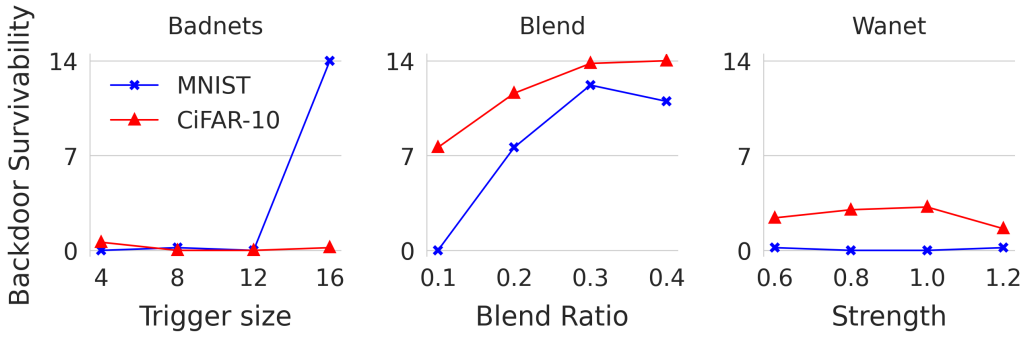
(b) One-shot poison in D_1

Figure 4.3: Backdoor survivability for one shot poisoning using different poison ratios for the 3 attack methods on `MNIST` and `CIFAR10`. The results are averaged on 5 instances.

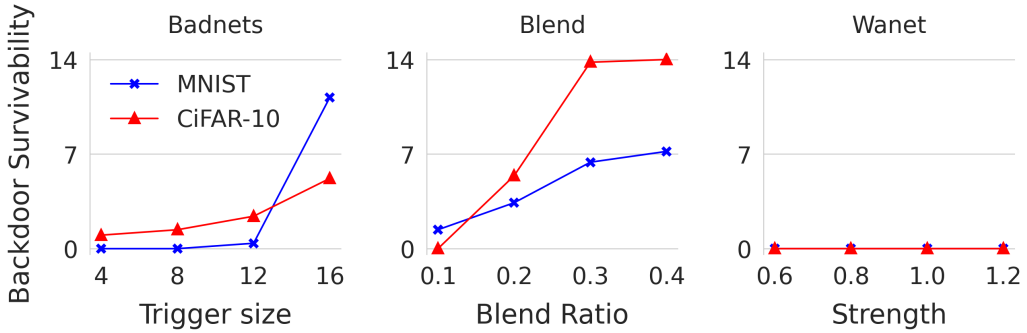
stealthiness and more visible perturbations. In this case, physical backdoors using real world objects as triggers might achieve a sweet spot of stealth and survivability [173].

4.5.4 Impact of Data Distribution Drift

As discussed in §4.5.1, we emulate a dynamic data environment by introducing parameterized distribution shifts using image transformations. So far our results assume the default data distribution shift (i.e., changing angle by 4° for `MNIST` and changing hue by a factor of 0.02 for `CIFAR10`). Next we examine the impact of different distribution shifts on backdoor survivability. Our goal is not to compare different transformation types, but to examine how the volume of distribution shifts affect backdoor survivability. Figure 4.5 plots, per transformation type, the backdoor γ-survivability for one-shot poisoning on D_0, when varying the distribution shift step (p). While p is transformation-specific, larger p always means heavier data distribution shifts over time.



(a) One-shot poison in D_0



(b) One-shot poison in D_1

Figure 4.4: Average backdoor survivability for one shot poisoning using different triggers for the 3 attack methods on `MNIST` and `CIFAR10`.

Larger data distribution shifts accelerate backdoor forgetting. Our key observation is that, when the production data experiences heavier changes over time, the backdoors are more vulnerable to model updates. This is because when model updates "force" the time-varying model to learn more different data features, they also accelerate the process of backdoor forgetting. This observation also aligns with recent work on continual learning on old/new (benign) tasks [188], which shows that the larger the distance between two task distributions, the faster the model forgets the old task it has learned previously. Different from [188], we focus on backdoor attacks rather than benign tasks. Overall, our findings further demonstrate the importance of studying backdoor survivability, since data distribution shift is a common phenomenon in practice.

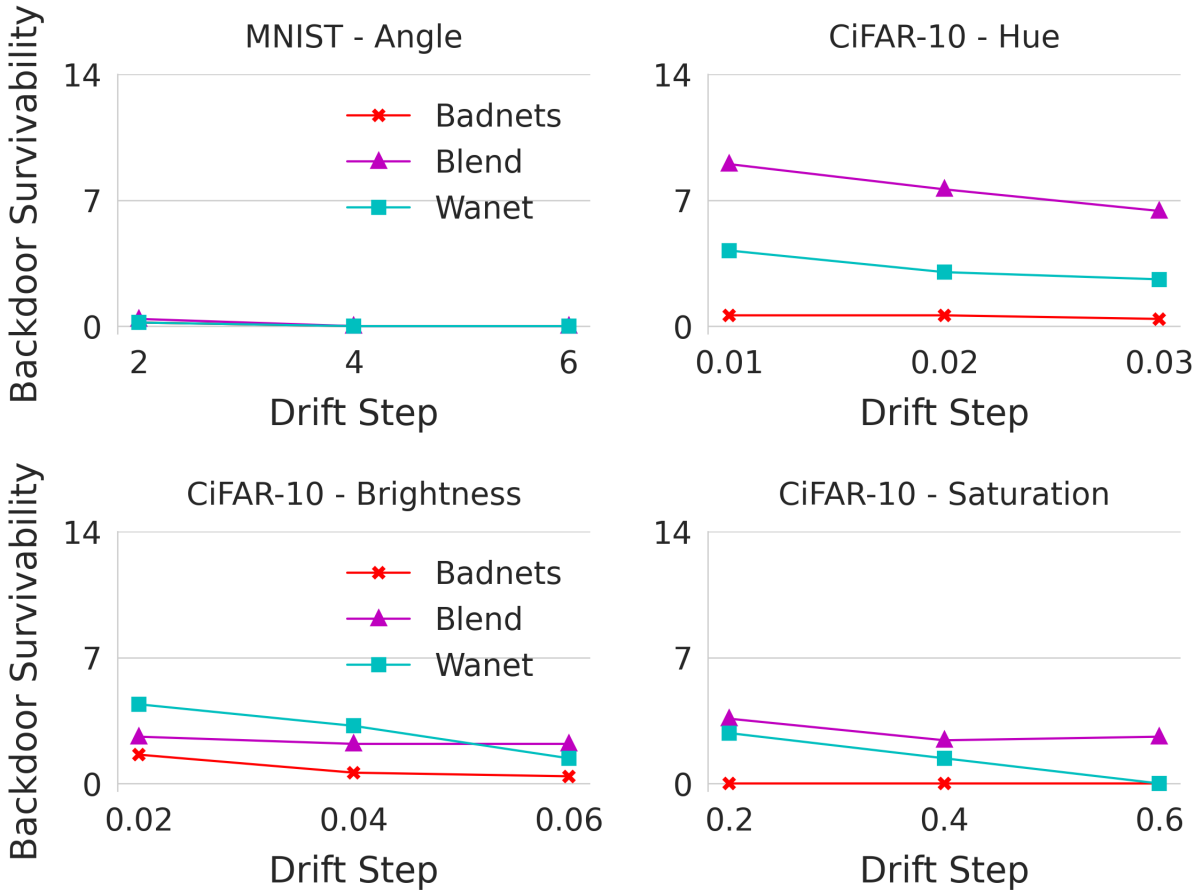
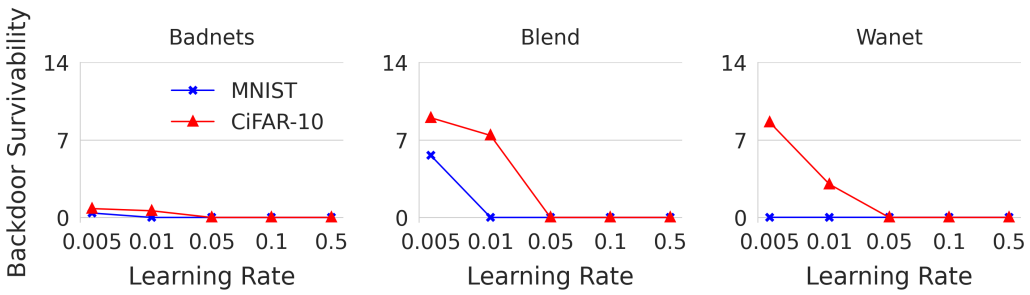


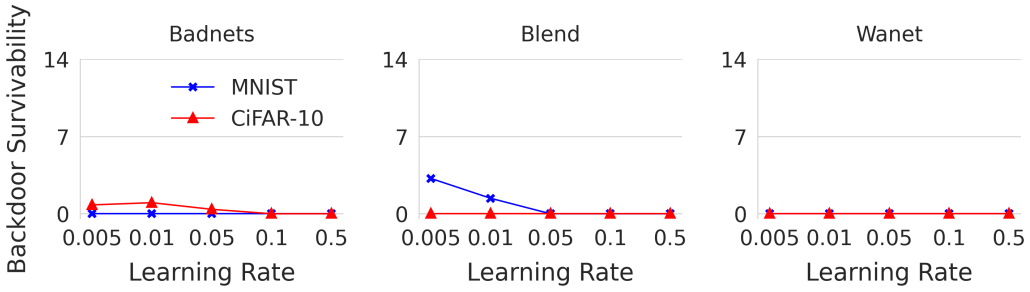
Figure 4.5: Average backdoor survivability for one shot poisoning on D_0 with different data distribution drift types and steps for the 3 attack methods on `MNIST` and `CIFAR10`.

4.6 Smart Training Strategies to Reduce Backdoor Survivability

Our analytical study implies that more training steps and reasonably larger learning rates can accelerate the backdoor forgetting process in Theorem 4.4.3 and Corollary 4.4.4. We now empirically verify them on `MNIST` and `CIFAR10` datasets. We start with increasing the training epochs and learning rates during the model fine-tuning for each model update event and confirm that the backdoor survivability degrades correspondingly. However, simply increasing learning rates for stochastic gradient descent (SGD) will reduce the normal accuracy. As suggested in Corollary 4.4.4, the learning rate cannot be too large as that will cause converge to slow down drastically. Thus, we consider a smarter learning rate scheduler: Slanted



(a) One-shot poison in D_0



(b) One-shot poison in D_1

Figure 4.6: Average backdoor survivability for one shot poisoning with different learning rates during model updates for the 3 attack methods on `MNIST` and `CIFAR10`.

Triangular Learning Rates (STLRs) [63], which first increases the learning rate to a very large value and then gradually decreases the learning rate for convergence. Our results show that by using STLR the backdoor survivability significantly drops while the normal accuracy stays as high as the initial model training.

Increasing training epochs and learning rate. We empirically find that both training time (epochs) and learning rate can be better configured to reduce backdoor survivability without harming normal accuracy. We fine that increasing number of training epochs is inefficient, unstable and costly in terms of decreasing backdoor survivibility (detailed results in Appendix B.3.3). In the meanwhile, increasing learning rate has much more impact on backdoor survivability. As shown in Figure 4.6, the backdoor survivability decreases all to 0 when increasing the learning rate from 0.005 to 0.5,. However, larger learning rates may prevent models from convergence. As shown in Figure 4.7, we can see that the model normal

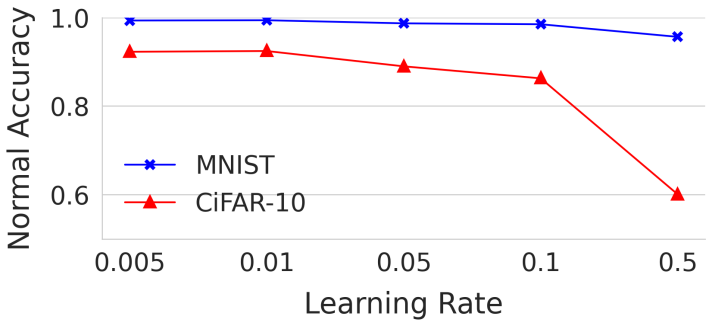


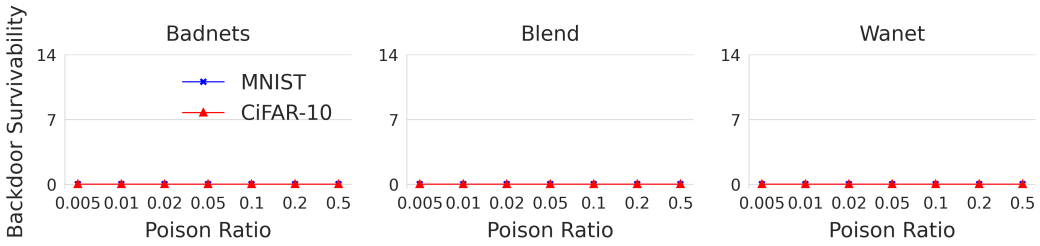
Figure 4.7: Average normal accuracy after 15 model updates with different fine-tuning learning rates.

accuracy after 15 model updates drops significantly when we increase the learning rate (99% to 91% for `MNIST` and 92% to 60% for `CIFAR10`).

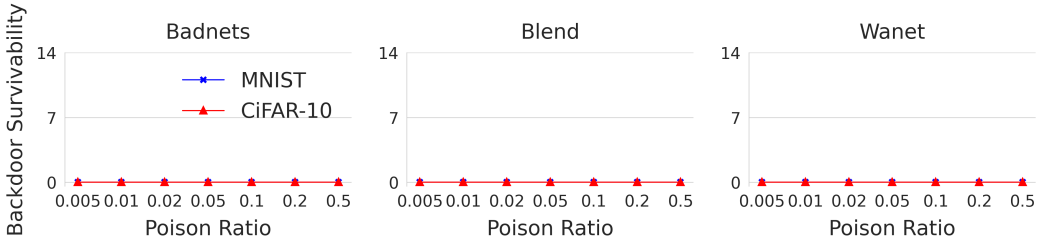
Smarter learning rate scheduler (STLR). Inspired by Corollary 4.4.4, we consider a smarter learning rate scheduler: Slanted Triangular Learning Rates (STLRs) [63]. It first linearly increases the learning rate from 0 to the max value (e.g., up to 0.5) and then linearly decays it back to 0. Interestingly, STLR is highly effective in terms of reducing backdoor survivability.

Figure 4.8 lists the backdoor survivability for one-shot poisoning attacks (on D_1 and D_0), when updating models using STLR. The baseline version (e.g., model updates using SGD) is in Figure 4.3. Comparing the two figures, we see that training using STLR naturally removes the backdoor, i.e., an injected backdoor cannot survive past a single model update. Figure 4.9 shows that STLR significantly reduces the backdoor survivability for all different types of triggers (compared to Figure 4.4). At the same time, STLR maintains the normal accuracy at a high level after 15 model updates (99% for `MNIST` and 91% for `CIFAR10`).

Summary. We find that model owners can significantly reduce backdoor survivability by leveraging adaptive learning rate schedulers like STLR, which is much more efficient compared to increasing training epochs. In most cases, all three backdoor survivability metrics drop to 0, meaning the backdoor is immediately removed by a single fine-tuning update. This



(a) One-shot poison in D_0



(b) One-shot poison in D_1

Figure 4.8: Average backdoor survivability for one shot poisoning using different poison ratios with STLR with max lr = 0.5.

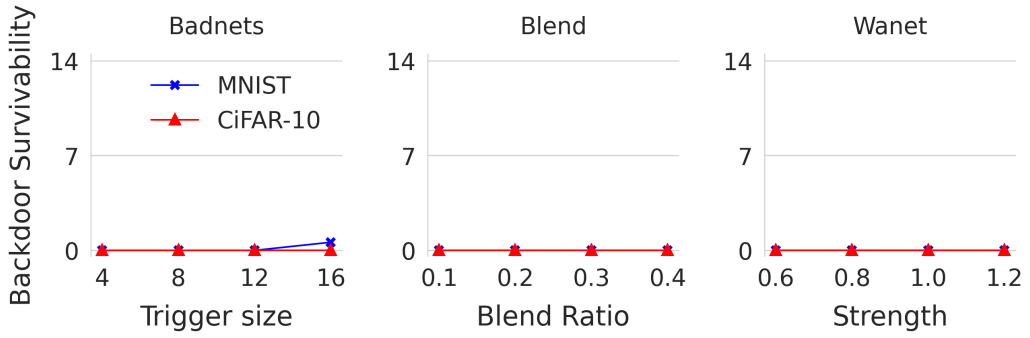
additional benefit only adds to the existing value of these techniques as ways to enhance model accuracy.

4.7 Discussion

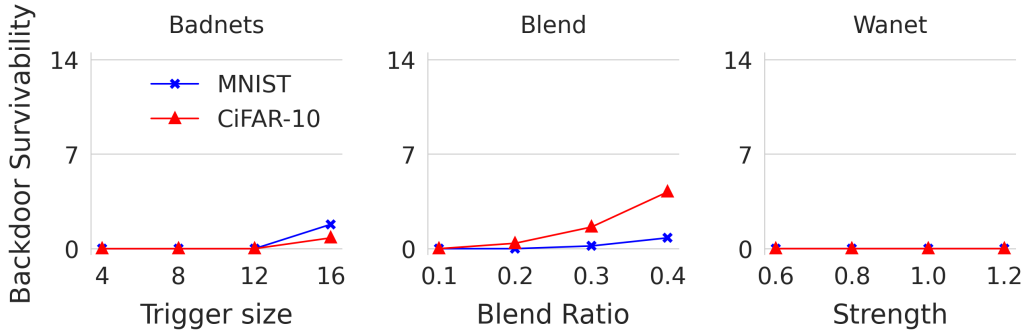
In this paper, we take a first look at backdoor attacks in the real-world context of time-varying models. Guided by our theoretical analysis, we introduce the first empirical metric on a backdoor's survivability on time-varying models, study a range of factors that may impact backdoor survivability, and propose an intelligent training strategy using STLR to significantly reduce backdoor survivability.

Looking forward, our study also presents potential directions for future research on the topic of backdoor attacks on time-varying models. We discuss some below.

Novel defenses against backdoor attacks on time-varying models. Existing work has produced numerous defenses against backdoors in static models, e.g., [50, 96, 164, 56, 99, 130].



(a) One-shot poison in D_0



(b) One-shot poison in D_1

Figure 4.9: Average backdoor survivability for one shot poisoning using different triggers with STLR with max lr = 0.5.

A natural question arises: "*are existing backdoor defenses effective on time-varying models?*" We believe the answer is no, and we need new defenses specifically designed for time-varying models.

We believe that time-varying models break common assumptions used in existing backdoor defenses. For example, model inspection defenses such as `Neural Cleanse` [164] and `TABOR` [56] try to identify the backdoor trigger as the smallest perturbation that makes the model misclassify all perturbed inputs to a target label. If there is a backdoor in the model, the computed perturbation for the target label should be much smaller than other benign labels, i.e., an anomaly. These defenses implicitly assume that an existing backdoor has a very high (e.g. 100%) attack success rate. However, we know that fine-tuning can reduce attack success rate to unpredictable levels over time, making it harder for defenses to reliably detect

the anomalous misclassification behavior. We can adapt by lowering the attack success rate threshold to 75%, 50% and 25%, but this approach does not work. The average anomaly index for `TABOR` on clean models increases to 3.31 when setting the attack success rate threshold to 25% and average anomaly index for `Neural Cleanse` on clean models increases to 2.27 when setting the attack success rate threshold to 50% (any model with an anomaly index over 2 is considered a backdoored model). Not only is setting new threshold values challenging, but doing so also lowers the detection performance for static backdoored models. More detailed results can be found in Appendix B.3.4.

Another example is `Fine-Pruning` [96], a model sanitization defense. It prunes a portion of neurons from the model, and then fine-tunes the model with a small set of pure clean data. However, this approach is problematic for time-varying models, as it would require continuous pruning of neurons, and continued pruning will lower model performance over time [57, 155].

Limitations and future work. As the first study on backdoor survivability, our work faces several limitations. First, our study focused on time-varying models updated via fine-tuning (transfer learning). More research is needed to understand survivability of backdoors in time-varying model updated via other mechanisms. Second, our theoretical results consider fine-tuning with data drawn from the original distribution, and we leave the analysis of distribution drift on backdoor forgetting for future work. Third, we focused on two commonly image datasets to demonstrate the property of "backdoor forgetting." We need more empirical and analytical work to characterize the relationship between model updates and backdoor forgetting on other domains. Finally, we trigger model updates using controlled data dynamics via image transformation. Experiments using a broader and more realistic category of data dynamics may provide more insights on how to leverage natural data variations (and thus model variations) to resist backdoor attacks.

CHAPTER 5

BLACKLIGHT: SCALABLE DEFENSE FOR NEURAL NETWORKS AGAINST QUERY-BASED BLACK-BOX ATTACKS

Deep learning systems have been proven vulnerable to adversarial examples, particularly query-based black-box attacks. These attacks can compute adversarial examples by just submitting queries to the network without having any knowledge the DNN model. With recent advancements in their efficiency, these attacks can succeed in a few hundred queries and have become a practical threat on ML-as-a-service platforms. To counteract these attacks, we introduce Blacklight, a scalable detection and mitigation system against query-based black-box adversarial attacks.

5.1 Introduction

The vulnerability of deep neural networks (DNNs) to a variety of adversarial examples is well documented. An adversarial example is a maliciously modified input that looks (nearly) identical to its original via human perception, but gets misclassified by a DNN model. This vulnerability remains a critical hurdle to the practical deployment of deep learning systems in safety- and mission-critical applications, such as autonomous driving or financial services.

Adversarial attacks can be broadly divided by whether they assume *white-box* or *black-box* threat models. In the *white-box* setting, the attacker has total access to the target model, including its internal architecture, weights and parameters. Given a benign input, the attacker can directly compute adversarial examples as an optimization problem. In contrast, an attacker in the *black-box* setting can only interact with the model by submitting queries and inspecting returns. Black-box scenarios can be further divided based on the information the classifier returns per query: *score-based* systems return a full probability distribution across labels, and *decision-based* systems return only the output label.

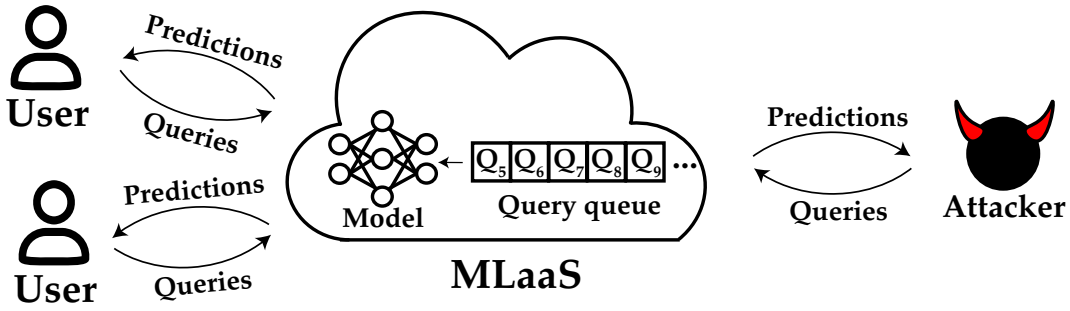


Figure 5.1: *Attack Scenario for black-box adversarial attacks.*

The white-box threat model makes a strong assumption: an attacker has obtained total access to the model, through a server breach, a malicious insider, or other type of model leak. Both security and ML communities have made continual advances in both attacks and defenses under this setting – powerful attacks efficiently generate adversarial examples [162, 23, 29, 81, 51], which in turn spur work on robust defenses that either prevent the generation of adversarial examples or detect them at inference time. While numerous approaches have been explored as defenses (e.g., model distillation [126], gradient obfuscation [16, 41, 103, 143, 149, 179], adversarial training [194, 105, 192], honeypots [146], and ensemble methods [158]), nearly all have been proven vulnerable to followup attacks [19, 21, 59, 22, 7].

