WASHINGTON UNIVERSITY IN ST.LOUIS

McKelvey School of Engineering
Department of Computer Science and Engineering

Dissertation Examination Committee:
Yevgeniy Vorobeychik, Chair
Ayan Chakrabarti
Sanmay Das
Bruno Sinopoli
Ning Zhang

Towards Deploying Robust Machine Learning Systems

by
Liang Tong

A dissertation presented to
The Graduate School
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

May 2021
St. Louis, Missouri

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I am extremely grateful to my advisor, Prof. Yevgeniy Vorobeychik, for his invaluable advice, patience, and continuous support during my PhD study. His immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. The work presented in this dissertation would not have been possible without him. I wish I could be mentored by him a little longer, but it is time to begin my new adventure and to become an independent researcher.

I also want to thank my committee members, Professor Bruno Sinopoli, Professor Ning Zhang, Professor Ayan Chakrabarti, and Professor Sanmay Das. Without fruitful discussions with them, I will not be able to develop all these wonderful ideas and solve challenging problems.

I want to express my appreciation to NEC Labs America for their fantastic research intern program. Special thanks go to Dr. Zhengzhang Chen and Dr. Haifeng Chen, who were my mentors during my summer internship in 2021, for offering me countless support on my research projects.

I would also like to thank all my labmates and friends, Bo Li, Haifeng Zhang, Sweta Panda, Jian Lou, Ayan Mukhopadhyay, Sixie Yu, Kai Zhou, Chen Hajaj, Jinghan Yang, Rajagopal