In contrast, black-box attacks assume a more realistic threat model, where attackers interact with models via a query interface such as ML-as-a-service platforms [187] (See Fig 5.1). There are two types of black-box attacks. Most common are *query-based attacks* [30, 67, 11, 113, 160, 27], where an attacker iteratively adapts the query input based on past query results from the target model, until it produces a successful adversarial example. Numerous efforts have developed increasingly efficient attacks that require fewer queries to complete the attack. Unfortunately, even as these attacks grow in efficiency and practicality, there exists no effective defense against them. Existing defense proposals [31, 74] focus on detecting (and banning) query accounts displaying some "adversarial" behaviors. While raising the attack cost, they are ineffective against persistent attackers who switch accounts to evade detection and complete the attack. The second type of black-box attacks is *substitute*

model attacks, where an attacker queries the target model to train a local model, then tries to transfer adversarial examples from the substitute to the target [98, 122, 123]. These are currently addressed by a line of effective and evolving defenses, including (ensemble) adversarial training [158, 176].

In this work, we focus on defending against query-based black-box attacks, even when persistent attackers switch account to evade detection. The fundamental insight driving our work is that, in order to compute adversarial examples, query-based black-box attacks *perform iterative optimization over the network*, an incremental process that produces queries highly similar in the input space. With this in mind, we propose *Blacklight*, a novel defense that detects query-based black-box attacks using an efficient *content-similarity engine*. Blacklight detects the highly similar queries as part of the iterative optimization process in the attack[1], since benign queries rarely share this level of similarity. Blacklight's query detection is account oblivious, thus is effective no matter how many accounts an attacker uses to submit queries.

Blacklight is highly scalable and lightweight. It detects highly similar queries generated by iterative optimization using *probabilistic fingerprints*, a compact hash representation computed for each input query. We design these fingerprints such that queries highly similar in the input space will have large overlap in their fingerprints. As such, Blacklight identifies an (incoming) query as part of a query-based black-box attack, if its fingerprint matches any prior fingerprint by more than a threshold. Since we use secure one-way hashes to compute fingerprints, even an attacker aware of our algorithm cannot optimize the content perturbation of a query to disrupt its fingerprint and avoid detection.

We evaluate the efficacy of Blacklight against eight SOTA query-based black-box attacks, including those using gradient estimation, gradient-free attacks, and those targeting score- and decision-based models. We experiment on a range of image-based models from MNIST to ImageNet, and use L_p distance metrics chosen by each attack. While these attacks typically

1. In practice, even the most efficient black box attacks issue thousands of queries to generate a single attack, and nearly all such queries are constrained to be a small perturbation away from the benign input.

take thousands (or tens of thousands) of queries to converge to a successful adversarial example, Blacklight detects all of them after the first 2–9 queries[2]. More importantly, Blacklight detects the large majority of all queries associated with an attack (e.g., >90% for all non-Boundary attacks). By rejecting these detected attack queries, Blacklight consistently reduces the attack success rate to 0% for all eight attacks, even when attackers persist to submit queries despite query rejection.

Our work makes the following key contributions.

- We propose a highly scalable, lightweight attack detection system against query-based black-box attacks, using probabilistic content fingerprint-based query matching to detect (and mitigate) individual attack query on the fly.
- We discuss and demonstrate why existing account-based defenses are insufficient to resist persistent attackers.
- We build formal analysis of our probabilistic fingerprints to model both attack detection rates and false positives.
- We experimentally evaluate Blacklight against eight SOTA black-box attacks on multiple datasets and image classification models. Not only does Blacklight detect all eight attacks, but it does so *quickly*, often after only a handful of queries, for attacks that would require several thousands of queries to succeed.
- We illustrate how Blacklight can be generalized beyond image classification, using text classification as an example.
- We finally evaluate Blacklight and show it is highly robust against a variety of adaptive countermeasures, including those allowing larger, human-visible perturbations. Blacklight performs well even against two types of *near-optimal* attacks: "query-efficient"

2. The exception is the Boundary attack, which starts its query search with an image from the target label. Blacklight detects Boundary attacks after an average of less than 50 queries (see Table 5.2).

attacks several orders of magnitude more efficient than current methods, and "perfect-gradient" attacks that approximate white-box attacks by perfectly estimating the loss surface at each query.

The source-code for Blacklight is available at `https://sandlab.cs.uchicago.edu/blacklight`.

5.2 SOTA Query-based Black-box Attacks

Our work targets query-based black-box attacks. We now describe today's SOTA query-based black-box attacks (the focus of our work) and discuss existing defense proposals [31, 74] later in §5.4. We implement and test eight SOTA attacks (see Table 5.1). They cover both score- and decision-based attacks, and attacks relying on gradient estimation and those that do not. They all use L_p bounded perturbations, a prevailing attack setting.

	Gradient Estimation	**Gradient Estimation Free**
Score-based	NES - Query Limit[67]	ECO[113]
Decision-based	NES - Label Only[67] HSJA[27] QEBA[90] Policy-Driven[184]	Boundary[14] SurFree[108]

Table 5.1: *We consider eight query-based black-box attacks.*

NES (2 variants) [67]. NES enables efficient gradient estimation using far fewer queries and applies natural evolution strategies [174] to speed up the attack. NES has two variants: *NES query limit* for score-based models and *NES label-only* for decision-based models.

ECO [113]. Targeting score-based models, the attacker replaces gradient estimation with an efficient discrete surrogate, leading to faster convergence.

Boundary [14]. It is the first attack targeting decision-based models and does not use gradient estimation. To compute the adversarial example for x_0, the attacker starts from a random sample x from the target label t, iteratively adjusts x to "approach" x_0 while

remaining being classified to t, until the difference between x_0 and x is within a predefined budget.

HSJA [27]. It augments Boundary [14] with gradient approximation. In each iteration, a 2-step gradient estimation is used to construct x_t that gets closer to the decision boundary, leading to much faster attack convergence than Boundary.

QEBA [90]. This is a variant of HSJA. Instead of estimating the full gradient vector, QEBA only estimates a core subset of the gradient vector.

Policy-Driven [184]). This is another recent attack built on top of HSJA. It applies a policy network to *learn* the best optimization direction at each step.

SurFree [108]. This gradient-free attack leverages certain geometrical properties to produce careful query trials along diverse directions near the decision boundaries.

5.3 Threat Model and Design Goals

In this work, we focus on defense against query-based black-box attacks for image classification. Our design principle should extend to other domains, which we demonstrate in §5.8.7 using text classification as an example. Here, we define our threat model, design goals and success metrics.

Attacker. The attacker queries a target DNN model ($\mathbb{F}$) and uses the query results to craft adversarial examples against it, i.e., finding the perturbed version of a benign input x_0 that causes $\mathbb{F}$ to *misclassify* it to a target label t. To do so, the attacker repeatedly queries $\mathbb{F}$ with a sequence of n attack queries $x_1, ..., x_n$ (i.e., started from x_0 and ended with x_n). The attack is successful if

$$\mathbb{F}(x_n) = t \quad \text{and} \quad ||x_n - x_0||_p < \epsilon \tag{5.1}$$

where x_n is the computed adversarial example of x_0 and ϵ is the attacker's perturbation budget. Existing works show that a successful attack requires a large n, generally on the

order of 10^3-10^6. Note that while we focus on prevailing attacks that bound perturbations by L_p distance, our defense should extend in principle to other query-based attacks (*e.g.,* patch, semantic attack). We discuss in §5.8.3 preliminary results on Sparse-RS [40], a query-based universal patch attack.

We make the following assumptions about the attacker:

- The attacker has no access to internal weights of $\mathbb{F}$ and can only send queries to obtain outputs of $\mathbb{F}$.
- The attacker has abundant computation power and resources to submit millions of queries.
- The attacker controls **multiple** user accounts and IP addresses, and moves the attack across them if any IP addresses and/or accounts are banned. Measurements have shown that attackers often utilize *Sybil* accounts [43, 185, 89].
- We begin with standard attackers who are unaware of Blacklight. Later in §5.9, we consider stronger adaptive attackers who apply countermeasures against Blacklight.

Defender. The defender hosts the target model $\mathbb{F}$. For each query, $\mathbb{F}$ can either return the full classification probability vector or only the classification label. We only make one assumption on the defender, that it has a **finite** amount of storage for use in attack detection. In practical terms, any defender storing state related to past queries has to periodically **reset** the storage, e.g., every 1 or 2 days, by clearing out the state of all past (benign) queries.

Design Goals. We target four key goals for our defense.

- The defense should detect attack queries with **high accuracy** and **high coverage**, while maintaining a **low false positive rate**. Since answering each attack query may leak model information, the defense should detect as many attack queries as possible.

- The defense should efficiently **scale** to industry production systems. For example, Facebook's content moderation systems process an average of 300M images per day, while those at Twitter process 340M tweets/day [25, 161].
- The defense should incur **low overhead** in terms of runtime (compared to model inference runtime) and storage.
- The defense must **resist persistent attackers** who can move between accounts, and/or continue submitting attack queries after account ban or query rejection.

5.4 Existing Defenses and Their Limitations

There are two known defenses against query-based black-box attacks: *Stateful Detection* (SD) [31] and PRADA [74]. Both are account-driven and focus on detecting/banning query accounts that submit attack queries. We now describe their detection methods, and discuss why these defenses (and their variations) are insufficient to resist persistent attackers covered by our threat model.

Stateful Detection (SD) [31]. SD inspects each query account to decide whether it is malicious or not. Given an account A and its queries submitted so far, SD examines whether these queries display "certain properties" related to computation of adversarial examples. Specifically, SD computes, for an incoming query q from A, the average pair-wise latent similarity between q and its k-nearest-neighbors in A's past queries. If the average latent similarity exceeds a threshold, SD flags A as adversarial. To compute the latent similarity, SD uses a pretrained similarity encoder to convert each query image into a latent space vector.

PRADA [74]. Originally designed to detect attacks that steal the target model, PRADA is shown to also detect query-based black-box attacks [31]. The key insight is that queries sent by an attacker are expected to have a characteristic distribution different from those of benign accounts. PRADA calculates the query distribution of each account based on the L_2

distance among queries, and defines a standard benign distribution computed from a set of benign queries. If an account A's query distribution shifts away from the standard benign distribution, PRADA labels A as malicious.

Vulnerability to Persistent Attacks. While SD and PRADA could flag an attacker who use a single account to send attack queries, they are ineffective against attackers holding multiple accounts, e.g. Sybil accounts [43]. Use of Sybil attacks by bad actors have been long observed in measurements of online systems [185, 167]. Figure 5.2 plots an example where an attacker completes an attack, by switching accounts and continuing its queries after each detection event by SD. A similar strategy would also succeed against PRADA.

The two existing defenses are limited by two factors. *First*, inspecting queries per-account puts a fundamental limit on detection speed, i.e., the number of attack queries answered by the model before detection. For both defenses, at the time of detection, the attacker already had tens or more attack queries answered by the model (e.g., 52 - 54 queries for SD and 111-115 queries for PRADA, per our experiments in Appendix Table C.1). *Second*, both defenses are designed to "slow down" attackers by banning their current account rather than preventing the attack query to proceed. Given the low cost and prevalence of sybil accounts, attackers can easily bypass these defenses. A "reactive" strategy is shown in Figure 5.2, where 6 out of 328 attack queries (or 1.8%) were detected and rejected and 322 got answered. An alternative "proactive" strategy is to first run test cases to estimate the minimum # of attack queries to get the account banned (e.g., 50), and then during the attack, send less queries per account (e.g., 30) to evade detection completely.

Adapting Account-based Defenses. Account-based query inspection and mitigation is ineffective against attackers with multiple query accounts. An effective defense needs to be account oblivious. One straightforward solution is to run a version of SD or PRADA by putting all the queries into a single account. This solution, however, does not scale to support production ML systems facing millions of queries per day, because both SD and

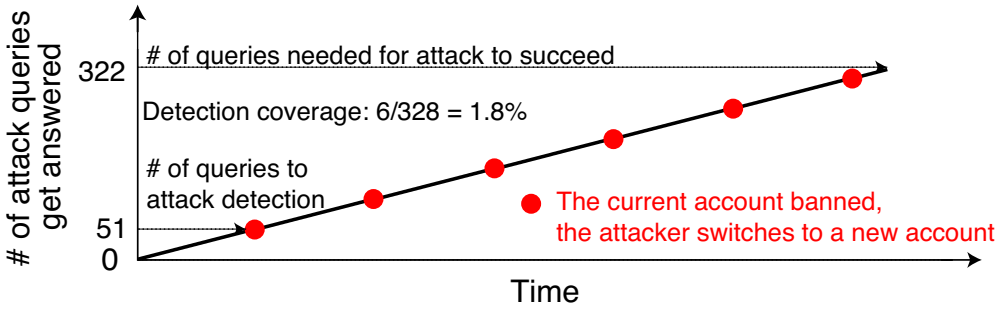


Figure 5.2: *Existing defenses cannot stop persistent attackers who switch accounts to continue attack queries.*

PRADA's runtime complexity grows with the number of prior queries. Consider a query database of 1 million low-resolution images (`CIFAR10`, 32×32), our experiments show that, for each incoming query, SD and PRADA introduce a run-time latency of 24,000% and 6,800% compared to the normal inference latency, respectively (details in §5.8.6). Furthermore, PRADA faces large accuracy drop, because each incoming query produces little impact on the query distribution.

We note that there is also another line of black-box attacks called substitute model attacks as discussed in §2.3 and the standard defense for substitute model attacks, ensemble adversarial training, can be combined with Blacklight as a hybrid defense against both substitute model attacks and query-based attacks (details in the Appendix §C.2).

5.5 Blacklight

We propose *Blacklight*, a new defense to detect and mitigate query-based black-box attacks against DNN models. Different from existing defenses, Blacklight is account oblivious and focuses on detecting individual attack queries on the fly regardless of who sent them. Our design is driven by a fundamental insight that query-based black-box attacks produce queries that are highly similar in the input space. Since benign queries rarely share this level of similarity, these attacks can be detected by identifying extremely high similarity in queries

while incurring low false positives. With this in mind, we design Blacklight to focus on achieving fast, scalable and robust similarity check across millions of image queries. Our design includes two key components: (i) *probabilistic content fingerprinting* for fast and scalable attack detection, and (ii) *salted pixel quantization* to resist adaptive attacks.

In the following, we present the fundamental insight driving our design, and the concept of probabilistic content fingerprinting. Later in §5.6, we describe the salted pixel quantization and Blacklight's detailed design.

5.5.1 Fundamental Insight: Presence of High Similarity in Attack Queries

Blacklight exploits a fundamental insight on query-based black-box attacks: in order to compute adversarial examples, attackers need to perform iterative optimization *over the network*, i.e., submitting one or more queries to the target model, observing the query results, and using them to configure further queries. While the specific design of iterative optimization is algorithm-dependent[3], the unified goal is to repeatedly refine the perturbation such that the query sequence converges to an adversarial example x_n satisfying eq (5.1). Therefore, iterative optimization inevitably produces *some* queries that are highly similar in the input space, i.e.,

$$\text{there exist } \; x_k, x_j, \text{where } \; ||x_k - x_j||_p \leq \mu.$$

If μ is sufficiently smaller than the difference between most benign images, we can accurately detect the attack by recognizing the presence of highly similar queries like (x_k, x_j) within the stream of queries. Evading this type of detection is extremely difficult (if not infeasible) since it requires *every attack query to be sufficiently dissimilar from any previous attack queries*.

We empirically verify the presence of highly similar queries by running the eight SOTA query-based black-box attacks (listed in Table 5.1) on the `ImageNet` classification model. For

3. Some attack designs start with the original input and perturbs it towards a misclassified target label [67, 113], while others start from an image in the target label and work backwards towards the original input [67, 14, 27].

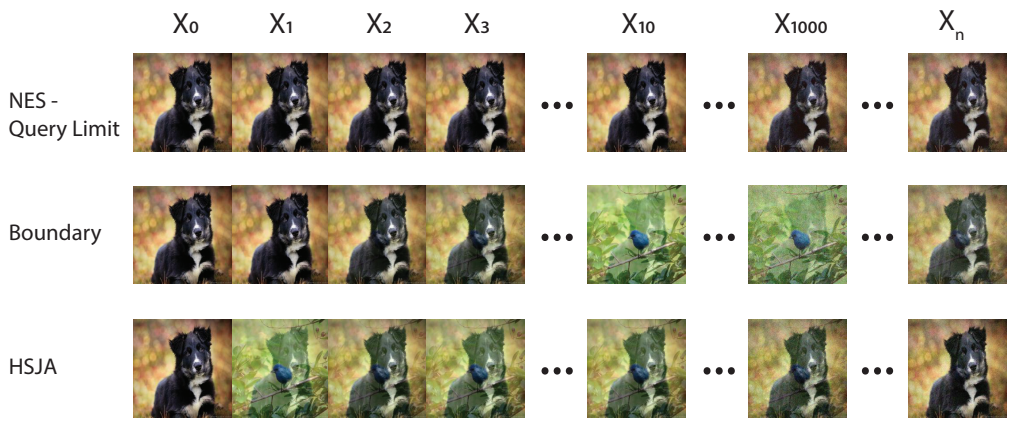


Figure 5.3: *Examples of attack query sequence* $(x_0, x_1, .., x_n)$*, produced by three black-box attacks (NES, Boundary, HSJA). While these attacks generate queries differently, the resulting query sequences all contain some highly similar images.*

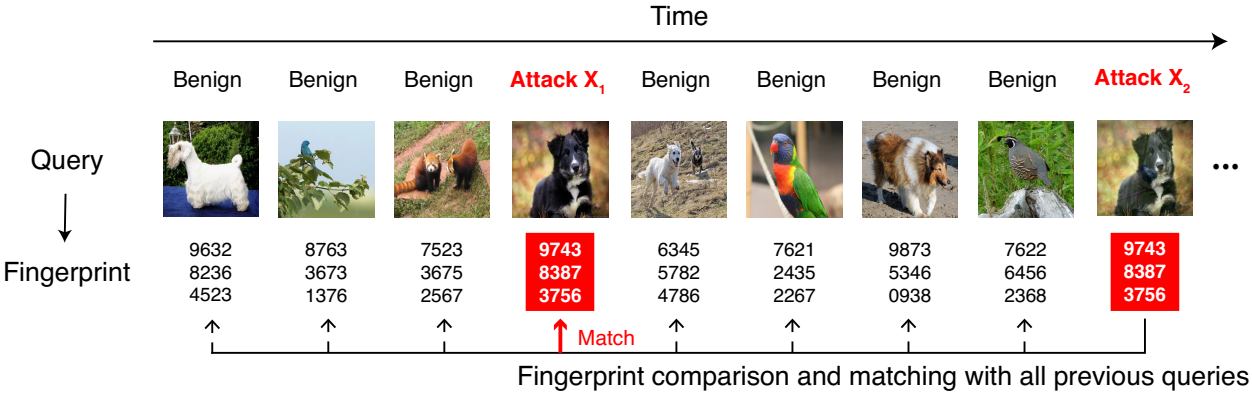


Figure 5.4: *For each raw image, Blacklight computes a small set of hash entries (as its probabilistic fingerprint). Blacklight detects attack images hidden inside a large stream of benign images by comparing and detecting highly similar fingerprints.*

all eight attacks, high similarity is consistently observed across images in their attack query sequence. The average L_2 distance between just consecutive queries in an attack sequence is already 20-380x smaller than analogous distance between benign images (estimated by randomly comparing 2000 pairs of benign images). Figure 5.3 shows some visual examples from attack query sequences generated by three attacks (NES-Query Limit, Boundary, HSJA). We omit the other attacks since they produce similar results.

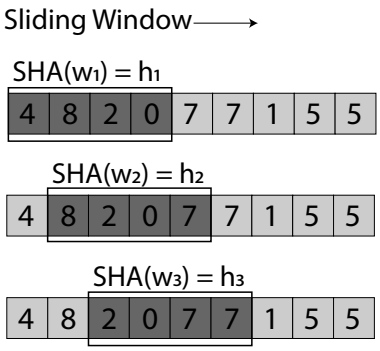


Figure 5.5: Computing content hashes by applying a sliding window over pixels.

5.5.2 Fast and Scalable Similarity Check via Probabilistic Fingerprinting

The above insight motivates us to detect query-based black-box attacks by searching for the presence of highly similar queries in a large stream of incoming and past queries. A key challenge is how to run a fast and efficient similarity check.

Strawman Solutions. We first discuss two strawman solutions and their problems. Earlier in §5.4 we discussed the query similarity check used by SD [31] and its scalability issue.

Computing L_p distances. A naive approach would store all past queries in a database and compare an incoming query x to the entire database of n queries by computing their image-level differences. Such raw comparison incurs heavy costs both in query storage and computation, i.e., $O(n)$. For example, even for low resolution image queries (224×224 pixels, `ImageNet`), it takes 23 minutes to compare a query to one million prior images, even using five threads on a 6-core Intel Xeon server. This is clearly intractable in practice.

Locality-sensitive (LS) hashing. An alternative is to compute a "signature" per query using LS hashes and compare queries by their signatures. Many have used perceptual hashing (*e.g.,* PhotoDNA [4], dhash [76]), a type of LS hashes, to match similar images for copyright resolution or child exploitation detection [4]. Using a hash table for lookup, the runtime cost for checking each incoming query could reach $O(1)$ regardless of n. Unfortunately, these hashes are designed to identify generic variants of an image, even those that have undergone

significant alterations. Thus they flag similar benign queries (*e.g.,* different frames of a video, multiple pictures of the same object) as adversarial, producing false positives. We test dhash [76] on our attack detection and find that it produces over 10% false positives on the Flickr dataset and 67% on our video dataset (§5.8.4). While unable to test PhotoDNA since it is proprietary, we expect that it faces the same issue since it focuses on detecting child exploitation in images which requires considerable alterations.

Probabilistic Fingerprints. Blacklight overcomes these challenges by applying probabilistic fingerprinting to detect highly similar images. Our goal is to design a hash function that is compact yet highly sensitive to very small changes in the image. This dictates that we should use a highly lossy function. Probabilistic fingerprinting achieves these properties and utilizes secure one-way hashes that cannot be easily reversed to evade detection and probabilistic downsampling for efficiency. To fingerprint an image x, Blacklight first transforms x into a set of continuous and overlapping segments of a fixed length $\mathbf{w}$, then applies a one-way hash to each segment to produce a large set of $\mathbf{N}$ hash values. From these $\mathbf{N}$ hash values, Blacklight chooses a small set probabilistically (e.g., the top 50) as x's probabilistic fingerprint.

Figure 5.4 illustrates Blacklight's attack detection process. For an incoming query x, Blacklight extracts its probabilistic fingerprint and stores it in the database. Blacklight runs an efficient hash match algorithm to detect overlaps between x's fingerprint and those in the database. Upon detecting sufficient overlap between x and an existing fingerprint y, it flags (x, y) as a pair of attack queries.

Key Benefits. Our fingerprint scheme has the property that any two highly similar queries will produce a near-perfect match in their fingerprints. In other words, small changes to an image are highly unlikely to impact its fingerprint. The use of secure one-way hash and probabilistic downsampling means that unless they can reverse the hashing algorithm, an adversary cannot alter an image's fingerprint without significantly altering its content (further confirmed in §5.9.1).

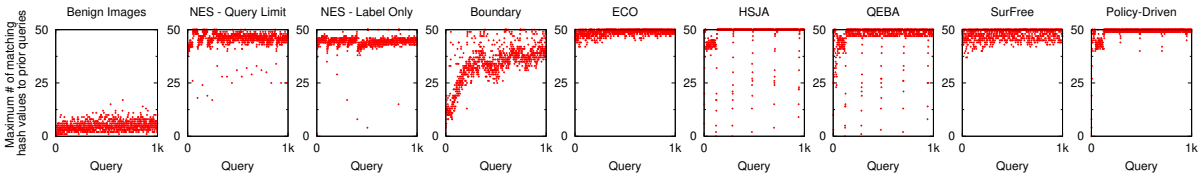


Figure 5.6: *We empirically show that probabilistic fingerprints preserve the query similarity in black-box attack sequences. We plot the* maximum fingerprint overlap *between x and that of any prior query in a benign query sequence (left most) and eight attack query sequences. Here the maximum matching is bounded by $\boldsymbol{S} = 50$.*

Our fingerprints also greatly reduce the storage overhead of past queries, and the computation costs of comparing queries in similarity. Specifically, the search for highly similar queries reduces down to a hash set comparison problem, which takes near-constant time in general (see §5.6).

Prior Work on Probabilistic Fingerprints. Probabilistic fingerprints have been used for similarity detection in text (e.g., detecting code plagiarism [147, 15, 139, 44], network intrusion and malware [148, 138, 118] and spam emails [195, 102]). It was also used in *sif*, a similarity detector for file systems [109]. The contributions of our work include i) extending probabilistic fingerprints beyond the text domain, ii) customizing its design to identify similar image queries to a DNN model (see §5.6), and iii) a formal analysis to model both false positives and attack detection rates and their dependency on fingerprinting parameters (see §5.7).

5.6 Detailed Design of Blacklight

We now present the detailed design of Blacklight, including preprocessing, probabilistic fingerprinting, and comparison algorithms, which together form our proposed detector. We also discuss options to mitigate attacks after detection. Note that Blacklight works as an external add-on, and requires no modifications to the DNN model.

5.6.1 Preprocessing: Salted Pixel Quantization

Given an incoming image query x, Blacklight first runs a quantization function on each pixel of x. This serves two purposes. First, it converts continuous pixel values into a finite set of discrete values, which are then used to compute hashes of x during fingerprinting. Second, quantization increases similarity between (attack) queries. This is particularly true for black-box attacks that iteratively optimize queries by gradually modifying every single pixel on the image [67, 27, 14] – the use of quantization effectively nullifies changes to image hashes created by these minor modifications without inducing false positives. We confirm this empirically in Figure C.2 where the hash overlap between attack queries (on `CIFAR10`) increases rapidly with the quantization step $\mathbf{q}$ to approach 100%, while those between benign queries remain low. Note that this step is used only for attack detection. If the input is considered benign, the original, unaltered query is sent to the DNN model.

Furthermore, Blacklight employs a *salted* pixel quantization function to resist reverse engineering attacks:

$$Q(x, salt_Q, \mathbf{q}) = \lfloor \frac{(x + salt_Q) \mod 255}{\mathbf{q}} \rfloor \tag{5.2}$$

where $salt_Q$ is a randomly generated salt image (of the same dimensions as x) and $\mathbf{q}$ is the quantization step (a system parameter). Here all pixel values of x and $salt_Q$ are normalized to $[0, 255]$. Later in §5.9 we show adding a random salt improves Blacklight's robustness against adaptive attacks.

5.6.2 Computing Probabilistic Fingerprints

We now describe the detailed process to compute the probabilistic fingerprint on a (quantized) query image x.

Converting an image into N segments. Blacklight first "flattens" the 2D image into a single pixel sequence by concatenating rows of pixels together; then applies a sliding window

of fixed size $\mathbf{w}$ on this sequence, iteratively moving the sliding window by $\mathbf{p}$ (referred to as the sliding step). This produces $\mathbf{N} = (|x| - \mathbf{w} + \mathbf{p})/\mathbf{p}$ overlapping pixel segments, each of length $\mathbf{w}$. Any two consecutive segments overlap by $\mathbf{w} - \mathbf{p}$ pixels, and each pixel in x is included in $\mathbf{w}/\mathbf{p}$ segments.

Hashing each segment. For each segment i ($i \in [1, \mathbf{N}]$), Blacklight applies a secure one-way hash function (e.g., SHA-3 combined with a random salt value chosen by the defender) and produces a hash value h_i. This creates a full hash set $\mathbf{H}_x = (h_1, h_2, ..., h_{\mathbf{N}})$ for query x, with $\mathbf{N}$ hash entries. For example, for `CIFAR10` ($|x| = 32 \times 32 \times 3 = 3072$), $\mathbf{N} = 3053$ when $\mathbf{w} = 20$, $\mathbf{p} = 1$. An illustration of this sliding window hashing scheme is shown in Figure 5.5.

Selecting a subset of hashes as the fingerprint. From x's full hash set $\mathbf{H}_x$, Blacklight selects the top $\mathbf{S}$ hash values (sorted by *numerical order*) as its probabilistic fingerprint, denoted as $\mathbb{S}(\mathbf{H}_x)$. Since the output distribution of the one-way hash is random, choosing the top $\mathbf{S}$ hash values by numerical order serves as an efficient downsampling algorithm that is deterministic[4] to the defender but unpredictable to an adversary (since predicting the top $\mathbf{S}$ hash values requires predicting the full hash set).

The use of probabilistic fingerprinting puts a *hard* limit on the overhead of fingerprint storage and comparison, while preserving the high similarity among attack queries. Figure 5.6 shows a sample measurement on query similarity, for the eight black-box attacks discussed in §5.2. Here we measure, for each query x_i in an attack sequence, the maximum number of matching hashes between x_i's fingerprint and any of its prior queries in the same sequence. For reference, we also compute the number of matching hashes among benign images. We see that many attack queries display fingerprints highly similar to at least one prior query in the same sequence, while benign queries share minimal overlap in fingerprints. Thus Blacklight can quickly detect black-box attacks after seeing only a small number of queries.

4. Deterministic means that the downsampled hash set holds the same property of the full hash set: highly similar (quantized) queries will have highly similar fingerprints. We also verified this empirically in Figure C.3.

5.6.3 Comparing and Matching Fingerprints

Upon receiving a new query x, Blacklight computes its fingerprint $\mathbb{S}(\mathbf{H}_x)$ and compares it to all prior fingerprints stored in the database. If any stored fingerprint shares more than $\mathbf{T}$ hash entries with $\mathbb{S}(\mathbf{H}_x)$, then x is flagged as an attack image. Here, the value of $\mathbf{T}$ can be configured to meet the desired false positive rate. Later in §5.7, we analytically show that by properly configuring $\mathbf{T}$ and $\mathbf{S}$, we can achieve accurate attack detection at a low false positive rate.

Computing the maximum overlap between the fingerprint of a query and n stored fingerprints is non-trivial. A simple algorithm would incur computation cost of $O(n)$. We use a better algorithm which stores a query x's fingerprint into a hashmap using each of its hash entry as a key. The maximum overlap with all n queries can be found by retrieving all queries associated with each key in x's fingerprints, and counting the max frequency of any query in that set. An efficient implementation can produce average runtime that is a constant independent of n. We leave the design and analysis of an efficient hashset matching algorithm to future work. We present detailed performance overheads in §5.8.6.

5.6.4 Mitigating Attacks after Detection

Detecting the presence of a query-based black-box attack is just a first step in protecting DNN models. A persistent attacker can simply switch accounts and/or IP addresses and continue with additional queries. Here, we discuss options for mitigation after an attack is detected.

Ban accounts. As a response, banning an account or blocking an IP address is not ideal. First, it means each false positive incurs a high penalty, which might be undesirable in some application settings. Second, this does little to deter resource rich attackers, who can continue the attack using Sybil accounts, which are difficult to eradicate in practice.

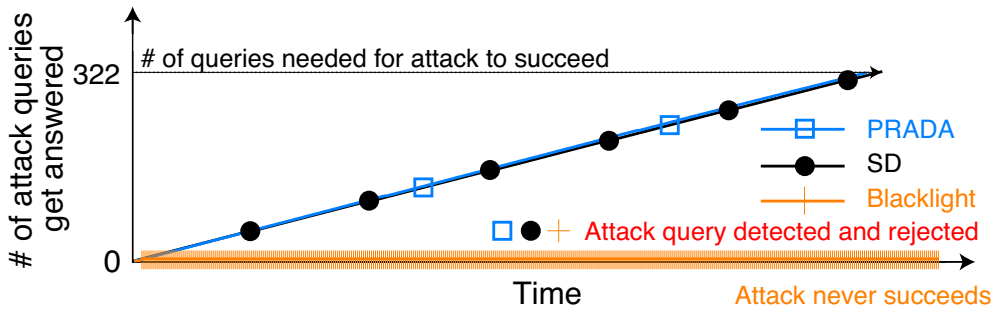


Figure 5.7: *By detecting/rejecting most of attack queries (regardless of account usage), Blacklight effectively resists persist attackers, which existing defenses fail to address.*

Return misguided outputs. We also consider a more elaborate scheme where the defender intentionally misleads the attacker by returning carefully biased query outputs, perhaps towards secondary goals like identifying the attacker. This approach faces additional challenges. First, crafting biased responses requires significantly more computation and state-keeping at the defender. Second, the defender must be careful to avoid returning valid responses to actual attack queries.

Reject all detected queries. Ultimately we chose a simple strategy: reject all detected attack queries. This mitigation is effective in preventing attacks *IFF* the ratio of attack queries detected is high. If most attack queries are rejected, the attack sequence takes a very long time to converge and succeed. The benefit of this approach is that it does not rely on detecting or reducing Sybil accounts, and false positives have minimal impact on benign users.

In §5.8, we evaluate the impact of mitigation on persistent attackers who continue to submit attack queries after query rejection. Figure 5.7 provides a preview in terms of the # of attack queries got answered under Blacklight, using the persistent attack trace of Figure 5.2. Blacklight rejects almost all the attack queries, preventing the attack from making progress.

5.7 Formal Analysis

We formally examine Blacklight by modeling the process of probabilistic fingerprinting. We derive analytical bounds on the probability of Blacklight flagging a query pair (x, y) as attacks $Q(\Delta)$ as a function of the full hash difference between the two, i.e., $\Delta = \text{diff}(\mathbf{H}_x, \mathbf{H}_y)$. We then estimate Blacklight's false positive rate and attack detection rate by $Q(\Delta_{benign})$ and $Q(\Delta_{attack})$. Here Δ_{benign} is the minimum full hash difference between benign queries and Δ_{attack} is the maximum full hash difference between attack queries. Our key results: are: (i) $Q(\Delta)$ decays fast with Δ, (ii) Blacklight can effectively detect attacks at a low false positive rate: $Q(\Delta_{attack}) \to 1$, $Q(\Delta_{benign}) \to 0$, if $\Delta_{benign} >> \Delta_{attack}$, (iii) the analytical bound on $Q(\Delta)$ can guide the selection of Blacklight's configuration parameters ($\mathbf{w}$, $\mathbf{p}$, $\mathbf{q}$, $\mathbf{S}$ and $\mathbf{T}$). For brevity, we leave the details to Appendix§C.1.

5.8 Experimental Evaluation

Using four different image classification tasks (and datasets), we empirically evaluate Blacklight against eight SOTA black-box attacks. Our experiments seek to understand 1) the effectiveness of Blacklight in both attack detection and mitigation; 2) the false positive rate under realistic settings; 3) impact of Blacklight configuration; 4) Blacklight's storage and computation cost; 5) applying Blacklight to other domain.

5.8.1 Experimental Setup

We apply Blacklight to protect DNN models developed for image classification. Our experiments cover a wide range of input size/content and model architectures, allowing us to evaluate Blacklight under a diverse set of conditions.

Image Classification Tasks. We consider four representative tasks: `MNIST` [86], `GTSRB` [150], `CIFAR10` [77] and `ImageNet` [141]. We summarize in Appendix §C.5 these

tasks and associated models in Table C.2, and detailed model architectures and training configurations in Table C.3 to C.5.

Task	Attack	w. Detection			w. Mitigation	w/o Blacklight	
		Attack detect %	Detection coverage	Avg queries to detection	Attack success	Attack success	Avg # attack queries
MNIST	NES - QL	100%	99.5%	2	0%	45%	66540
	NES - LO	100%	99.0%	2	0%	1%	95973
	Boundary	100%	64.2%	18	0%	21%	85467
	ECO	100%	99.9%	2	0%	43%	52780
	HSJA	100%	98.1%	6	0%	59%	9924
	QEBA	100%	98.4%	8	0%	92%	12141
	SurFree	100%	97.9%	7	0%	84%	10034
	Policy-Driven	100%	99.0%	8	0%	74%	9538
GTSRB	NES - QL	100%	98.5%	2	0%	66%	48429
	NES - LO	100%	98.0%	3	0%	17%	83823
	Boundary	100%	64.3%	22	0%	37%	76643
	ECO	100%	100.0%	2	0%	80%	27782
	HSJA	100%	98.2%	5	0%	95%	10392
	QEBA	100%	99.5%	8	0%	99%	9832
	SurFree	100%	98.3%	8	0%	98%	9192
	Policy-Driven	100%	98.1%	5	0%	100%	13021
CIFAR10	NES - QL	100%	98.3%	2	0%	100%	12621
	NES - LO	100%	98.7%	2	0%	89%	67126
	Boundary	100%	64.4%	25	0%	95%	6082
	ECO	100%	99.4%	2	0%	89%	16887
	HSJA	100%	97.1%	7	0%	100%	1205
	QEBA	100%	96.9%	6	0%	99%	1009
	SurFree	100%	96.8%	8	0%	100%	1396
	Policy-Driven	100%	97.3%	7	0%	100%	1198
ImageNet	NES - QL	100%	99.4%	2	0%	99%	11201
	NES - LO	100%	98.2%	2	0%	20%	63492
	Boundary	100%	95.1%	42	0%	74%	67356
	ECO	100%	99.6%	2	0%	93%	11304
	HSJA	100%	98.7%	7	0%	99%	12402
	QEBA	100%	98.3%	6	0%	100%	10293
	SurFree	100%	97.6%	7	0%	100%	8783
	Policy-Driven	100%	99.1%	8	0%	100%	10368

Table 5.2: *Blacklight's detection and mitigation results. In the last two columns, we included attack performance in absence of Blacklight: attack success rate and average attack queries required to complete an attack.*

Attack Configurations. We implement and run the eight black-box attacks list in Table 5.1 against each of the above four classification models. For `MNIST`, `GTSRB` and `CIFAR10`, we randomly select 1000 images from their test datasets and use each as the source image of the attack (*i.e.* x_0). For `ImageNet`, we randomly select 500 source images (due to its higher

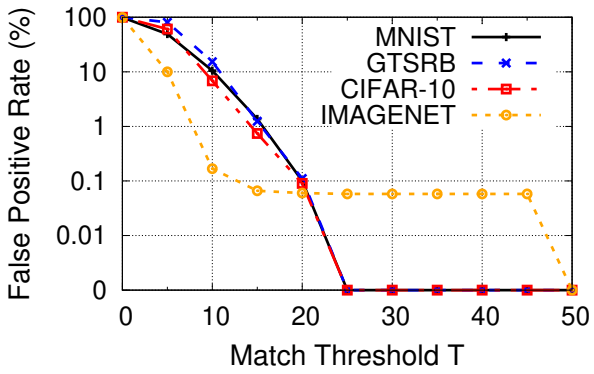


Figure 5.8: *Blacklight's false positive rate when fixing* $\mathbf{S} = 50$ *and varying* $\mathbf{T}$.

Label	# of Filtered	FPR	Label	# of Filtered	FPR
balloon	953	0.07%	packet	1158	0.21%
boathouse	1572	0.73%	peacock	556	0.68%
daisy	656	0.10%	pier	309	0.07%
fly	188	0.03%	rifle	905	0.48%
geyser	896	0.11%	snail	350	1.01%
hay	1192	0.79%	swing	510	0.48%
knot	817	0.14%	teapot	1715	0.14%
menu	1232	0.37%	tiger cat	1315	0.28%
mortar	1229	0.38%	toaster	3298	0.37%
nail	1696	0.83%	vault	182	0.04%

Table 5.3: *Blacklight's false positives on benign images crawled from Flickr. "# of Filtered" is # of images that are duplicated and have the same hash value with prior queries; "FPR" is the false positive rate per label. For each label, we run Blacklight on 80,000 Flickr images (crawled via this label).*

computation cost). We run each attack until it terminates (i.e., successfully generating an adversarial example) or reaches 100K queries, whichever occurs first.

When configuring each attack, we follow its original paper and use the same L_p distance metric (L_2 or L_∞) stated in the paper. Since L_2 distance depends on model input size, we use the *normalized* L_2 distance $\sqrt{\frac{1}{|x|}\sum_{i=0}^{|x|}(x_i - x'_i)^2}$. The detailed attack parameters and distance metrics are listed in Table C.6.

For all these attacks, we set the perturbation budget ϵ such that most attacks succeed in absence of defenses. As reference, the standard ϵ for white-box attacks is 0.03 for L_∞ and <0.03 for *normalized* L_2 [23]. Black-box attacks should use a larger budget because they are

naturally harder to succeed. In fact, our experiments on the eight SOTA black-box attacks confirm that a budget of 0.03 leads to significant attack failures. Thus we increase ϵ=0.05 for both L_∞ and *normalized* L_2 to allow most attacks to succeed. The only exceptions are L_∞ attacks against `MNIST` since ϵ=0.1 is necessary for them to succeed. The perturbation budgets are listed in Table C.6.

Blacklight Configuration. Table C.8 in Appendix lists the default values for Blacklight's key parameters: sliding window size (**w**), sliding step (**p**), quantization step (**q**), # of hash entries per fingerprint (**S**), and fingerprint matching threshold (**T**). To demonstrate the generality of Blacklight, we set these parameters to be the same default values for all four tasks, rather than "optimizing" them per task. The only exception is **w** – our default value is 20, but we increase it to 50 for `MNIST` (due to its large black background) and `ImageNet` (due to its large image size).

We choose these values following our formal analysis. In particular, we choose $\mathbf{T} = \mathbf{S}/2 = 25$ by modeling how **T** affects false positive and detection coverage. Figure 5.8 shows the measured false positive rates when varying **T**, confirming that **T**=25 achieves less than 0.1% false positive for all four tasks. In §5.8.5, we further explore the impact of parameter configuration by varying **w**, **p**, **q** and **S**.

Evaluation Metrics. We use the following metrics to quantify the effectiveness and cost of Blacklight.

- **False positive rate**: % of benign queries detected as attack.
- **Attack detection rate:** % of black-box attacks detected before the attack completes.
- **Detection coverage:** % of queries in an attack's query sequence identified as attack queries.
- **Avg # of queries to detection:** Average # of attack queries accepted (thus answered) before detecting an attack query.

- **Attack success rate w. mitigation:** Success rate of a persistent attack when all detected attack queries are rejected.
- **Detection overhead:** Run-time latency and storage costs.

5.8.2 Attack Detection and Mitigation

We evaluate Blacklight's detection rate by implementing and performing each of the eight black-box attacks against each classification model. For each attack and task combination, we run 1000 instances of the attack (500 for `ImageNet`). Each attack instance selects a random image from the test dataset as source image of the attack (x_0), and a random incorrect label as the misclassification target label.

The results for all attacks are listed in Table 5.2. As reference, the last two columns report the performance of these attacks *without* the Blacklight defense, in terms of attack success rate and the speed of convergence (# of queries before successfully producing an adversarial example). We see that recent attacks, especially HSJA, QEBA, SurFree, Policy-Driven, are highly successful in absence of Blacklight. Boundary and NES-LO take the longest time to converge. Some attack instances do fail to converge even after generating 100k queries (e.g., less than 50% of NES-LO complete in 100K queries for MNIST, GTSRB and ImageNet). Most of them remain unsuccessful even when increasing the query bound to 300k. Overall, a successful attack takes several thousands to tens of thousands of queries to complete.

Next, we summarize key results on Blacklight's attack detection (as shown by column 3-5 in Table 5.2). We see that the attack detection rate remains 100% for all attack instances, indicating that Blacklight detects *all* attacks on all models in progress. The detection coverage is also extremely high – Blacklight detects more than 96% of all attack queries, except on the Boundary attack. Another key observation is that Blacklight detects a new attack instance very quickly, often after a handful of 2–8 queries (again, more queries required for Boundary because it converges slower). In all cases, Blacklight detects an attack in less than 1% of the

average number of queries required to complete the attack.

Blacklight detects Boundary slower than others. This is because Boundary advances slower in shrinking perturbation towards the L_p ball of the target, thus Blacklight detects them at a "later" stage with 100% detection rate. The three improved versions of Boundary (HSJA, QEBA, Policy) converge faster, thus Blacklight detects them faster. To further evaluate the slower Boundary attack, we run the attack for 1 million queries. We find Blacklight continues to detect (and reject) attack queries in this longer sequence, leaving the attacker with 0% success (for all four tasks). The detailed results are listed in Table C.9 in Appendix.

Finally, column 6 in Table 5.2 reports the attack success rate when Blacklight rejects queries identified as attack queries. We see that none (0%) of persistent attackers manage to complete their attack within 100K queries. Blacklight's mitigation is highly effective because it is able to detect nearly all attack queries. Rejecting these queries prevents the attacker from making forward progress in probing model classification boundaries. This confirms that a high detection coverage is critical to defend against query-based black-box attacks.

Key Takeaways. Our results against eight SOTA black-box attacks show that Blacklight detects all attacks on all models, detects the overwhelming majority of queries in the attack sequence, and detects the attack quickly (usually in less than 8 queries, with the exception of the slow converging Boundary attack). Furthermore, by rejecting all detected attack queries, Blacklight's mitigation module ensures no attacks can complete (at least in 100K queries) for all our tested attacks and target DNN models.

Comparison to Existing Defenses. As reference, we show the performance of SD and PRADA in Table C.1, using the same attack experiments described above. As discussed in §5.4, SD and PRADA are not designed to stop persistent attackers who switch account to continue attack. Results in Table C.1 confirm this and their low detection coverage (0.8%-2.1%).

5.8.3 Detecting Universal Patch Attacks

We evaluate Blacklight against the only known query-based universal patch attack, Sparse-RS [40]. Table C.10 in Appendix shows that Blacklight is also highly effective in detecting Sparse-RS (100% detection success rate and $> 97.6\%$ detection coverage). Since query-based universal patch attacks are emerging, additional work is required to thoroughly evaluate the robustness of Blacklight against them.

5.8.4 False Positives in Real World Settings

Since Blacklight relies on a similarity detection algorithm to detect attacks, one might wonder if duplicates or near-duplicates of images will trigger false positives. Figure 5.8 reports its false positives between distinctive inputs. But what about "naturally" similar images, such as different versions of the same image, or closeby frames of the same video?

We begin with a simple test to confirm that naturally occurring false positives are very low in large image repositories like ImageNet. We turn off database resets, randomly sample 1 million images from ImageNet training data, send them as queries to Blacklight, and observe a very low false positive rate of 0.37%.

False Positives in Similar Images. Next, we look at similar images of the same objects, e.g. inputs that should classify to the same labels. We crawl a large number of public real world images from Flickr [183] using keyword search on their public API. We pick 20 random labels from ImageNet, and use each as a search keyword to crawl $80,000$ images for that label. We filter out images that are perfectly identical at the pixel level (we found an average of 1036 ± 696 duplicate images per label). We then take each label, and run our $80,000$ images as queries to Blacklight. Even across Flickr images labeled with the same keyword, Blacklight produces a very low false positive rate of $0.37\% \pm 0.29$ over 20 labels. Detailed results for all labels are shown in Table 5.3.

False Positives in Video Frames. Finally, we consider the scenario where the system

might receive benign queries that are highly similar by nature, e.g. image stills taken from video frames. We explore how Blacklight responds under such scenarios by testing it for false positives on the YouTube Faces dataset [175]. YouTube Faces is a collection of $3,425$ videos of $1,595$ different people, designed for studying unconstrained facial recognition. We use common image extraction techniques [164] to extract $587,137$ video frame images from videos for $1,283$ celebrities. Of these, we filter out $33,227$ images that are pixel-level identical to other images, and send the remaining video frames to Blacklight. The result is a false positive rate of 1.74%. Even if Blacklight takes over half million queries per reset cycle for the highly similar queries, the false positive rate is still very low.

5.8.5 Impact of Parameter Configuration

As discussed in AppendixC.1, we leverage our formal analysis of Blacklight to configure its five system parameters: $\mathbf{w}$, $\mathbf{p}$, $\mathbf{q}$, $\mathbf{S}$, and $\mathbf{T}$. Earlier in Figure 5.8 we show empirically how Blacklight's false positive rate varies with $\mathbf{T}$ and verify our strategy on configuring $\mathbf{T}$. In the following, we study the impact of the other four parameters by testing Blacklight against the same set of attacks while varying each of these parameters. We report the false positive rate and detection coverage since the attack detection rate is always 100%. The detailed results are listed in Figure C.4 in Appendix.

We summarize the key findings below. First, we confirm that $\mathbf{q}$ is a critical parameter for Blacklight – the detection coverage increases quickly as $\mathbf{q}$ goes from 1 (no quantization) to 50 (the default value) and stabilizes after that (except for Boundary). When $\mathbf{q}$ approaches 100, we start to see visible increase in false positives (>0.1%). Second, as expected, the sliding window size $\mathbf{w}$ is negatively correlated to false positive rate and detection coverage, while the sliding step $\mathbf{p}$ has little impact (note that $\mathbf{p} < \mathbf{w}$). Thus Blacklight should select $\mathbf{w}$ as a small value to meet the desired false positive rate. Finally, as expected $\mathbf{S}$ should be small to reduce complexity but not too small (e.g., <20) to introduce visible false positives. Overall, these

results confirm our proposed theory-guided principle for choosing Blacklight's parameters.

5.8.6 Overhead of Blacklight

Storage. Blacklight requires a database to store fingerprints of prior queries. Our probabilistic fingerprints are extremely small. Across all of our experiments, a fingerprint is $\leq 32 \cdot \mathbf{S}$ bytes and 1.6KB for the default configuration in Table C.8. A database of 1 million queries only requires 2GB storage, a "negligible" value for modern servers.

Runtime. Blacklight's per-query runtime includes latency to generate the fingerprint from a query and latency to lookup the fingerprint in the query database. The former depends on the image size and the parameters ($\mathbf{w}$, $\mathbf{p}$) and the latter depends on the size of query database n. We configure Blacklight to its default configuration and explore the impact of sliding step $\mathbf{p}$ (i.e., increasing $\mathbf{p}$ from 1 to 10 or 25 to speed up hash computation) and the query database size n. We run Blacklight on an Intel i7 desktop server with 64 GB memory, and report the per-query runtime for two types of query images (32×32,`CIFAR10`) and (224×224, `ImageNet`) in Figure 5.9 as a function of n. The curves remain flat over n, suggesting that Blacklight's detection cost is independent of n. More specifically, a `CIFAR10` model inference takes 50ms (on a Nvidia Titan RTX) while Blacklight (on Intel i7) takes 4-8ms (8%-16% over 50ms) for n=1 million queries.

As reference, we compute the runtime of SD and PRADA on the same Intel i7 server, putting all n queries into a single account. They only run on `CIFAR10`, which we report in Figure 5.9. The latencies scale linearly with n (note the log Y axis). For n=1 million queries, SD and PRADA take 12s and 3.4s per query (24,000% and 6,800% over inference).

Further optimization. Blacklight's per query latency is dominated by the sliding window-based hash computation (99% of total runtime). We further optimize this computation using GPUs. A modified version of Blacklight running a Nvidia Titan RTX reduces per-query latency by 20x, to 0.4ms for `CIFAR10` and 20ms for `ImageNet`, almost "negligible" compared

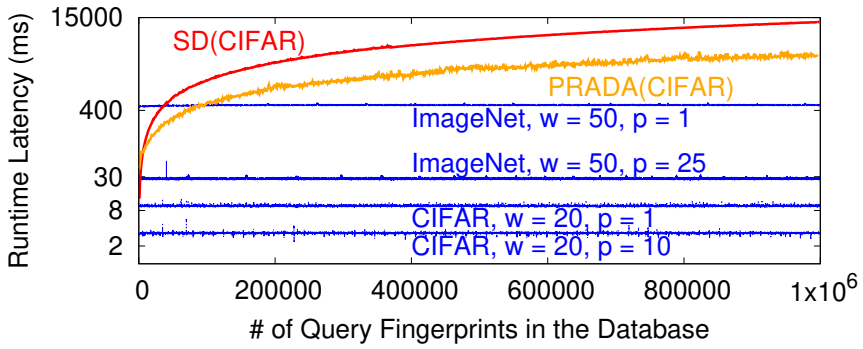


Figure 5.9: *Blacklight's runtime latency vs. n. Note the log Y axis. We include latency of SD and PRADA for reference*

.

to the inference latency.

5.8.7 Blacklight for Text Classification

Blacklight should in principle extend to other domains where black-box adversarial attacks produce highly similar queries in the input space. The domain-specific design task is how to generate query fingerprints to enable efficient and accurate detection. Below, we show an initial Blacklight design for text classification, a critical task in NLP. DNN-based text classification is shown to be vulnerable to query-based black-box attacks [91, 71, 107, 48], with three SOTA attacks: TextFooler [71], TextBugger [91] and HardLabel [107].

Fingerprinting a sentence. The input to a text classifier is a sentence, from which Blacklight produces a fingerprint. First, we convert the sentence into an array by replacing each word with its word embedding. We quantize the array, apply a sliding window to move through the quantized array and compute hashes, and select the top **S** hashes as the query fingerprint. The parameter choices are listed in Table C.8 for `IMDB` text queries. **S** and w are smaller since text sentences create "shorter" arrays, while **T** remains **S**/2.

Blacklight performance. We run Blacklight on the three SOTA attacks on the `IMDB` dataset [104]. The results in Table 5.4 show that Blacklight achieves 100% detection rate and >99.7% detection coverage, only takes 2 queries to detect an attack (and reject the second

Attack	w. Detection			w. Mitigation	w/o Blacklight	
	Attack detect %	Detection coverage	Avg queries to detection	Attack success	Attack success	Avg # attack queries
TextBugger [91]	100%	99.7%	2	0%	86.0%	537
TextFooler [71]	100%	99.7%	2	0%	100.0%	669
Hard Label [107]	100%	99.9%	2	0%	100.0%	4642

Table 5.4: *Blacklight's detection and mitigation results on query-based black-box attacks for text classification.*

query). As such, no attack ever succeeds. For all of these tests, the false positive rate is only 0.49%. Overall, these results offer clear evidence that Blacklight can potentially generalize to other domains using the same probabilistic fingerprint methodology.

5.9 Adaptive Attacks

A meaningful defense must be robust against adaptive countermeasures from attackers with full knowledge of the defense. We explored a number of customized adaptive attacks against Blacklight, and present the strongest countermeasures, organized into three groups: 1) reducing query similarity for attack sequences, 2) reducing queries needed for successful attacks and 3) leveraging resets in Blacklight. Given the similarity between the attacks, we only apply countermeasures to 5 of 8 attacks: NES (QL & LO), Boundary, ECO and HSJA.

5.9.1 Reducing Query Similarity

With knowledge of how Blacklight works, the straightforward adaptive attack is to evade detection by reducing similarity between attack queries. Below we present four types of adaptive attacks that add perturbations to attack queries to reduce similarity between them.

Evasion via Image Transformations. An attacker can try to evade detection by adding additional perturbations to attack queries, where ideally these perturbations do not disrupt the iterative optimization process, but are significant enough to make fingerprints of attack queries different. We explore two types of image transformations: 1) adding Gaussian noise,

Attack Type	Default $\mathbf{T} = 25$ (FPR = 0.0%)				$\mathbf{T} = 15$ (FPR = 0.74%)			
	0.05	0.1	0.15	0.2	0.05	0.1	0.15	0.2
NES - QL	100%	100%	100%	100%	100%	100%	100%	100%
NES - LO	100%	100%	100%	100%	100%	100%	100%	100%
Boundary	100%	100%	75%	40%	100%	100%	100%	95%
ECO	100%	100%	100%	100%	100%	100%	100%	100%
HSJA	100%	100%	55%	40%	100%	100%	80%	40%

Table 5.5: *Blacklight detection rate for attacks using larger perturbation budgets (0.1-0.2) for* `CIFAR10`. *Lowering* $\mathbf{T}$ *largely improves detection when attackers operate on very large perturbations, with a small increase in false positives.*

and 2) applying image augmentation like shift, rotation, zoom and blending. We apply these transformations to attack queries and send them to Blacklight. We first examine how these transformations affect the attack in absence of Blacklight, and confirm that they do introduce different levels of disruptions (none to 100%). On the other hand, for all the transformed attack sequences that will lead to a successful attack in absence of Blacklight, Blacklight detects all of them, i.e., 100% attack detection rate. Further details are in Appendix§ C.7.1 and Table C.11.

Increasing Learning Rates. The attacker can also try to increase dissimilarity between consecutive queries by tweaking their learning rate parameter. Learning rate controls the difference between two adjacent queries when estimating gradients. This does not apply to gradient estimation free attacks (Boundary and ECO). We only explore different learning rate for NES-QL, NES-LO and HSJA attacks. For two variants of NES, we gradually increase learning rate more than 1000 fold. While the attack success rate drops to 0%, detection success rate remains 100%. For HSJA, we gradually grow learning rate up to a factor of 10^6, until changes in learning rate no longer impact gradient estimation results. Here, attack success rate steadily drops (eventually to 15%), but detection remains at 100% throughout.

Increasing Perturbation Budgets. Our evaluation so far assumes the attacker's perturbation budget is limited to commonly accepted values: 0.05 for both L_∞ and *normalized* L_2. Future attacks might tolerate a higher perturbation budget in specific settings. Thus, we evaluate Blacklight's detection performance against attacks on `CIFAR10` with larger

Task	Blacklight's $salt_Q$ off			$salt_Q$ on			attacker knows $(\mathbf{p}, \mathbf{q}, \mathbf{w})$, $salt_Q$ on		
	Boundary	ECO	HSJA	Boundary	ECO	HSJA	Boundary	ECO	HSJA
`MNIST`	0%	0%	0%	0%	0%	0%	0%	0%	0%
`GTSRB`	10%	5%	5%	0%	0%	0%	0%	0%	0%
`CIFAR10`	20%	15%	25%	0%	0%	0%	0%	0%	0%
`ImageNet`	5%	10%	20%	0%	0%	0%	0%	0%	0%

Table 5.6: *Attack success rate using guided transformations attacks.*

Metrics	NES	AutoZOOM
Attack success %	100%	100%
Attack detect %	100%	100%
Detection coverage	99.1%	98.9%
Avg queries to detection	2	2
Avg # of attack queries	1473	1240

Table 5.7: *Blacklight vs. hybrid batch attacks.*

perturbation budgets. For all attacks, we incrementally increase the budget from 0.05 all the way up to 0.2, and measure Blacklight's attack detection rates for each budget level (running 20 attack instances for each data point). We show that the attack detection rates for NES variants and ECO remain steady at 100%, but Boundary and HSJA begin to evade detection when *normalized* $L_2 = 0.15$ (or $L_2 = 8.3$).

Blacklight can compensate by tuning the fingerprint matching threshold $\mathbf{T}$. Table 5.5 shows that by lowering $\mathbf{T}$ from 25 to 15, we can dramatically increase detection rates, restoring perfect detection to most attacks (except HSJA at *normalized* $L_2 = 0.15/0.2$ ($L_2 = 8.3/11.1$) and Boundary at *normalized* $L_2 = 0.2$ ($L_2 = 11.1$)). This drop in $\mathbf{T}$ only increases false positive rates by 0.74%.

We further validate our results on the other three tasks for the two fastest converging attacks (ECO and HSJA) and the results (Table C.12) are consistent with `CIFAR10`. Finally, we also perform analysis on the L_2 distances between benign images to provide a baseline for reasonable L_2 budget for adversarial attacks in Appendix §C.7.2.

Evasion via Guided Transformation. Beyond first order adaptive attacks, we worked hard to design more powerful attacks specifically targeting Blacklight. Assuming a Blacklight system's parameters $\mathbf{q}$ and $\mathbf{w}$ are unknown to an attacker, the strongest attack we could

design is the two-pronged reverse engineer attack, where an attacker first uses queries to probe the limits of **q** and **w**, and then leverages those results to optimize a guided transformation attack.

The high-level intuition is that an attacker can optimally spread out their perturbation budget across the image, if they understand Blacklight and learned its specific configuration parameters. As long as there is at least one pixel changed (after pixel quantization) for some sliding window, hash values of the window will be changed. Thus, the attacker just needs to make sure that for each window, at least one pixel is different from all prior queries after quantization. In this case, Blacklight's use of $salt_Q$ in eq (5.2) is crucial to resisting these guided transformation attacks. Next, we summarize the attack and results when Blacklight turns $salt_Q$ off or on.

Guided transformation (Blacklight's $salt_Q$ off). An attacker begins by estimating quantization step **q** and using it to compute quantization boundary B, followed by estimating value of **w**. It does this by issuing pairs of queries with a minimal perturbation based on an initial estimate of **q** or **w**, and observing whether the second query is detected as an attack. This is repeated using binary search until both **q** and **w** are determined. Finally, the attacker computes B from **q**, and then the optimal layout of modified pixels to maximize the number of substring windows affected by the perturbation. The attacker uses this process to modify each query to evade detection while iteratively optimizing queries to generate the adversarial example. We implement this attack on top of the two fastest converging attacks (ECO and HSJA) and the slowest attack (Boundary). Table 5.6 shows that the attacker achieves no more than 25% success rate for all tasks.

Guided transformations (Blacklight's $salt_Q$ on). The defender can overcome the above adversary by making it harder to extract the quantization boundary. Blacklight does so by adding a "salt" to the quantization process, i.e., $salt_Q$ in eq. (5.2). This defeats attempts by the attacker to reverse engineer **q** and B. Without knowledge of **q**, an attacker can still

launch a weaker version of the attack, but must overshoot on perturbation to increase chances of it persisting through the salted quantization and alter the hashes. We implement such attack by altering 5, 10, and 15 out of every 20 pixels within the perturbation budget. When applying this new attack on top of ECO, HSJA, and Boundary, the attacker still achieves 0% success on all tasks, while Blacklight maintains a high detection coverage (78%). This confirms the significant robustness gained by adding the salt.

Guided Transformations when Attacker Knows $(\mathbf{q}, \mathbf{p}, \mathbf{w})$. Finally, we consider the strongest guided transformation attack – the attacker knows the exact values of $\mathbf{q}$, $\mathbf{p}$, $\mathbf{w}$ and can better perturb queries to evade detection.

To make a query x evade detection, the attacker must ensure that for each window, at least one pixel of x is different from all prior queries after quantization. This is because Blacklight's one-way hash distribution and the top $\mathbf{S}$ hash choices remain unpredictable to the attacker. Knowing $\mathbf{q}$, $\mathbf{p}$, $\mathbf{w}$ helps the attacker to optimize the pixel perturbation. For example, now in each window changing a pixel by $\mathbf{q}$ or $-\mathbf{q}$ will change the hash despite the use of $saltQ$. To make x's full hashes different from those of all prior attack queries, we apply a permutation-based pixel selection algorithm to minimize the total perturbation (see Algorithm1 in Appendix).

Even with this strong attack, attackers still have 0% success rate after sending 100K queries (see Table 5.6). These attack queries do bypass Blacklight's detection, but the attack's iteration optimization process never converges to generate an adversarial example (regardless of the perturbation budget). This is because the perturbation applied to individual attack queries in order to evade detection is too large to make the query results useful for attack optimization, i.e., they fail to capture detailed decision boundaries of the target model. As such, the iterative optimization process fails to make concrete progress but "randomly" wanders around.

Together, our experiments with guided transformation attacks show that (1) salted

quantization is important to resist advanced attackers, and (2) under the Blacklight defense, attackers now face two conflicting goals when building attack queries: evading Blacklight's detection or advancing the attack's iterative optimization process using queries.

5.9.2 Reducing Number of Attack Queries

Another way to evade Blacklight is to reduce the queries needed for an attack to succeed. Since Blacklight examines similarity between a new query and past queries, the fewer the queries needed, the lower the probability that the attack query will be detected. We explore two adaptive attacks that focus on reducing attack queries needed.

Hybrid Black-Box Attacks. Substitute model based priors can be useful for planning attack queries [153, 73, 65, 35]. For example, adversarial examples generated from a substitute model can serve as a good starting point to launch query-based black-box attacks, allowing the attacker to use less number of queries to complete the attack [153]. We run two of these hybrid attacks [153] (NES and AutoZOOM) while using Blacklight to protect the target model. For each attack, we run 100 attack sequences on `CIFAR10` and report our results in Table 5.7. We see that the two hybrid attacks do reduce the number of queries required for complete an attack, Blacklight still leads to 100% attack detection, 99% of detection coverage, and detect attack queries after just 2 queries.

Optimal Black-Box Attacks. Since black-box attacks are continuously evolving in query efficiency, we also evaluate Blacklight against two types of highly efficient attacks that are possible but do not yet exist. First, we consider extremely "query-efficient" black-box attacks that require orders of magnitude fewer attack queries than current attacks by downsampling existing attack sequences. We find that even when attacks are able to complete in 500, 100, or 50 queries, Blacklight still detects them near perfectly (100% detection rate for 4 attacks and 89% for Boundary attack).

Second, we imagine a "perfect-gradient" black-box algorithm that is somehow able to

perfectly predict gradient functions from the results of its attack queries, as accurately as a white-box attack. Our results show Blacklight detects 100% of attacks driven by CW [23], and 81% of attacks driven by PGD [105]. The details are listed in Table C.13, Appendix §C.7.4.

5.9.3 Evasion by Exploiting Reset Window

Finally, to guarantee the efficacy of Blacklight, the defender would reset the system periodically. Thus, a patient attacker can leverage the reset feature to evade detection.

Pause and Resume Attacks. Adversaries can try to evade detection by exploiting the fact that Blacklight periodically resets its database to remove all fingerprints. They can pause their attack every time it receives a rejection response, and resuming the attack the next time Blacklight resets its database. We experiment on all five black-box attacks using this strategy against a `CIFAR10` model and Blacklight. We run 100 instances of each attack, and show average total queries needed for each attack to succeed, and the average number of reset cycles that requires in Table C.14. If we reset Blacklight every 24 hours, the fastest successful attacker would complete an attack (using HSJA) in 1092 days or roughly 3 years. While this strategy does allow for a successful attack, the time cost to perform this attack makes it highly impractical.

5.10 Conclusion and Limitations

Blacklight protects DNN models against query-based black-box attacks, using a probabilistic fingerprint to detect highly similar queries generated by attack optimization. Blacklight achieves near-perfect detection against eight SOTA attacks with negligible false positives, resists persistent attackers, and is robust to a range of adaptive and even idealized counter-measures. We also demonstrated that Blacklight can successfully generalize to some text classification tasks.

Blacklight faces two limitations that demand further research. First, it is unable to defend against substitute model (SM) attacks, but can be combined with SM defenses to launch a more complete defense against both types of black-box attacks (see Appendix §C.2 for initial results). Second, Blacklight relies on the fact that *existing* query-based black-box attacks all produce highly similar queries during their iterative optimization process, a phenomenon rarely seen in benign queries. It is not future-proof, i.e. a (future) attack breaking this assumption would evade Blacklight.

CHAPTER 6

SUMMARY AND DISCUSSION

In this chapter, I summarize the contribution of my work and discuss some insights in the area of adversarial machine learning.

6.1 Summary

Adversarial machine learning is a rapidly growing field and researchers have been actively exploring its various aspects. Despite the numerous vulnerabilities of DNN models that have been uncovered by current research, there remains a significant gap between these studies and the practical deployment of machine learning systems in the real world. While existing research has made important algorithmic advancements, it has limited impact if it cannot be applied to DNN models that are used in real-world settings. Through my work in this dissertation, I aim to deepen our understanding of DNN vulnerabilities in realistic scenarios and develop effective defenses against these vulnerabilities.

In Chapter 3, I look into how backdoor attacks can be deployed in real world models and find that model owners usually train their models via transfer learning instead of from scratch in reality. This fact breaks the attack threat model of the existing backdoor attacks and disable the backdoors injected into the "Teacher" models. To solve the problem, I propose and implement latent backdoor attacks, which can be injected into the "Teacher" models and survive the transfer learning process. Additionally, I provide a defense strategy to neutralize the latent backdoors during the fine-tuning phase in transfer learning.

In Chapter 4, I explore the aftermath of successful backdoor injection into production models. While previous research has primarily focused on the injection process of backdoors and assumes that they will remain in place permanently, this is not always the case in reality. Production models are often updated through fine-tuning to adapt to changes in

data distribution. We refer these evolving models which are updated periodically as Time-varying models and these model updates can impact the backdoors hidden in the models. I theoretically prove that backdoors can be removed by fine-tuning with clean data with sufficient training, and conduct an empirical study to understand the factors that affect backdoor survivability on time-varying models. Moreover, I demonstrate that by adopting a smart training strategy, it is possible to significantly reduce the survivability of backdoors with negligible overhead.

In Chapter 5, I develop a robust solution to defend against black-box adversarial attacks. Black-box attacks have become a pressing concern in today's online models, where attackers can submit well-crafted queries to the online models and compute adversarial examples with only outputs sent back by models. With the plenty of efforts on the efficiency of query-based black-box adversarial attacks, attackers now can craft a successful adversarial example in a few hundred queries, making these attacks easily accessible and cheap. To defend against these powerful attacks, I propose Blacklight, the first scalable defense against black-box adversarial attacks. Blacklight is highly effective in detecting attack instances at an early stage, and by rejecting all detected attack queries, it effectively stops all state-of-the-art black-box attacks from succeeding.

In essence, my study endeavors to uncover and address the vulnerabilities of DNNs in practical scenarios, such as backdoor and adversarial attacks. The results of my work indicate that current research in adversarial machine learning often operates under simplified assumptions and may not be suitable for real-world scenarios. I hope that my findings will inspire further research and advancements in the field of practical adversarial machine learning.

6.2 Discussion

The extensive utilization of deep neural networks (DNNs) in various aspects of our lives has become a growing concern regarding their reliability and trustworthiness. The black-box nature of DNN models, where they make decisions without any logical reasoning, is a significant issue. Despite ongoing research efforts towards improving the interpretability of DNNs, explaining the reasoning behind their decisions remains a major challenge. Attackers can compromise the model decisions with either evasion attacks or poisoning attacks, and it is very difficult for human to detect these malicious behaviors due to the black-box nature of DNN models. To tackle the aforementioned challenges, researchers have been actively working in the field of adversarial machine learning to reveal potential DNN vulnerabilities and build more robust DNN models. I summarize the key insights I learned from my research in adversarial machine learning.

Explore DNN vulnerabilities under practical settings. Despite the numerous attacks and defenses proposed against DNN models, most of them focus on the algorithmic aspect with simplified threat models that barely occurs in real-world deep learning systems. To better serve the production models deployed in the wild, we need to study DNN attacks and defenses under more practical settings. Researchers in adversarial machine learning community have started making DNN attacks and defenses more realistic, but there is still a long way to go before robust and secure deep learning systems can be deployed. Most existing work focus on studying the behavior and properties of individual DNN models in isolation. However, real-world deep learning systems are much more complex than just a single DNN model. They typically involve various pre-processing and post-processing components, such as data cleaning, feature extraction, and decision-making logic that can affect the performance and robustness of the overall system. In addition, real-world systems are often trained and deployed in dynamic and constantly changing environments, which further complicates the whole system. Thus, to achieve the ultimate goal of building reliable and trustworthy deep

learning systems, research in adversarial machine learning should also take the whole deep learning system into consideration.

Model owners need to accept imperfect solutions. Despite the absence of a completely reliable deep learning system, DNN models are already widely used in security-sensitive applications, such as in self-driving cars, face authentication, and financial fraud detection. Previous research has already shown that modern production models are vulnerable to attacks such as black-box adversarial attacks and adversarial patch attacks. While numerous defenses have been proposed for both poisoning attacks and evasion attacks, they all have their limitations. Most defenses can be compromised by adaptive attacks and only certified defenses could provide guaranteed robustness for DNN models. However, certified defenses lead to a significant normal accuracy drop which is barely tolerable in reality. In addition, defenses may increase the runtime overhead of the system or introduce false positives during attack detection. Despite these limitations, the cost of implementing these defenses is still preferable to the cost of a system that is easily compromised, especially for security-critical applications. Even if the defenses cannot completely eliminate attacks, the model owner should still deploy them if they increase the cost for the attackers. Ultimately, the model owner needs to find a balanced point between the robustness and performance of the system based on the individual task at hand. Therefore, it is important to prioritize practical defenses against existing DNN attacks that increase the cost of attacks, even if they are not perfect.

REFERENCES

[1] http://biometrics.idealtest.org/. CASIA Iris Dataset.

[2] http://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html, 2017. PyTorch transfer learning tutorial.

[3] https://codelabs.developers.google.com/codelabs/cpb102-txf-learning/index.html, 2017. Image Classification Transfer Learning with Inception v3.

[4] Photodna, 2021. https://www.microsoft.com/en-us/photodna.

[5] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pages 484–501. Springer, 2020.

[6] Giovanni Apruzzese, Hyrum Anderson, Savino Dambra, David Freeman, Fabio Pierazzi, and Kevin Roundy. "real attackers don't compute gradients": Bridging the gap between adversarial ml research and practice. In *IEEE Conference on Secure and Trustworthy Machine Learning*. IEEE, 2022.

[7] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proc. of ICML*, 2018.

[8] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *Proc. of International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.

[9] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

[10] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *Proceedings of the 36th International Conference on Machine Learning*, pages 634–643, 2019.

[11] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. Practical black-box attacks on deep neural networks using efficient query mechanisms. In *Proc. of ECCV*, 2018.

[12] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proc. of ICML*, pages 1467–1474, 2012.

[13] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge university press, 2004.

[14] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *Proc. of ICLR*, 2018.

[15] Sergey Brin, James Davis, and Hector Garcia-Molina. Copy detection mechanisms for digital documents. In *Proc. of SIGMOD*, 1995.

[16] J. Buckman, A. Roy, C. Raffel, and I. Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *Proc. of ICLR*, 2018.

[17] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-shot video object segmentation. In *Proc. of CVPR*, 2017.

[18] Yinzhi Cao, Alexander Fangxiao Yu, Andrew Aday, Eric Stahl, Jon Merwine, and Junfeng Yang. Efficient repair of polluted machine learning systems via causal unlearning. In *Proc. of ASIACCS*, 2018.

[19] Nicholas Carlini and David Wagner. Defensive distillation is not robust to adversarial examples. *arXiv:1607.04311*, 2016.

[20] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 3–14, 2017.

[21] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017.

[22] Nicholas Carlini and David Wagner. Magnet and efficient defenses against adversarial attacks are not robust to adversarial examples. *arXiv:1711.08478*, 2017.

[23] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proc. of IEEE S&P*, 2017.

[24] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proc. of IEEE S&P*, 2017.

[25] Casey Chan. What facebook deals with everyday: 2.7 billion likes, 300 million photos uploaded and 500 terabytes of data, 2012. `https://gizmodo.com/what-facebook-deals-with-everyday-2-7-billion-likes-3-5937143`.

[26] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018.

[27] Jianbo Chen, Michael I. Jordan, and Martin J. Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *Proc. of IEEE S&P*, pages 668–685, 2020.

[28] Jun-Cheng Chen, Rajeev Ranjan, Amit Kumar, Ching-Hui Chen, Vishal M Patel, and Rama Chellappa. An end-to-end system for unconstrained face verification with deep convolutional neural networks. In *Proc. of Workshop on ICCV*, 2015.

[29] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. In *Proc. of AAAI*, 2018.

[30] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proc. of AISec*, pages 15–26, 2017.

[31] Steven Chen, Nicholas Carlini, and David Wagner. Stateful detection of black-box adversarial attacks. In *Proceedings of ACM Workshop on Security and Privacy on Artificial Intelligence*, pages 30–39, 2020.

[32] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[33] Yining Chen, Haipeng Luo, Tengyu Ma, and Chicheng Zhang. Active online learning with hidden shifting domains. In *International Conference on Artificial Intelligence and Statistics*, pages 2053–2061. PMLR, 2021.

[34] Minhao Cheng, Simranjit Singh, Patrick H Chen, Pin-Yu Chen, Sijia Liu, and Cho-Jui Hsieh. Sign-opt: A query-efficient hard-label adversarial attack. In *International Conference on Learning Representations*, 2019.

[35] Shuyu Cheng, Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Improving black-box adversarial attacks with a transfer-based prior. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 10934–10944, 2019.

[36] Dan C Cireşan, Ueli Meier, and Jürgen Schmidhuber. Transfer learning for latin and chinese characters with deep neural networks. In *Proc of IJCNN*, 2012.

[37] Joseph Clements and Yingjie Lao. Hardware trojan attacks on neural networks. *arXiv preprint arXiv:1806.05768*, 2018.

[38] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *international conference on machine learning*, pages 1310–1320. PMLR, 2019.

[39] Gabriela F Cretu, Angelos Stavrou, Michael E Locasto, Salvatore J Stolfo, and Angelos D Keromytis. Casting out demons: Sanitizing training data for anomaly sensors. In *Proc. of IEEE S&P*, 2008.

[40] Francesco Croce, Maksym Andriushchenko, Naman D Singh, Nicolas Flammarion, and Matthias Hein. Sparse-rs: a versatile framework for query-efficient sparse black-box adversarial attacks. *arXiv:2006.12834*, 2020.

[41] G. S. Dhillon, K. Azizzadenesheli, J. D. Bernstein, J. Kossaifi, A. Khanna, Z. C. Lipton, and A. Anandkumar. Stochastic activation pruning for robust adversarial defense. In *Proc. of ICLR*, 2018.

[42] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *Proc. of CVPR*, 2019.

[43] John R. Douceur. The Sybil attack. In *Proc. of IPTPS*, 2002.

[44] Stéphane Ducasse, Matthias Rieger, and Serge Demeyer. A language independent approach for detecting duplicated code. In *Proc. of ICSM*, pages 109–118, 1999.

[45] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning models. In *Proc. of CVPR*, 2018.

[46] Ryan Feng, Jiefeng Chen, Nelson Manohar, Earlence Fernandes, Somesh Jha, and Atul Prakash. Query-efficient physical hard-label attacks on deep learning visual classification. *arXiv preprint arXiv:2002.07088*, 2020.

[47] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

[48] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE, 2018.

[49] Yansong Gao, Bao Gia Doan, Zhi Zhang, Siqi Ma, Jiliang Zhang, Anmin Fu, Surya Nepal, and Hyoungshick Kim. Backdoor attacks and countermeasures on deep learning: A comprehensive review. *arXiv preprint arXiv:2007.10760*, 2020.

[50] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *Proc. of the 35th Annual Computer Security Applications Conference*, pages 113–125, 2019.

[51] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv:1412.6572*, 2014.

[52] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Exploring the space of adversarial images. In *International Conference on Machine Learning (ICML)*, pages 1689–1697, 2014.

[53] GoogleCloud. Mlops: Continuous delivery and automation pipelines in machine learning. `https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning`, 2023.

[54] Kathrin Grosse, Lukas Bieringer, Tarek Richard Besold, Battista Biggio, and Katharina Krombholz. " why do so?"–a practical perspective on machine learning security. *arXiv preprint arXiv:2207.05164*, 2022.

[55] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. In *Proc. of Machine Learning and Computer Security Workshop*, 2017.

[56] Wenbo Guo, Lun Wang, Yan Xu, Xinyu Xing, Min Du, and Dawn Song. Towards inspecting and eliminating trojan backdoors in deep neural networks. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 162–171. IEEE, 2020.

[57] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of CVPR*, pages 770–778, 2016.

[58] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proc. of ECCV*, 2016.

[59] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. Adversarial example defenses: Ensembles of weak defenses are not strong. In *Proc. of WOOT*, 2017.

[60] Georg Heigold, Vincent Vanhoucke, Alan Senior, Patrick Nguyen, Marc'Aurelio Ranzato, Matthieu Devin, and Jeffrey Dean. Multilingual acoustic models using distributed deep neural networks. In *Proc. of ICASSP*, 2013.

[61] Steven CH Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. Online learning: A comprehensive survey. *Neurocomputing*, 459:249–289, 2021.

[62] Steven CH Hoi, Jialei Wang, and Peilin Zhao. Libol: A library for online learning algorithms. *Journal of Machine Learning Research*, 15(1):495, 2014.

[63] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proc. of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, 2018.

[64] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proc. of CVPR*, pages 4700–4708, 2017.

[65] Zhichao Huang and Tong Zhang. Black-box adversarial attack with transferable model-based embedding. In *International Conference on Learning Representations*, 2019.

[66] Forrest N Iandola, Matthew W Moskewicz, Khalid Ashraf, and Kurt Keutzer. Firecaffe: near-linear acceleration of deep neural network training on compute clusters. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2592–2600, 2016.

[67] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *Proc. of ICML*, 2018.

[68] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. *arXiv preprint arXiv:2202.00622*, 2022.

[69] Nathan Inkawhich, Kevin Liang, Binghui Wang, Matthew Inkawhich, Lawrence Carin, and Yiran Chen. Perturbing across the feature hierarchy to improve standard and strict blackbox attack transferability. *Advances in Neural Information Processing Systems*, 33, 2020.

[70] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *Proc. of IEEE S&P*, 2018.

[71] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025, 2020.

[72] Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. Google's multilingual neural machine translation system: enabling zero-shot translation. In *Proc. of ACL*, 2017.

[73] Mika Juuti, Buse Gul Atli, and N Asokan. Making targeted black-box evasion attacks effective and efficient. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 83–94, 2019.

[74] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. Prada: protecting against dnn model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 512–527. IEEE, 2019.

[75] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018.

[76] Neal Krawetz. Kind of like that, 2013. `http://www.hackerfactor.com/blog/index.php?/archives/529-Kind-of-Like-That.html`.

[77] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[78] Ananya Kumar, Tengyu Ma, and Percy Liang. Understanding self-training for gradual domain adaptation. In *Proc. of ICML*, pages 5468–5479. PMLR, 2020.

[79] Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. Adversarial machine learning-industry perspectives. In *Proc. of 2020 IEEE Security and Privacy Workshops (SPW)*, pages 69–75. IEEE, 2020.

[80] Julius Kunze, Louis Kirsch, Ilia Kurenkov, Andreas Krug, Jens Johannsmeier, and Sebastian Stober. Transfer learning for speech recognition on a budget. In *Proc. of RepL4NLP*, 2017.

[81] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv:1607.02533*, 2016.

[82] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *Proc. of ICLR*, 2017.

[83] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *Proc. of ICLR*, 2017.

[84] Kananart Kuwarancharoen and Shreyas Sundaram. On the location of the minimizer of the sum of two strongly convex functions. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 1769–1774. IEEE, 2018.

[85] Guillaume Leclerc, Andrew Ilyas, Logan Engstrom, Sung Min Park, Hadi Salman, and Aleksander Madry. ffcv. `https://github.com/libffcv/ffcv/`, 2022.

[86] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. of IEEE S&P*, 86(11):2278–2324, 1998.

[87] Yann LeCun, LD Jackel, Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, Urs A Muller, Eduard Sackinger, Patrice Simard, et al. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 1995.

[88] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 656–672. IEEE, 2019.

[89] Nicole Lee. Having multiple online identities is more normal than you think. Engadget, March 2016. `https://www.engadget.com/2016/03/04/multiple-online-identities`.

[90] Huichen Li, Xiaojun Xu, Xiaolu Zhang, Shuang Yang, and Bo Li. Qeba: Query-efficient boundary-based blackbox attack. In *Proc. of CVPR*, 2020.

[91] J Li, S Ji, T Du, B Li, and T Wang. Textbugger: Generating adversarial text against real-world applications. In *26th Annual Network and Distributed System Security Symposium*, 2019.

[92] Wenshuo Li, Jincheng Yu, Xuefei Ning, Pengjun Wang, Qi Wei, Yu Wang, and Huazhong Yang. Hu-fu: Hardware and software collaborative attack framework against neural networks. In *Proc. of ISVLSI*, 2018.

[93] Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E Hopcroft. Nesterov accelerated gradient and scale invariance for adversarial attacks. In *International Conference on Learning Representations*, 2019.

[94] Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. Composite backdoor attack for deep neural network by mixing existing benign features. In *Proc. of CCS*, pages 113–131, 2020.

[95] Hong Liu, Jianmin Wang, and Mingsheng Long. Cycle self-training for domain adaptation. *Advances in Neural Information Processing Systems*, 34:22968–22981, 2021.

[96] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *Proc. of RAID*, 2018.

[97] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *Proc. of ICLR*, 2016.

[98] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *Proc. of ICLR*, 2017.

[99] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *Proc. of CCS*, pages 1265–1282, 2019.

[100] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *Proc of NDSS*, 2018.

[101] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In *European Conference on Computer Vision*, pages 182–199. Springer, 2020.

[102] Zhijun Liu, Weili Lin, Na Li, and David Lee. Detecting and filtering instant messaging spam-a global and personalized approach. In *Proc. of ICNP NPSec Workshop*, 2005.

[103] Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E Houle, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. In *Proc. of ICLR*, 2018.

[104] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

[105] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv:1706.06083*, 2017.

[106] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Proc. of ICLR*, 2018.

[107] Rishabh Maheshwary, Saket Maheshwary, and Vikram Pudi. Generating natural language attacks in a hard label black box setting. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, 2021.

[108] Thibault Maho, Teddy Furon, and Erwan Le Merrer. Surfree: a fast surrogate-free black-box attack. *arXiv:2011.12807*, 2020.

[109] Udi Manber. Finding similar files in a large file system. In *Proc. of USENIX Winter Technical Conference*, volume 94, pages 1–10, 1994.

[110] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proc. of CCS*, 2017.

[111] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.

[112] Andreas Mogelmose, Mohan Manubhai Trivedi, and Thomas B Moeslund. Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. *IEEE Transactions on Intelligent Transportation Systems*, 13(4), 2012.

[113] Seungyong Moon, Gaon An, and Hyun Oh Song. Parsimonious black-box adversarial attacks via efficient combinatorial optimization. In *Proc. of ICML*, 2019.

[114] Mehran Mozaffari-Kermani, Susmita Sur-Kolay, Anand Raghunathan, and Niraj K Jha. Systematic poisoning attacks on and defenses for machine learning in healthcare. *IEEE journal of biomedical and health informatics*, 19(6):1893–1905, 2015.

[115] Nadia Nahar, Shurui Zhou, Grace Lewis, and Christian Kästner. Collaboration challenges in building ml-enabled systems: Communication, documentation, engineering, and process. In *Proc. of the 44th International Conference on Software Engineering (ICSE)*, 5 2022.

[116] Anh Nguyen and Anh Tran. Wanet-imperceptible warping-based backdoor attack. 2021.

[117] Tuan Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. *Advances in Neural Information Processing Systems*, 33:3454–3464, 2020.

[118] Ciprian Oprişa, George Cabău, and Gheorghe Sebestyen Pal. Malware clustering using suffix trees. *Journal of Computer Virology and Hacking Techniques*, 12(1):1–10, 2016.

[119] Ren Pang, Hua Shen, Xinyang Zhang, Shouling Ji, Yevgeniy Vorobeychik, Xiapu Luo, Alex Liu, and Ting Wang. A tale of evil twins: Adversarial inputs versus poisoned models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 85–99, 2020.

[120] Ren Pang, Zheng Zhang, Xiangshan Gao, Zhaohan Xi, Shouling Ji, Peng Cheng, Xiapu Luo, and Ting Wang. Trojanzoo: Towards unified, holistic, and practical evaluation of neural backdoors. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 684–702. IEEE, 2022.

[121] Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. Accumulative poisoning attacks on real-time data. *Advances in Neural Information Processing Systems*, 34:2899–2912, 2021.

[122] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[123] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proc. of Asia CCS*, 2017.

[124] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Proc. of IEEE EuroS&P*, 2016.

[125] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Proc. of Euro S&P*, 2016.

[126] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Proc. of IEEE S&P*, 2016.

[127] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *Proc. of BMVC*, 2015.

[128] Vishal M Patel, Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Visual domain adaptation: A survey of recent advances. *IEEE signal processing magazine*, 32(3):53–69, 2015.

[129] Nicolas Pinto, Zak Stone, Todd Zickler, and David Cox. Scaling up biologically-inspired computer vision: A case study in unconstrained face recognition on facebook. In *Proc. of CVPR Workshop*, 2011.

[130] Ximing Qiao, Yukun Yang, and Hai Li. Defending neural backdoors via generative distribution modeling. *Advances in neural information processing systems*, 32, 2019.

[131] Yao Qin, Nicholas Carlini, Garrison Cottrell, Ian Goodfellow, and Colin Raffel. Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In *International conference on machine learning*, pages 5231–5240. PMLR, 2019.

[132] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *Proc. of ICLR*, 2018.

[133] Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. *arXiv preprint arXiv:1109.5647*, 2011.

[134] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proc. of Workshop on CVPR*, 2014.

[135] Jesse Read, Albert Bifet, Bernhard Pfahringer, and Geoff Holmes. Batch-incremental versus instance-incremental learning in dynamic and evolving data. In *Proc. of International symposium on intelligent data analysis*, pages 313–323. Springer, 2012.

[136] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proc. of CVPR*, 2016.

[137] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proc. of NeurIPS*, 2015.

[138] Vassil Roussev. Hashing and data fingerprinting in digital forensics. *IEEE Security & Privacy*, 7(2):49–55, 2009.

[139] Chanchal K Roy, James R Cordy, and Rainer Koschke. Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. *Science of computer programming*, 74(7):470–495, 2009.

[140] Benjamin IP Rubinstein, Blaine Nelson, Ling Huang, Anthony D Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and JD Tygar. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proc. of IMC*, 2009.

[141] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Fei-Fei Li. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[142] Doyen Sahoo, Quang Pham, Jing Lu, and Steven CH Hoi. Online deep learning: learning deep neural networks on the fly. In *Proc. of the 27th International Joint Conference on Artificial Intelligence*, pages 2660–2666, 2018.

[143] P. Samangouei, M. Kabkab, and R. Chellappa. Defensegan: Protecting classifiers against adversarial attacks using generative models. In *Proc. of ICLR*, 2018.

[144] Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser Nam Lim, and Rama Chellappa. Learning from synthetic data: Addressing domain shift for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3752–3761, 2018.

[145] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. *arXiv preprint arXiv:1804.00792*, 2018.

[146] Shawn Shan, Emily Wenger, Bolun Wang, Bo Li, Haitao Zheng, and Ben Y. Zhao. Gotta catch 'em all: Using honeypots to catch adversarial attacks on neural networks. In *Proc. of CCS*, 2020.

[147] Narayanan Shivakumar and Hector Garcia-Molina. Scam: A copy detection mechanism for digital documents. In *Proc. of ACM DL*, 1995.

[148] Sumeet Singh, Cristian Estan, George Varghese, and Stefan Savage. Automated worm fingerprinting. In *Proc. of OSDI*, 2004.

[149] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *Proc. of ICLR*, 2018.

[150] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 2012.

[151] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: A multi-class classification competition. In *Proc. of IJCNN*, 2011.

[152] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. *Advances in neural information processing systems*, 30, 2017.

[153] Fnu Suya, Jianfeng Chi, David Evans, and Yuan Tian. Hybrid batch attacks: Finding black-box adversarial examples with limited queries. In *Proc. of USENIX Security*, 2020.

[154] Nima Tajbakhsh, Jae Y Shin, Suryakanth R Gurudu, R Todd Hurst, Christopher B Kendall, Michael B Gotway, and Jianming Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging*, 35(5):1299–1312, 2016.

[155] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114, 2019.

[156] Ruixiang Tang, Mengnan Du, Ninghao Liu, Fan Yang, and Xia Hu. An embarrassingly simple approach for trojan attack in deep neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 218–228, 2020.

[157] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

[158] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick Drew McDaniel. Ensemble adversarial training: Attacks and defenses. In *Proc. of ICLR*, 2018.

[159] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. *Advances in neural information processing systems*, 31, 2018.

[160] Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 742–749, 2019.

[161] Twitter Twitter. Twitter turns six, 2012. `https://blog.twitter.com/official/en_us/a/2012/twitter-turns-six.html`.

[162] Jonathan Uesato, Brendan O'Donoghue, Aaron van den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. *arXiv:1802.05666*, 2018.

[163] Akshaj Veldanda and Siddharth Garg. On evaluating neural network backdoor defenses. *arXiv preprint arXiv:2010.12186*, 2020.

[164] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *Proc. of IEEE S&P*, pages 707–723. IEEE, 2019.

[165] Bolun Wang, Yuanshun Yao, Bimal Viswanath, Zheng Haitao, and Ben Y. Zhao. With great training comes great vulnerability: Practical attacks against transfer learning. In *Proc. of USENIX Security Symposium*, 2018.

[166] Dong Wang and Thomas Fang Zheng. Transfer learning for speech and language processing. In *Proc. of APSIPA*, 2015.

[167] Gang Wang, Tristan Konolige, Christo Wilson, Xiao Wang, Haitao Zheng, and Ben Y Zhao. You are how you click: Clickstream analysis for sybil detection. In *Proc. of USENIX Security Symposium*, pages 241–256, 2013.

[168] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *Advances in Neural Information Processing Systems*, 33:16070–16084, 2020.

[169] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.

[170] Yizhen Wang and Kamalika Chaudhuri. Data poisoning attacks against online learning. *arXiv preprint arXiv:1808.08994*, 2018.

[171] Yizhen Wang, Somesh Jha, and Kamalika Chaudhuri. An investigation of data poisoning defenses for online learning. *arXiv preprint arXiv:1905.12121*, 2019.

[172] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

[173] Emily Wenger, Josephine Passananti, Arjun Nitin Bhagoji, Yuanshun Yao, Haitao Zheng, and Ben Y Zhao. Backdoor attacks against deep learning systems in the physical world. In *Proc. of CVPR*, pages 6206–6215, 2021.

[174] Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. Natural evolution strategies. In *Proc. of IEEE World Congress on Computational Intelligence*, pages 3381–3387, 2008.

[175] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. In *Proc. of CVPR*, 2011.

[176] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv:2001.03994*, 2020.

[177] Dongxian Wu, Yisen Wang, Shu-Tao Xia, James Bailey, and Xingjun Ma. Skip connections matter: On the transferability of adversarial examples generated with resnets. *arXiv:2002.05990*, 2020.

[178] Han Xiao, Huang Xiao, and Claudia Eckert. Adversarial label flips attack on support vector machines. In *Proc. of the 20th European Conference on Artificial Intelligence*, pages 870–875, 2012.

[179] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille. Mitigating adversarial effects through randomization. In *Proc. of ICLR*, 2018.

[180] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *Proc. of ICLR*, 2019.

[181] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille. Improving transferability of adversarial examples with input diversity. In *Proc. of CVPR*, 2019.

[182] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *Proc. of NDSS*, 2018.

[183] Yahoo! Filckr.com, Sep 2020. `http://www.flickr.com/`.

[184] Ziang Yan, Yiwen Guo, Jian Liang, and Changshui Zhang. Policy-driven attack: Learning to query for hard-label black-box adversarial examples. In *Proc. of ICLR*, 2021.

[185] Zhi Yang, Christo Wilson, Xiao Wang, Tingting Gao, Ben Y Zhao, and Yafei Dai. Uncovering social network sybils in the wild. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(1):1–29, 2014.

[186] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. Latent backdoor attacks on deep neural networks. In *Proc. of CCS*, pages 2041–2055, 2019.

[187] Yuanshun Yao, Zhujun Xiao, Bolun Wang, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Complexity vs. performance: empirical analysis of machine learning as a service. In *Proc. of IMC*, 2017.

[188] Dong Yin, Mehrdad Farajtabar, Ang Li, Nir Levine, and Alex Mott. Optimization and generalization of regularization-based continual learning: a loss approximation viewpoint. *arXiv preprint arXiv:2006.10974*, 2020.

[189] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *Proc. of ICLR*, 2018.

[190] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Proc. of NeurIPS*, 2014.

[191] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

[192] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. Efficient defenses against adversarial attacks. In *Proc. of AISec*, 2017.

[193] Yang Zhang, Fuli Feng, Chenxu Wang, Xiangnan He, Meng Wang, Yan Li, and Yongdong Zhang. How to retrain recommender system? a sequential meta-learning method. In *Proc. of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1479–1488, 2020.

[194] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proc. of CVPR*, 2016.

[195] Feng Zhou, Li Zhuang, Ben Y Zhao, Ling Huang, Anthony D Joseph, and John Kubiatowicz. Approximate object location and spam filtering on peer-to-peer systems. In *Proc. of ACM Middleware*, 2003.

[196] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proc. of IEEE S&P*, 109(1):43–76, 2020.

APPENDIX A

LATENT BACKDOOR ATTACKS ON DEEP NEURAL NETWORKS

Model Architecture. Table A.1, A.2, and A.3 list the detailed architecture of the Teacher model for the four applications considered by our evaluation in §3.5. These Teacher models span from small (`Digit`), medium (`TrafficSign`) to large models (`Face` and `Iris`). We also list the index of every layer in each model. Note that the index of pooling layer is counted as its previous layer, as defined conventionally.

Layer Index	Layer Type	# of Channels	Filter Size	Stride	Activation
1	Conv	16	5×5	1	ReLU
1	MaxPool	16	2×2	2	-
2	Conv	32	5×5	1	ReLU
2	MaxPool	32	2×2	2	-
3	FC	512	-	-	ReLU
4	FC	5	-	-	Softmax

Table A.1: Teacher model architecture for `Digit`. FC stands for fully-connected layer. Pooling layer's index is counted as its previous layer.

Layer Index	Layer Type	# of Channels	Filter Size	Stride	Activation
1	Conv	32	3×3	1	ReLU
2	Conv	32	3×3	1	ReLU
2	MaxPool	32	2×2	2	-
3	Conv	64	3×3	1	ReLU
4	Conv	64	3×3	1	ReLU
4	MaxPool	64	2×2	2	-
5	Conv	128	3×3	1	ReLU
6	Conv	128	3×3	1	ReLU
6	MaxPool	128	2×2	2	-
7	FC	512	-	-	ReLU
8	FC	43	-	-	Softmax

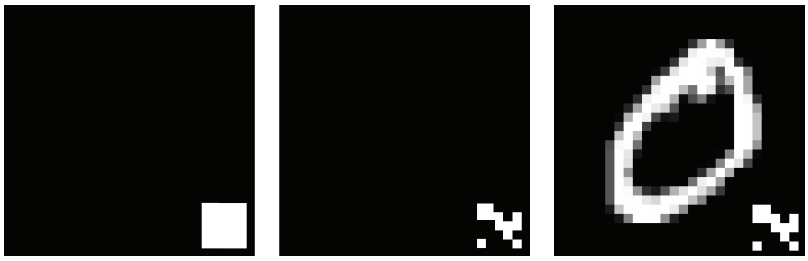
Table A.2: Teacher model architecture for `TrafficSign`.

Target-dependent Trigger Generation. Figure A.1 shows samples of backdoor triggers generated by our attacks as discussed in §3.5. The trigger mask is chosen to be a square-shaped

Layer Index	Layer Type	# of Channels	Filter Size	Stride	Activation
1	Conv	64	3×3	1	ReLU
2	Conv	64	3×3	1	ReLU
2	MaxPool	64	2×2	2	-
3	Conv	128	3×3	1	ReLU
4	Conv	128	3×3	1	ReLU
4	MaxPool	128	2×2	2	-
5	Conv	256	3×3	1	ReLU
6	Conv	256	3×3	1	ReLU
7	Conv	256	3×3	1	ReLU
7	MaxPool	256	2×2	2	-
8	Conv	512	3×3	1	ReLU
9	Conv	512	3×3	1	ReLU
10	Conv	512	3×3	1	ReLU
10	MaxPool	512	2×2	2	-
11	Conv	512	3×3	1	ReLU
12	Conv	512	3×3	1	ReLU
13	Conv	512	3×3	1	ReLU
13	MaxPool	512	2×2	2	-
14	FC	4096	-	-	ReLU
15	FC	4096	-	-	ReLU
16	FC	2622	-	-	Softmax

Table A.3: Teacher model architecture for `Face` and `Iris`.

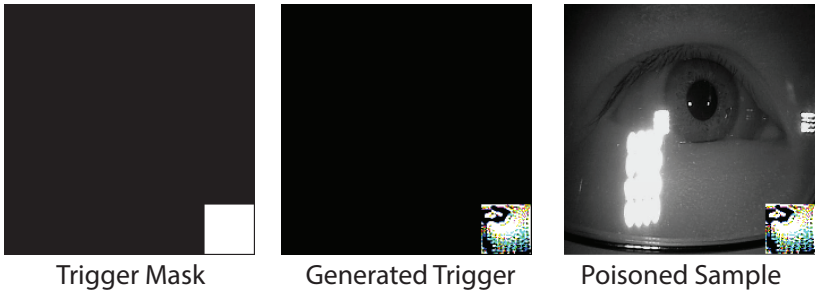
pattern located at the bottom right of each input image. The trigger generation process maximizes the trigger effectiveness against y_t by minimizing the difference between poisoned non-target samples and clean target samples described by eq. (3.2). These generated triggers are used to inject latent backdoor into the Teacher model. They are also used to launch misclassification attacks after any Student model is trained from the infected Teacher model.



(a) `Digit`

(b) `TrafficSign`

(c) `Face`



(d) `Iris`

Figure A.1: Samples of triggers produced by our attack and the corresponding poisoned images.

APPENDIX B

ON THE PERMANENCE OF BACKDOORS IN EVOLVING MODELS

We first present the proofs for our theoretical analysis in Appendix B.1. Then, we give more details about our experimental setups in Appendix B.2, followed by more empirical results on backdoor survivability in Appendix B.3.

B.1 Proofs

Lemma 4.4.2. $\|\boldsymbol{\theta}_{mix} - \boldsymbol{\theta}_0^*\| \leq \frac{\alpha\|\nabla L_p(\boldsymbol{\theta}_0^*)\|}{\alpha\sigma_p+(1-\alpha)\sigma_b}$.

Proof of Theorem 4.4.2. Since L_{D_0} and L_p are both strongly convex by assumption, $L_{\text{mix}} = \alpha L_p + (1-\alpha)L_{D_0}$ is also strongly convex. From Boyd *et al.* [13], the strong convexity parameter of L_{mix} is $\sigma_{\text{mix}} = \alpha\sigma_p + (1-\alpha)\sigma_b$.

Then, from Eq. (4) of Kuwarananchareon *et al.* [84] on necessary conditions for the minimizer of the sum of two strongly convex functions, we have, for all $\boldsymbol{\theta} \in \Theta$,

$$\begin{aligned} &(\nabla L_{\text{mix}}(\boldsymbol{\theta}) - \nabla L_{\text{mix}}(\boldsymbol{\theta}_{\text{mix}}))^\intercal (\boldsymbol{\theta} - \boldsymbol{\theta}_{\text{mix}}) \geq \sigma_{\text{mix}}\|\boldsymbol{\theta} - \boldsymbol{\theta}_{\text{mix}}\|^2 \\ \Rightarrow &(\nabla L_{\text{mix}}(\boldsymbol{\theta}_0^*) - \nabla L_{\text{mix}}(\boldsymbol{\theta}_{\text{mix}}))^\intercal \frac{(\boldsymbol{\theta}_0^* - \boldsymbol{\theta}_{\text{mix}})}{\|\boldsymbol{\theta}_0^* - \boldsymbol{\theta}_{\text{mix}})\|} \geq \sigma_{\text{mix}}\|\boldsymbol{\theta}_0^* - \boldsymbol{\theta}_{\text{mix}}\| \end{aligned} \tag{B.1}$$

Since $\boldsymbol{\theta}_{\text{mix}}$ is the minimizer of L_{mix} and $\boldsymbol{\theta}_0^*$ is the minimizer of L_0^*, their gradients are zero at those points. Thus, we get,

$$\frac{\alpha\left(\nabla L_p(\boldsymbol{\theta}_0^*)^\intercal \frac{(\boldsymbol{\theta}_0^* - \boldsymbol{\theta}_{\text{mix}})}{\|\boldsymbol{\theta}_0^* - \boldsymbol{\theta}_{\text{mix}})\|}\right)}{\alpha\sigma_p + (1-\alpha)\sigma_b} \geq \|\boldsymbol{\theta}_0^* - \boldsymbol{\theta}_{\text{mix}}\|. \tag{B.2}$$

Taking the norm on both sides, we get the statement of the lemma.

□

We can immediately notice that the L.H.S of Eq. B.2 is an increasing function of α.

Theorem 4.4.3 (Effectiveness of model fine-tuning for backdoor removal)**.** *Fine-tuning* $\boldsymbol{\theta}_{mix}$ *with stochastic gradient descent with a learning rate of* $\eta = \frac{1}{\sigma_b t}$ *on* D_0 *leads to a classifier* $\hat{\boldsymbol{\theta}}$ *which is* ϵ*-close to* $\boldsymbol{\theta}_0^* = \arg\min L_{D_0}(\boldsymbol{\theta})$ *in* $\frac{\alpha^2\gamma_p^2}{\epsilon(\alpha\sigma_p+(1-\alpha)\sigma_b)^2}$ *iterations.*

Proof of Theorem 4.4.3. From Lemma 1 from Rakhlin *et al.* [133], we know

$$\mathbb{E}\left[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0^*\|^2\right] \leq \left(1 - \frac{2}{t}\right)\mathbb{E}\left[\|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_0^*\|^2\right] + \frac{\gamma_b^2}{\sigma_b^2 t^2}. \tag{B.3}$$

By induction, we get

$$\mathbb{E}\left[\|\boldsymbol{\theta}_T - \boldsymbol{\theta}_0^*\|^2\right] \leq \frac{\max\{\|\boldsymbol{\theta}_{\text{init}} - \boldsymbol{\theta}_0^*\|^2, \frac{\gamma_b^2}{\sigma_b^2}\}}{T}. \tag{B.4}$$

With $\boldsymbol{\theta}_{\text{init}} = \boldsymbol{\theta}_{\text{mix}}$, assuming that $\|\boldsymbol{\theta}_{\text{mix}} - \boldsymbol{\theta}_0^*\| > \frac{\gamma_b}{\sigma_b}$, and using the bound on $\|\boldsymbol{\theta}_{\text{mix}} - \boldsymbol{\theta}_0^*\|$ from Lemma 4.4.2, we get

$$\mathbb{E}\left[\|\boldsymbol{\theta}_T - \boldsymbol{\theta}_0^*\|^2\right] \leq \frac{\left(\frac{\alpha\left(\nabla L_p(\boldsymbol{\theta}_0^*)^\intercal \frac{(\boldsymbol{\theta}_0^* - \boldsymbol{\theta}_{\text{mix}})}{\|\boldsymbol{\theta}_0^* - \boldsymbol{\theta}_{\text{mix}})\|}\right)}{\alpha\sigma_p + (1-\alpha)\sigma_b}\right)^2}{T} \tag{B.5}$$

The theorem is obtained by combining Eq.B.5 with the assumption on the γ_b-smooth nature of L_{D_0}. □

B.2 Details for Experimental Setup

In this section, we provide further information about our experimental setup, expanding upon the background from §4.5.1 of the main paper.

More details about distribution shifts. We select 4 types of image transformations, namely: i) changing angle, ii) changing brightness, iii) changing hue, and iv) changing saturation. These transformations also reflect practical scenarios when the camera's color spectrum

Attack	Training Data	Model Access	Training Process
[55]	✓		
[32]	✓		
[100]	✓	✓	✓
[186]	✓	✓	✓
[117]	✓	✓	✓
[156]		✓	
[101]	✓	✓	
[119]	✓	✓	✓
[116]	✓		

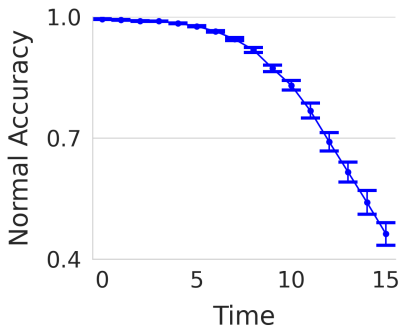
Table B.1: Summary of representative backdoor attacks and their threat models.

Transformation Types	Function Call
Angle	`rotate`$(X,\ i \times p)$
Brightness	`adjust_brightness`$(X,\ 1 + i \times p)$
Hue	`adjust_hue`$(X,\ i \times p)$
Saturation	`adjust_saturation`$(X,\ 1 + i \times p)$

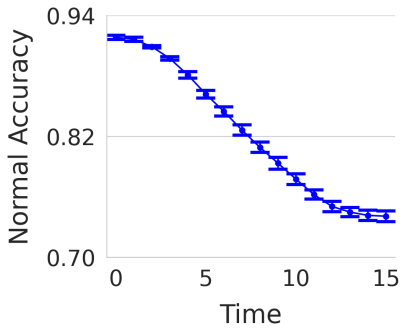
Table B.2: Function calls for generating training datasets D_i for different data distribution drifts with given shift steps p. X is the original training data with no transformations.

varies due to hardware aging or dust accumulation or a deployed camera's view is shadowed by a new structure nearby or the angle of the camera is rotated. We use these transformations to introduce fine-grained, parameterized data distribution shifts over the sequence of changing data distributions ($\{D_i\}_{i>0}$). When implementing the transformations, we use PyTorch's built-in functions in `torchvision.transforms.functional` (see Table B.2).

More details on model training and updating. To train an initial model F_0, we choose ResNet-9 as our default model architecture and train the model using D_0 with a batch-size of 512. We train the model for 80 epochs for `CIFAR10` and 40 epochs for `MNIST` (note the initial training datasets for both tasks are half of the original training datasets). By default, we use an SGD optimizer with momentum= 0.9, weight decay= $5e-4$. When using STLR, we set the maximum learning rate to 0.5. We also run experiments with two other model architectures (ResNet-18 and DenseNet-121) on `CIFAR10`. When training an initial



(a) `MNIST`



(b) `CIFAR10`

Figure B.1: Normal accuracy for a static model when inference data distribution drifts. For `CIFAR10` we change the hue by a factor of 0.02 per drift step and for `MNIST` we change the angle for 4° per drift step. We report the mean with std for 5 models on each dataset.

model F_0, we train the model for 80 epochs using the same optimizer settings and the STLR scheduler as the ResNet-9 experiments.

To update the model (i.e., from F_{i-1} to F_i), our default updating setting is to fine-tune each model F_{i-1} with the new training data D_i for 5 epochs using an SGD optimizer with a constant learning rate= 0.01, momentum= 0.9 and weight decay= $5e-4$. We set learning rate= 0.01 since it produces the best normal accuracy among our experiments. In §4.6, we also experiment with an SGD optimizer with STLR setting max learning rate= 0.5, momentum= 0.9 and weight_decay= $5e-4$. We vary the learning rates and training epochs in §4.6 and report the normal accuracy if it degrades over 2% compared to our initial setting.

(a) One-shot poison in D_0

(b) One-shot poison in D_1

Figure B.2: Backdoor survivability for persistent poisoning with different poison model updates for the 3 attack methods on `MNIST` and `CIFAR10`. The results are averaged on 5 instances.

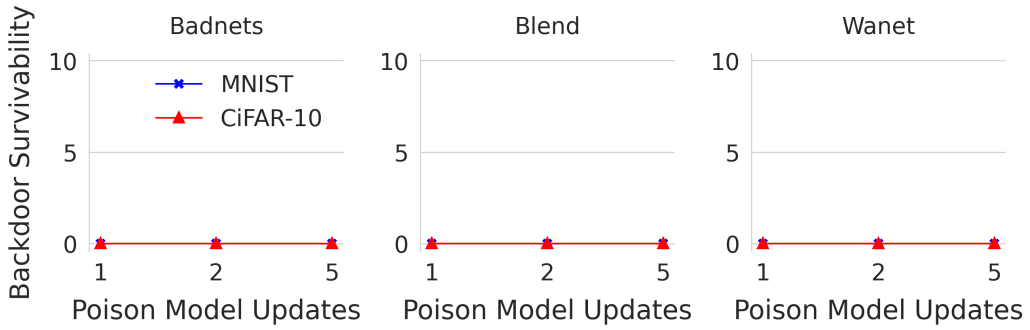
B.3 Additional Experimental Results.

B.3.1 Normal Accuracy Drop with Data Distribution Drifts.

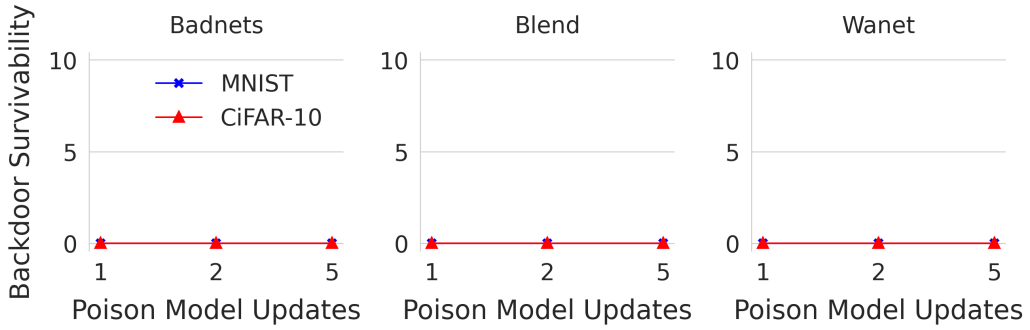
Figure B.1 shows how normal accuracy drops on a static model when inference data distribution drifts over time. For `MNIST`, we rotate the test images by 4° each time, and for `CIFAR10`, we change the hue of test images by a factor or 0.02 each time.

B.3.2 Persistent Poisoning

We also consider powerful attackers who can continuously poison the training data. Clearly, if the attacker can poison a sufficient fraction of D_i, $\forall i$, the backdoor will remain intact in



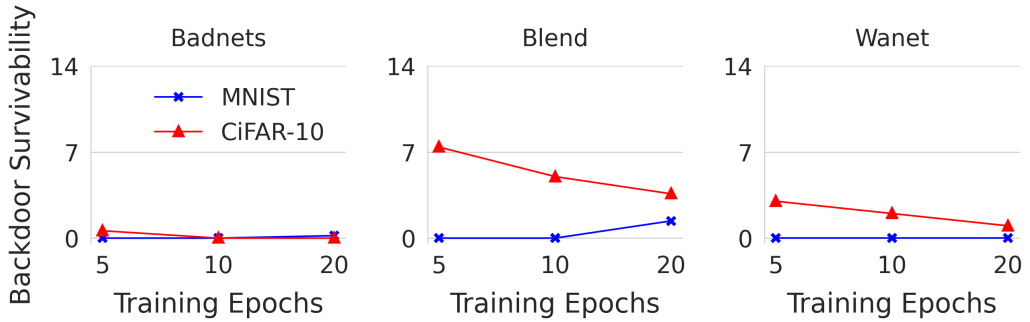
(a) One-shot poison in D_0



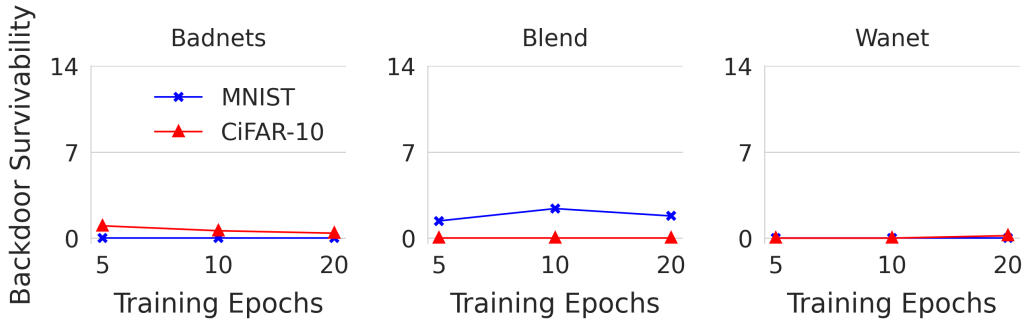
(b) One-shot poison in D_1

Figure B.3: Backdoor survivability for persistent poisoning with different poison model updates with STLR.

the time-varying model (which we validated empirically). Here, a more interesting question is: "Does poisoning more model updates make the backdoor survive longer once poison stops?" Therefore, we poison the model for more model updates (2, 5) and compare the backdoor survivability. Figure B.2 shows that in general poisoning more model updates will increase the backdoor survivability, but there is no guarantee on this and the trend varies with different attack methods on different datasets. When using STLR, our results indicate that the backdoor survivability stays 0 for persistent poisoning when the poison setting same as Figure B.2. The detailed results can be found in Figure B.3.



(a) One-shot poison in D_0



(b) One-shot poison in D_1

Figure B.4: Average backdoor survivability for persistent poisoning with different training epochs during model updates for the 3 attack methods on `MNIST` and `CIFAR10`.

B.3.3 Number of Training Epochs.

Figure B.4 shows that increasing the number of training epochs during model updates from 5 to 20 reduces the backdoor survivability in most cases. While we are multiplying the training efforts, the impact on the backdoor survivability is very limited. This is likely because although additional training epochs allows the model to learn more new data features (and thus forget existing features like the backdoor faster), the model weights might be trapped in the local minimum in DNNs.

Models	Neural Cleanse	TABOR
Clean	1.52	1.84
Badnets	3.13	2.12
Blend	3.35	3.11
Wanet	2.44	2.01

Table B.3: Average anomaly index for Neural Cleanse and TABOR on clean models and backdoored models (F_0). The attack success rate threshold is set to 99%. For each type of models, we average the anomaly index over 5 models.

Models	Neural Cleanse	TABOR
Badnets	**1.00**	**1.54**
Blend	**1.52**	2.12
Wanet	**0.87**	**1.45**

Table B.4: Average anomaly index for Neural Cleanse and TABOR on fine-tuned backdoored models after poison stops (F_1, the backdoors are injected in F_0). The attack success rate threshold is set to 99%. **Bold numbers** indicate failures to detect the backdoors.

B.3.4 Existing Defenses

We test Neural Cleanse and TABOR with the three attacks (Badnets, Blend and Wanet) on CIFAR10 dataset. We first run the two defenses on clean and backdoored initial models (F_0s). As shown in Table B.3, when we set the attack success rate threshold to 99% as guided in the original papers [164, 56], both Neural Cleanse and TABOR work well on all three attacks (both defenses suggest that models with anomaly index over 2 are detected as backdoored). However, when we run Neural Cleanse and TABOR on the fine-tuned backdoored models after the poison stops, Table B.4 shows that Neural Cleanse and TABOR fail to detect the backdoor models in most cases (only TABOR can detect the backdoored models attacked by Blend).

A straightforward way of adapting the defenses is to decrease the attack success rate threshold to a smaller threshold, like 75%, 50% or 25%, since the attack success rate on fine-tuned models are lower than 99%. The defender can set the threshold according to their need. However, our results imply that this direct adaptation does not work. Table B.5 shows

Attack Success Rate Threshold / Models	75%		50%		25%	
	`Neural Cleanse`	`TABOR`	`Neural Cleanse`	`TABOR`	`Neural Cleanse`	`TABOR`
Clean	1.26	1.90	**2.27**	1.34	1.53	**3.31**
Badnets	3.38	2.89	3.23	2.06	**1.70**	2.41
Blend	2.29	2.46	2.11	3.02	**1.28**	2.01
Wanet	**1.64**	**1.86**	**1.53**	**1.52**	**1.63**	**1.51**

Table B.5: Average anomaly index for `Neural Cleanse` and `TABOR` on clean and backdoored models (F_0) with different attack success rate threshold. **Numbers in bold** indicate instances where either the backdoors were not detected or false positive detections of backdoors were made on clean models.

that when reducing the attack threshold, the detection results are unstable in two aspects: 1) the anomaly index for clean models may increase over 2 (attack success rate threshold as 50% for `Neural Cleanse` and attack success rate threshold 25% for `TABOR`); 2) the anomaly index for backdoored model may drop below 2.

APPENDIX C

BLACKLIGHT: SCALABLE DEFENSE FOR NEURAL NETWORKS AGAINST QUERY-BASED BLACK-BOX ATTACKS

This appendix consists of the following items:

- §C.1 presents the full detail of our formal analysis mentioned in §5.7, including the key theoretical result and its proof and how these results can be used to guide Blacklight's parameter configuration;
- §C.2 describes how ensemble adversarial training can be combined with Blacklight as a hybrid defense against both substitute model attacks and query-based black-box attacks (mentioned in §2.3);
- §C.3 is a supplement of §5.4 by providing detailed evaluation of SD and PRADA under a persistent attacker, who switches to a new account when the current account is detected as adversarial and thus banned.
- §C.4 show the results where we empirically verify the two assumptions we made in §5.6.
- §C.5 summarizes the experimental configurations used by our experiments, including classification tasks, datasets, model training configurations, model architectures, and attack perturbation budgets, as well as Blacklight's configuration.
- §C.6 includes additional results for Blacklight's performance on detection and mitigating Boundary Attacks with 1 million query limit (mentioned in §5.8.2), universal adversarial patches (discussed in §5.8.3) and the impact for Blacklight parameter settings (discussed in §5.8.5).
- §C.7 provides detailed discussion and experimental results on adaptive attacks discussed in §5.9, including 5 subsections: Evasion via Image Transformations, Increasing Per-

turbation Budget, Guided Transformations when Attacker Knows $(\mathbf{q}, \mathbf{p}, \mathbf{w})$, Optimal Black-Box Attacks and Pause and Resume Attacks.

C.1 Formal Analysis of Blacklight

We formally examine Blacklight by modeling its process of probabilistic fingerprinting. We derive analytical bounds on the probability of Blacklight flagging a query pair (x, y) as attacks, and subsequently estimate Blacklight's false positive rate and attack query detection coverage.

C.1.1 Definitions

We first introduce the terms that we will use to model the proposed probabilistic fingerprinting process on input queries.

Definition C.1.1. Hash Function is a function that, for a given input x, produces $\mathbf{N}$ hash values, $\mathbf{H}_x = (h_1, h_2, ..., h_{\mathbf{N}})$. Each entry h_i is a positive integer that is independent and identically distributed (I.I.D.) in the hash space $[1, \Omega]$, where Ω is a very large positive integer, $\Omega >> \mathbf{N}$. Without loss of generality, h_i follows a uniform distribution within $[1, \Omega]$.

Definition C.1.2. Given two queries x and y, we represent their full hash set as $\mathbf{H}_x = \mathbf{H}_{sh} \cup \hat{\mathbf{H}}_x$ and $\mathbf{H}_y = \mathbf{H}_{sh} \cup \hat{\mathbf{H}}_y$, where $\mathbf{H}_{sh} = \mathbf{H}_x \cap \mathbf{H}_y$, $\hat{\mathbf{H}}_x \cap \hat{\mathbf{H}}_y = \emptyset$, $|\mathbf{H}_{sh}| = N - \Delta$, $|\hat{\mathbf{H}}_x| = |\hat{\mathbf{H}}_y| = \Delta$. For simplicity, we assume $\mathbf{H}_{sh} \cap \hat{\mathbf{H}}_x = \emptyset$, $\mathbf{H}_{sh} \cap \hat{\mathbf{H}}_y = \emptyset$.

Note that Δ represents the amount of full hash differences between x and y. We also empirically validate the assumption of $|\mathbf{H}_{sh} \cap \hat{\mathbf{H}}_x|/|\mathbf{H}_x| \approx 0$ on `CIFAR10`.

Definition C.1.3. Probabilistic Fingerprinting (PF) is a function performed on the full hash set that samples top S hash entries out of $\mathbf{H}_x$, *i.e.* $\mathbb{S}(\mathbf{H}_x) = (h'_1, h'_2, ..., h'_S) \subset \mathbf{H}_x$.

Finally, Blacklight operates on $\mathbb{S}(\mathbf{H}_x)$ to detect attack queries rather than the full hash set $\mathbf{H}_x$. Blacklight marks (x, y) as attack images if $|\mathbb{S}(\mathbf{H}_x) \cap \mathbb{S}(\mathbf{H}_y)| > \mathbf{T}$.

C.1.2 Key Results

Our analysis led to the following theorem.

Theorem C.1.4. *Let $Q(\Delta)$ be the probability of Blacklight flagging a query pair (x, y) as attack queries where x and y's full hashes differ by Δ entries. Then $Q(\Delta) \leq Q^{upper}(\Delta)$.*

$$
\begin{aligned}
Q(\Delta) &\triangleq Pr\left(Blacklight\,(x,y) = attack \mid diff\,(\mathbf{H}_x, \mathbf{H}_y) = \Delta\right) \\
Q^{upper}(\Delta) &= \sum_{k=\mathbf{T}+1}^{\min(\mathbf{S},\mathbf{N}-\Delta)} \binom{\mathbf{N}-\Delta}{k} \cdot \binom{\Delta}{\mathbf{S}-k} \Big/ \binom{\mathbf{N}}{\mathbf{S}}
\end{aligned}
\tag{C.1}
$$

where $\mathbf{N}$*,* $\mathbf{S}$ *and* $\mathbf{T}$ *are parameters of Blacklight (see §5.6.).*

Proof. Clearly $\mathbb{S}(\mathbf{H}_x) = \mathbb{S}(\mathbf{H}_{sh} \cup \hat{\mathbf{H}}_x)$ will contain entries from $\mathbf{H}_{sh}$ and $\hat{\mathbf{H}}_x$. The same applies to $\mathbb{S}(\mathbf{H}_y)$. Since $\hat{\mathbf{H}}_x \cap \hat{\mathbf{H}}_y = \emptyset$, the overlapping entries of $\mathbb{S}(\mathbf{H}_x)$ and $\mathbb{S}(\mathbf{H}_y)$ will only come from $\mathbf{H}_{sh}$. That is,

$$(\mathbb{S}(\mathbf{H}_x) \cap \mathbb{S}(\mathbf{H}_y)) \subset \mathbf{H}_{sh}, \tag{C.2}$$

$$(\mathbb{S}(\mathbf{H}_x) \setminus \mathbb{S}(\mathbf{H}_y)) \subset \hat{\mathbf{H}}_x \tag{C.3}$$

$$(\mathbb{S}(\mathbf{H}_y) \setminus \mathbb{S}(\mathbf{H}_x)) \subset \hat{\mathbf{H}}_y \tag{C.4}$$

To calculate the upper bound on $Pr(|\mathbb{S}(\mathbf{H}_x) \cap \mathbb{S}(\mathbf{H}_y)| > \mathbf{T})$, we consider the "optimal scenario" using a custom-designed[1] probabilistic fingerprinting process, so that when picking entries from $\mathbf{H}_x$ and $\mathbf{H}_y$, the chosen entries in $\mathbf{H}_{sh}$ are always the same for x and y. This is to maximize the similarity between $\mathbb{S}(\mathbf{H}_x)$ and $\mathbb{S}(\mathbf{H}_y)$, which will be higher than that offered by selecting top $\mathbf{S}$ entries. Thus we compute the upper bound as the probability of more than $\mathbf{T}$ entries in $\mathbb{S}(\mathbf{H}_x)$ (and $\mathbb{S}(\mathbf{H}_y)$) come from $\mathbf{H}_{sh}$ and the rest come from $\hat{\mathbf{H}}_x$ ($\hat{\mathbf{H}}_y$).

Since each hash entry's value is i.i.d., and $|\mathbf{H}_{sh}| = \mathbf{N} - \Delta$, $|\hat{\mathbf{H}}_x| = \Delta$, we calculate the

1. One possible design is picking hash entries by their indices. If the fingerprinting process chooses the same set of hash indices for x and y, the chosen entries in $\mathbf{H}_{sh}$ will be the same for x and y.

probability following the hypergeometric distribution and arrive at the upper bound shown in the theorem. □

Key Observation: $Q^{upper}(\Delta)$ Decaying Fast with Δ. While unable to simplify its symbolic expression, we empirically found that $Q^{upper}(\Delta)$ can be approximated by a symmetrical sigmoidal function of Δ (with the goodness of fit R^2=0.9996). For instance, consider two configurations that Blacklight uses to scan `CIFAR10` image queries: $\mathbf{N} = 3053$ ($w = 20$, $p = 1$), $\mathbf{S} = 50$, and $\mathbf{T}$ =25 or 40. Then $Q^{upper}(\Delta)$ can be approximated as:

$$Q^{upper}(\Delta) \approx \begin{cases} 1.011 \cdot \left(1 + \left(\frac{\Delta}{1494.85}\right)^{11.77}\right)^{-1} - 0.013, & \mathbf{T} = 25 \\ 1.006 \cdot \left(1 + \left(\frac{\Delta}{584.51}\right)^{5.97}\right)^{-1} - 0.009, & \mathbf{T} = 40 \end{cases}$$

Note that we followed the standard curve fitting process to approximate $Q^{upper}(\Delta)$, and 1.011/1.006 are function parameters generated by curve fitting. For the above configurations, Figure C.1 plots the upper bound $Q^{upper}(\Delta)$ as a function of Δ and also $Q(\Delta)$ measured by running Blacklight on both benign and attack queries generated from `CIFAR10`. We see that the upper bound is reasonably tight. More importantly, both decay very fast with Δ.

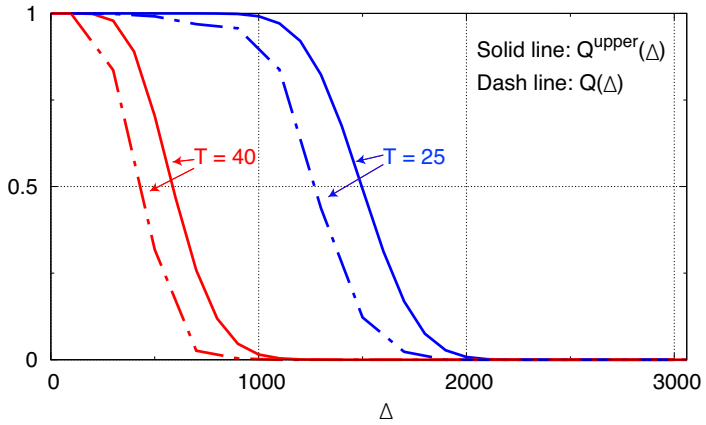


Figure C.1: *Measured $Q(\Delta)$ and its theoretical upper-bound $Q^{upper}(\Delta)$, both decaying fast with Δ. The results are for* `CIFAR10` *queries ($\mathbf{N} = 3053, \mathbf{S} = 50, \mathbf{T} = 25$ or 40).*

Blacklight's Detection Coverage & False Positive Rate. We model Blacklight's two performance metrics from $Q(\Delta)$:

$$\text{False positive rate} \leq Q(\Delta_{benign})$$
$$\text{Attack query detection coverage} \geq Q(\Delta_{attack})$$

where the attack query detection coverage is the probability of detecting a pair of attack queries as adversarial, and the false positive rate is the probability of detecting a pair of benign queries as adversarial. And Δ_{benign} is the minimum full hash difference between benign queries, and Δ_{attack} is the maximum full hash difference between attack queries. Since $Q(\Delta)$ decays fast with Δ, a properly configured Blacklight system can effectively detect attack queries at a low false positive rate, i.e., $Q(\Delta_{attack}) \rightarrow 1,\ Q(\Delta_{benign}) \rightarrow 0$, as long as $\Delta_{benign} >> \Delta_{attack}$.

C.1.3 Guiding the Parameter Configuration

Our analysis shows that Blacklight's system parameters will impact $Q^{upper}(.)$ and thus its false positive rate and detection coverage. We leverage this modeled relationship to guide Blacklight's parameter configuration. The goal is to meet a desired false positive while maximizing detection coverage.

Choosing T. Among the five parameters, **T** is of particular importance since it defines the threshold of fingerprint matching. Figure C.1 already shows that **T** can largely alter the range of $Q^{upper}(.)$. To select **T**, we propose to examine the model's training data to compute Δ_{benign} and use it to choose **T** to meet a desired false positive rate. For example, when $\mathbf{N} = (|x| - \mathbf{w} + \mathbf{p})/\mathbf{p}$=3053, the full hashes of quantized images in `CIFAR10` produce Δ_{benign}=2638. Thus if **S**=50, then $\mathbf{T} = mathbfS/2 = 25$ should produce a reasonably small false positive rate while maintaining a high detection rate.

Choosing w, p, q, S. We divide these parameters into three groups: [**w**, **p**], [**q**], and [**S**]. Group one decides how to map a query into a full set of **N** hashes, and how many hashes a single pixel could affect. The second group (**q**), together with the first group, controls the full hash similarity among attack and benign queries, i.e., Δ_{attack} and Δ_{benign} defined by our formal analysis. Finally, **S** (and **T**) determine how to compare queries' similarity by their hashes. With these in mind, we propose the following guidelines.

We should choose **q** as a moderate value to make attack queries' hashes highly similar (i.e., small Δ_{attack} thus high detection coverage $Q(\Delta_{attack})$), but not too large to diminish the difference between benign queries (i.e., large Δ_{benign} to maintain a low false positive rate approximated by $Q(\Delta_{benign})$).

S should be much less than **N** for scalability, yet not too small so the fingerprint has enough capacity to capture the difference between benign queries, thus keeping Δ_{benign} sufficiently large to maintain a low false positive rate $Q(\Delta_{benign})$.

The choice of **w** could affect both false positive rate and detection coverage. The larger the **w**, the more hashes that changing one pixel will affect, and more sensitive the fingerprint will react to content variation, thus increasing Δ_{attack} and Δ_{benign}. As such, increasing w will reduce both false positive $Q(\Delta_{benign})$ and detection coverage $Q(\Delta_{attack})$. Ideally, one should choose the smallest w that meets the desired false positive rate.

C.2 Hybrid Defense against the Substitute Model Attack

Blacklight is designed to detect query based black-box attacks. It cannot defend against attacks transferred from a substitute model. As we discussed in §2.3, substitute model attacks can be effectively stalled by an existing defense called ensemble adversarial training (EAT) [158]. EAT adversarially trains an ensemble of models with different architectures [105], which are shown to be robust against the substitute model attack. Hence, to defend against all types of black-box attacks, the defender can combine Blacklight with EAT to build a

Attack	Detection coverage	Avg queries to detect	Attack success w. mitigation	Attack success w/o mitigation
NES - QL	1.8% / 0.8%	52 / 112	97% / 97%	97%
NES - LO	1.3% / 0.9%	52 / 111	85% / 85%	85%
Boundary	1.0% / 0.8%	54 / 115	86% / 86%	86%
ECO	1.8% / 0.9%	53 / 112	88% / 88%	88%
HSJA	1.7% / 0.9%	52 / 111	100% / 100%	100%
QEBA	1.6% / 0.9%	52 / 111	100% / 100%	100%
SurFree	1.9% / 0.9%	52 / 111	100% / 100%	100%
Policy-Driven	2.1% / 0.9%	53 / 111	98% / 98%	98%

Table C.1: *Detection performance of Stateful Detection [31] and PRADA [74] when attackers change their accounts after detected and disabled on* `CIFAR10`*, in terms of attack detection and mitigation. The result is presented as "Stateful Detection / PRADA".*

hybrid defense system.

We build and evaluate a hybrid Blacklight and EAT defense on the cifar task. Specifically, we build an ensemble model with three different architectures (6-layer CNN, 8-layer CNN, ResNet-20) and adversarially train the network using PGD attacks as suggested by [105]. We use the same Blacklight configuration as before.

We perform both substitute model based attacks and query based black-box attacks against the above ensemble model defended by Blacklight. For the substitute model attack we run the state-of-art attack proposed by Papernot et al [123], and for the query-based attacks we run the same five black-box attacks. The result shows that the hybrid defense works well and the two defenses do not interfere with each other. The substitute model attack achieves 0% success (thanks to EAT), and Blacklight achieves the same accurate attack query detection as reported before. Thus, we conclude that Blacklight, when combined with EAT, can defend against today's black-box attacks.

C.3 Additional Results for §5.4

We show the detection performance of SD and PRADA under the assumption where attackers will switch to a new account when the current account is detected as malicious and banned in Table C.1.

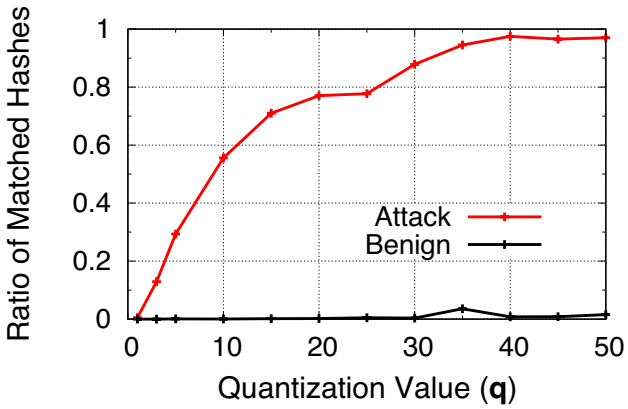


Figure C.2: *Average ratio of matched hashes in fingerprints of 10,000 pairs of quantized attack queries and 10,000 pairs of quantized benign queries, all for* `CIFAR10`*, when varying the quantization step (*$\mathbf{q}$*). We can see that quantization does increase the similarity between attack query fingerprints but have 'negligible' impact on beign query fingerprints.*

C.4 Additional Results for §5.6

We empirically validate two assumptions we make in §5.6.

- Quantization increases the similarity between attack queries. We empirically validate it by showing the average number of matched hashes in fingerprints of attack/benign queries with different quantization step ($\mathbf{q}$). As shown in Figure C.2, quantization not only increase the similarity between attack queries, but also have little impact on benign queries, which is ideal for attack detection.

- Highly similar (quantized) queries will have highly similar fingerprints. We empirically verify this with Figure C.3. We can see that the images with smaller L_2 distances have a higher ratio of hashes match in their fingerprints.

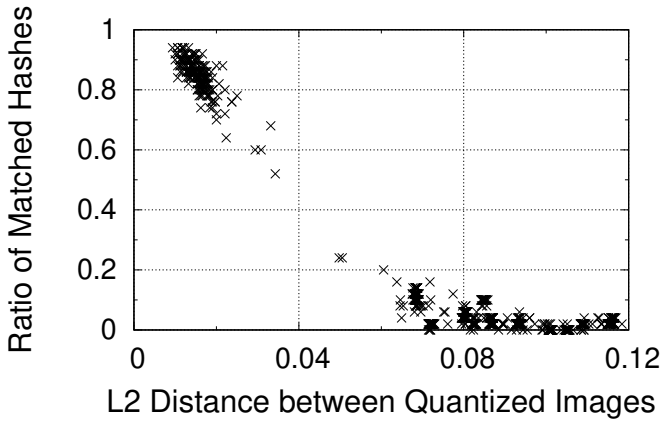


Figure C.3: *We empirically demonstrate that highly similar image queries (after quantization) also have highly similar fingerprints, based on 10000 pairs of attack queries on* `CIFAR10`*. In x-axis we plot the* L_2 *distance between a pair of attack queries after they are quantized, and in y-axis, we plot the ratio of matching in their probabilistic fingerprints. We see that the two metrics are strongly (negatively) correlated.*

C.5 Experimental Configurations

C.5.1 Classification Tasks and Models

Table C.2 summarizes the four image classification tasks that we use for our experiments. Their associated models are listed below:

- `MNIST` (Table C.3) is a convolutional neural network (CNN) consisting of two pairs of convolutional layers connected by max pooling layers, followed by two fully connected layers.
- `GTSRB` (Table C.4) is a CNN consisting of three pairs of convolutional layers connected by max pooling layers, followed by two fully connected layers.
- `CIFAR10` is a ResNet-20 [58] that includes 20 sequential convolutional layers, followed by pooling, dropout, and fully connected layers.
- `ImageNet` is the ResNet-152 [57] model trained on the ImageNet dataset [141]. It has 152 residual blocks with over 60 million parameters.

Task	Dataset	# Classes	Training data size	Test data size	Input size	Model architecture	Model accuracy
Digit Recognition (`MNIST`)	MNIST	10	60,000	10,000	(28, 28, 1)	6 Conv + 3 Dense	99.36%
Traffic Sign Recognition (`GTSRB`)	GTSRB	43	39,209	12,630	(48, 48, 3)	6 Conv + 3 Dense	97.59%
Object Recognition - Small (`CIFAR10`)	CIFAR-10	10	50,000	10,000	(32, 32, 3)	ResNet20	91.48%
Object Recognition - Large (`ImageNet`)	ImageNet	1000	1,281,167	50,000	(224, 224, 3)	ResNet152	73.05%

Table C.2: *Overview of image classification tasks with their associated datasets and models.*

Layer Index	Layer Name	Layer Type	# of Channels	Filter Size	Activation	Connected to
1	conv_1	Conv	32	3× 3	ReLU	
2	conv_2	Conv	32	3× 3	ReLU	conv_1
2	pool_1	MaxPool	32	2× 2	-	conv_2
3	conv_3	Conv	64	3× 3	ReLU	pool_1
4	conv_4	Conv	64	3× 3	ReLU	conv_3
4	pool_2	MaxPool	64	2× 2	-	conv_4
5	conv_5	Conv	128	3× 3	ReLU	pool_2
6	conv_6	Conv	128	3× 3	ReLU	conv_5
6	pool_3	MaxPool	128	2× 2	-	conv_6
7	fc_1	FC	512	-	ReLU	pool_3
8	fc_2	FC	512	-	ReLU	fc_1
8	fc_3	FC	10	-	Softmax	fc_2

Table C.3: *Model Architecture for* `MNIST`.

C.5.2 Black-box Attack and Blacklight Configurations

Attack Configurations. We set the L distance metrics and perturbation budgets for different attacks following Table C.6, C.7. In these tables, $\mathrm{L}_\infty(x, x') = \max_i(|x_i - x'_i|)$ and normalized L_2 distance, i.e., *normalized* $\mathrm{L}_2(x, x') = \sqrt{\frac{1}{|x|}\sum_{i=0}^{|x|}(x_i - x'_i)^2}$.

Blacklight Configurations. We list the default parameter configurations for Blacklight in Table C.8. We discuss the impact of those parameters in §5.8.5 and §C.1. we include Blacklight configurations for both 4 image classification tasks and the text classification task we use in §5.8.7.

C.6 Additional Results for §5.8 Evaluation

We now present additional results for §5.8 including Blacklight performance on boundary attacks with 1 million query limits, Blacklight performance on universal patch attacks, and the detailed results for Blacklight parameter configuration impacts.

Layer Index	Layer Name	Layer Type	# of Channels	Filter Size	Activation	Connected to
1	conv_1	Conv	32	3× 3	ReLU	
2	conv_2	Conv	32	3× 3	ReLU	conv_1
2	pool_1	MaxPool	32	2× 2	-	conv_2
3	conv_3	Conv	64	3× 3	ReLU	pool_1
4	conv_4	Conv	64	3× 3	ReLU	conv_3
4	pool_2	MaxPool	64	2× 2	-	conv_4
5	conv_5	Conv	128	3× 3	ReLU	pool_2
6	conv_6	Conv	128	3× 3	ReLU	conv_5
6	pool_3	MaxPool	128	2× 2	-	conv_6
7	fc_1	FC	512	-	ReLU	pool_3
8	fc_2	FC	512	-	ReLU	fc_1
8	fc_3	FC	43	-	Softmax	fc_2

Table C.4: *Model Architecture for* `GTSRB`.

Model	Training Configuration
`MNIST`	epochs=50, batch=128, optimizer=Adam, lr=0.001
`GTSRB`	epochs=50, batch=128, optimizer=Adam, lr=0.001
`CIFAR10`	epochs=200, batch=32, optimizer=Adam, lr=0.001 (learning rate reduced after 80, 120, 160, 180 epochs)
`ImageNet`	Model trained and shared by He *et al.* [57]

Table C.5: *Detailed information on model training configurations for image classification tasks.*

Boundary attacks with 1 million queries. Table C.9 shows that blacklight still has 100% attack detect rates for boundary attacks with 1 million query limits. Furthermore, we find that the detection coverages are even higher for attacks with 1 million query limits than those with 100K query limits. This validates our hypothesis that Blacklight detects boundary attacks at later stage because boundary attack advances slower in converging to the successful adversarial examples. Finally, boundary attacks still have 0% attack success rate with Blacklight mitigation even with 1 million queries.

Blacklight's performance on universal patch attack. Table C.10 lists the detailed results for Blacklight's detection and mitigation results on Sparse-RS universal patch attack.

Impacts for Blacklight parameter configuration. We show the experimental results for the impact of Blacklight parameters (Quantization step (**q**), # of hashes per fingerprint (**S**), Sliding window size (**w**), and Sliding step (**p**)) by plotting the Detection Coverage (%) and False Positive Rate (%) with different parameter settings in Figure C.4.

Attack	Distance Metric	Perturbation Budget	Attack	Distance Metric	Perturbation Budget
NES - QL	L_∞	0.05 (0.1 for `MNIST`)	NES - LO	L_∞	0.05 (0.1 for `MNIST`)
Boundary	*normalized* L_2	0.05	ECO	L_∞	0.05 (0.1 for `MNIST`)
HSJA	*normalized* L_2	0.05	QEBA	*normalized* L_2	0.05
SurFree	*normalized* L_2	0.05	Policy-Driven	*normalized* L_2	0.05

Table C.6: *Black-box attack configurations. For brevity, we report the* normalized L_2 *distance in the Perturbation Budget for L_2 distance metric since L_2 distance varies a lot according to input sizes. We report the corresponding L_2 distance for different tasks in Table C.7.*

Task	*normalized* L_2	L_2	Task	*normalized* L_2	L_2
`MNIST`	0.05	1.4	`GTSRB`	0.05	4.2
`CIFAR10`	0.05	2.8	`ImageNet`	0.05	19.4

Table C.7: *The* normalized L_2 *distance and corresponding L_2 distance budgets for different tasks we use in our experiments.*

C.7 Additional Results for §5.9 Adaptive Attacks

We now provide more analysis on countermeasures.

C.7.1 Evasion via Image Transformations.

We report the details for our experiments against Image Transformations here. After applying these transformations to attack queries, we report the attack success rate (without the Blacklight defense) and Blacklight's attack detection rate, on the `CIFAR10` task. Like before, we report attack detection rate only *successful attacks*. For each setting, we run 20 attack instances.

For Gaussian noise based transformations, we vary the standard deviation (STD) of noise from 0.0001 to 0.05 (with all query inputs normalized to [0,1]). Results in Table C.11 show that as noise levels increase, attack success rates drop quickly. But at all noise levels tested, Blacklight is able to detect all successful attacks. Intuitively, sufficiently high noise will disrupt classification of both benign and attack queries, thus degrading the attack success rate. We see that Blacklight is generally more robust than the attack's iterative optimization

Task	Image classification				Text classification
	MNIST	GTSRB	CIFAR10	ImageNet	IMDB
Quantization step (**q**)	50	50	50	50	50
Sliding window size (**w**)	50	20	20	50	10
Sliding step (**p**)	1	1	1	1	1
# of hashes per fingerprint (**S**)	50	50	50	50	30
Matching threshold (**T**)	25	25	25	25	15

Table C.8: *Experiment configuration of Blacklight.*

Task	w. Detection			w. Mitigation	w/o Blacklight	
	Attack detect %	Detection coverage	Avg queries to detect	Attack success	Attack success	Avg # attack queries
MNIST	100%	76.3%	16	0%	26%	892350
GTSRB	100%	71.2%	19	0%	40%	902931
CIFAR10	100%	69.7%	27	0%	96%	829124
ImageNet	100%	97.2%	39	0%	79%	738452

Table C.9: *Blacklight's detection and mitigation results on Boundary attack. We stop the boundary attack if it is no successful after 1 million attack queries.*

process – Blacklight continues to detect attacks at noise levels where the noise has long since disrupted the attack.

For image augmentation, we test 4 cases where the attacker shifts each input horizontally/vertically by up to 10%, rotate by up to 10°, zoom in by up to 10%, and a combination of all three. Table C.11 shows that while different attacks react differently to image augmentation techniques (some still produce successful attacks while others fail completely), Blacklight is able to detect all successful attack sequences under different transformations.

C.7.2 Increasing Perturbation Budget

In order to provide a comprehensive evaluation on the impact of increasing perturbation budget on the detection performance for Blacklight, we run experiments on all tasks for the two fastest converging attacks (ECO and HSJA) with larger perturbation budgets. Table C.12 shows that Blacklight achieves 100% on all tasks for ECO attacks even with perturbation

Task	w. Detection			w. Mitigation	w/o Blacklight	
	Attack detect %	Detection coverage	Avg queries to detect	Attack success	Attack success	Avg # attack queries
MNIST	100%	98.4%	8	0%	32.9%	88021
GTSRB	100%	98.9%	14	0%	10.8%	98386
CIFAR10	100%	97.6%	12	0%	54.7%	87201
ImageNet	100%	98.7%	9	0%	27.7%	92039

Table C.10: *Blacklight's detection and mitigation results on Sparse-RS universal patch attack.*

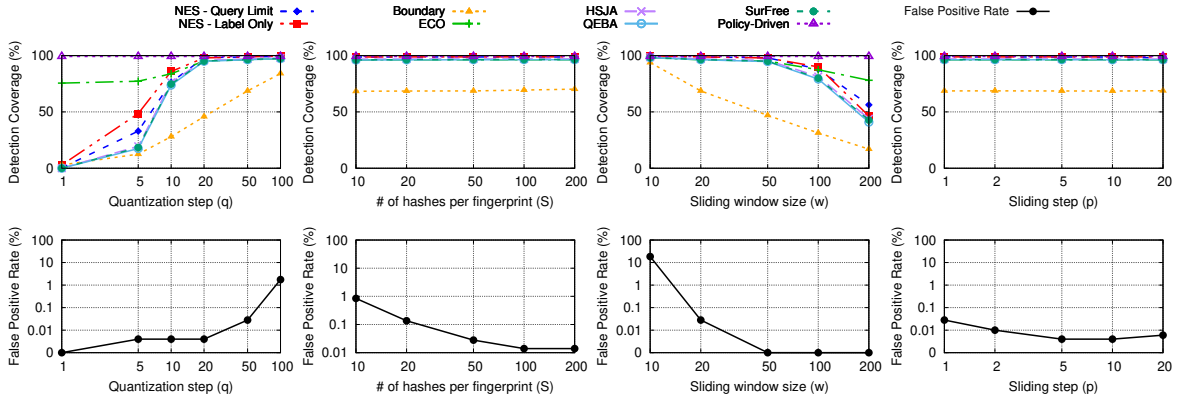


Figure C.4: *Detection Coverage (%) and False Positive Rate (%) with different settings on Blacklight parameters: Quantization step (***q***), # of hashes per fingerprint (***S***), Sliding window size (***w***), and Sliding step (***p***).*

budget up to 0.2. For HSJA attack, Blacklight can detect 100% of attacks on all tasks when the *normalized* L_2 perturbation budgets are no more than 0.1. When the *normalized* L_2 perturbation budgets get larger, Blacklight's detection rate drops gradually. However, we believe this is reasonable since the *normalized* L_2 budget is too large that even exceeds the *normalized* L_2 distances between some benign images.

We analyze the distribution of the *normalized* L_2 distances between benign images from different labels. We randomly pick 50K benign image pairs from different labels and calculate the *normalized* L_2 distances between these pairs. Figure C.5 shows the distribution for the *normalized* L_2 distances between benign images from different labels for all four tasks. We can see that for all tasks, the majority of image pairs have a *normalized* L_2 distance no more than 0.4 and there are a significant proportion of benign image pairs having a *normalized* L_2 distance no more than 0.3.

Transformation / Attack		Gaussian Noise w. Different STD				Image Augmentation			
		0.0001	**0.0005**	**0.005**	**0.05**	**Shift**	**Rotate**	**Zoom**	**Comb.**
NES - QL	**ASR**	85%	80%	15%	0%	100%	75%	80%	60%
	ADR	100%	100%	100%	N/A	100%	100%	100%	100%
NES - LO	**ASR**	25%	20%	15%	0%	100%	45%	70%	20%
	ADR	100%	100%	100%	N/A	100%	100%	100%	100%
Boundary	**ASR**	90%	90%	85%	0%	90%	90%	90%	90%
	ADR	100%	100%	100%	N/A	100%	100%	100%	100%
ECO	**ASR**	85%	0%	0%	0%	0%	0%	0%	0%
	ADR	100%	N/A	N/A	N/A	N/A	N/A	N/A	N/A
HSJA	**ASR**	95%	20%	5%	0%	0%	5%	10%	15%
	ADR	100%	100%	100%	N/A	N/A	100%	100%	100%

Table C.11: *Attack success rate (ASR) w/o Blacklight mitigation and Blacklight attack detection rate (ADR) of successful attacks as attackers add different image transformations. Column 3-6 report the results for adding Gaussian Noise with different standard deviation (STD) and Column 7-10 report the results for applying different image transformations to each attack queries.*

Task	ECO				HSJA			
	0.05	0.1	0.15	0.2	0.05	0.1	0.15	0.2
`MNIST`	100%	100%	100%	100%	100%	100%	75%	40%
`GTSRB`	100%	100%	100%	100%	100%	100%	70%	50%
`CIFAR10`	100%	100%	100%	100%	100%	100%	80%	40%
`ImageNet`	100%	100%	100%	100%	100%	100%	90%	85%

Table C.12: *Blacklight detection rate for attacks using larger perturbation budgets for all tasks. We use* $\mathbf{T} = 15$ *with a small increase in false positives (0.74%).*

We say that a reasonable L_2 perturbation budget for adversarial examples should smaller than half of the *normalized* L_2 distances for most of the benign image pairs. Otherwise, by simply blending two benign images (calculating the mean of two images), the attacker can create a successful adversarial example: assume the attacker has an image pair (x_a, x_b), the model $\mathbb{F}$ classifies x_a to label A and x_b to label B, $x' = \frac{x_a + x_b}{2}$ cannot be classified both to label A and B. If *normalized* $L_2(x_a, x')$ is smaller than perturbation budget, x' is an adversarial example for target label $\mathbb{F}(x')$ either from original image x_a or from image x_b.

Figure C.6 shows examples where the attacker can successfully create an adversarial example by simply calculating the average of two benign images when the *normalized* L_2 budget is set to 0.2. In such case, although the attack will succeed within one single query, we believe this is not a reasonable perturbation budget for adversarial attacks.

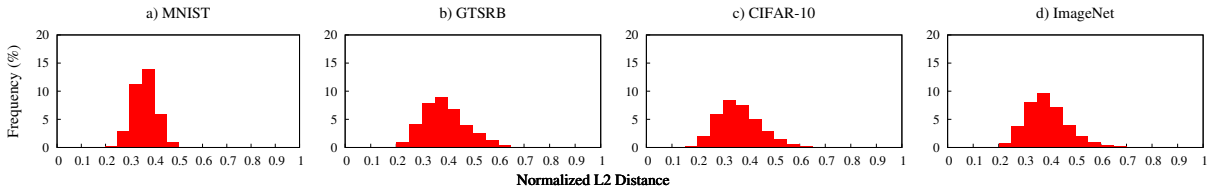


Figure C.5: *Frequency of the* normalized L_2 *distances between benign images from different labels for all tasks.*

C.7.3 Guided Transformations when Attacker Knows $(\mathbf{q}, \mathbf{p}, \mathbf{w})$*.*

We show the algorithm we use for guided transformations when attacker knows $(\mathbf{q}, \mathbf{p}, \mathbf{w})$ in Algorithm 1.

C.7.4 Optimal Black-Box Attacks.

We give more discussion in optimal black-box attacks. First, to simulate a near-optimal query-efficient attack, we evenly downsample attack query sequences from 5 attacks to generate attack sequences that are a tiny fraction of current sequences. We then test Blacklight's detection performance on these subsampled attack sequences. Table C.13 shows that even when attacks are able to complete in 500, 100, or 50 queries, Blacklight still detects them near perfectly (100% detection for 4 attacks and 89% for Boundary attack). Even when these attacks complete within 10 queries, Blacklight is still highly successful at detecting NES-QL, ECO and HSJA.

We note that NES-LO and Boundary attacks have much lower detection rates than other attacks when only choosing 10 queries from attack sequences. This is because both NES-LO and Boundary attacks are both boundary attacks that jump back and forth between two images (original and target image). Random subsets of 10 out of thousands of queries are more likely to be variants of the source or target that are sufficiently different from each other as to avoid detection.

Second, for "perfect-gradient" black-box algorithm, each iteration of the gradient calcula-

Algorithm 1 Algorithm for Guided Transformation Attacks when Attacker Knows $(\mathbf{q}, \mathbf{p}, \mathbf{w})$

Parameter: Sliding window size of Blacklight: $\mathbf{w}$, quantization step of Blacklight: $\mathbf{q}$
Input: Attack query x
Output: Modified attack query x

procedure INITIZATION($\mathbf{w}$, $\mathbf{q}$)
    # We save all combinations for pixels modification in a queue.
    $PermList \leftarrow []$
    for $i = 1$ **to** $\mathbf{w}$ **do**
        # compute all combinations for selecting i pixels from $\mathbf{w}$ pixels, which generates $C^i_{\mathbf{w}}$ choices.
        pixelCombination = Combination(i, w)
        # For each pixel selected there are 2 choices for combinations $(+\mathbf{q}/-\mathbf{q})$, which generates $2^i \times C^i_{\mathbf{w}}$ choices in total.
        allPixelCombination = Update2ChoicesPerPixel (pixelCombination)
        PermList.append(allPixelCombination)
    end for
    return PermList
end procedure
procedure GUIDEDTRANSFORMATION(x)
    # we pop the first element from the queue, which is the modification choice with smallest number of pixel changes in the remaining choices.
    CurrentPermutation = PermList.pop()
    Apply the modification for CurrentPermutation to every $\mathbf{w}$ pixels of x.
    return x
end procedure

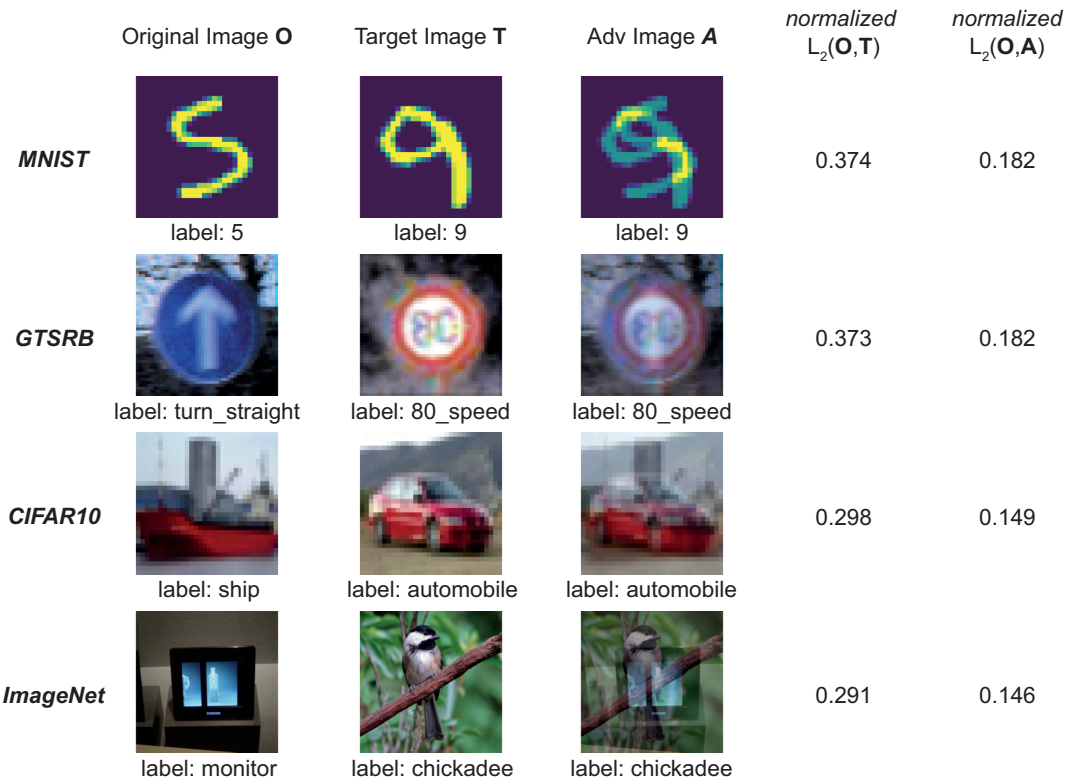


Figure C.6: *Examples of successful adversarial attacks via blending two benign images when the perturbation budget is set to* 0.2.

tion for an analogous white-box attack would translate to a single query over the network by the black-box attacker. This idealized black-box attack uses CW [23] and PGD [105] to generate attack sequences against our `CIFAR10` model. On average, CW and PGD converge after only 6.3 and 3.1 queries. Against simulated black-box attacks using these attack queries, Blacklight detects 100% of attacks driven by CW, and 81% of PGD-driven attacks.

C.7.5 Pause and Resume Attacks.

Table C.14 shows the exact number of average reset cycles needed for different attacks. We also include average total queries needed for attacks as reference.

Attack Type \ N	500	100	50	10
NES - Query Limit	100%	100%	100%	95%
NES - Label Only	100%	100%	100%	31%
Boundary	100%	90%	89%	48%
ECO	100%	100%	100%	100%
HSJA	100%	100%	100%	91%
CW	Average $N = 6.33$, Detection rate = 100%			
PGD	Average $N = 3.13$, Detection rate = 81%			

Table C.13: *Blacklight's performance against near-optimal "query-efficient" and "perfect-gradient" black-box attacks.*

Attack Type	Average Reset Cycles Needed	Average Total Queries
NES-QL	11471	12695
NES-LO	65837	67099
Boundary	2285	6160
ECO	16590	16591
HSJA	1092	1121

Table C.14: *Average reset cycles needed for a successful Pause and Resume attack on* `CIFAR10`*. The fastest attack (HSJA) can succeed in roughly 3 years.*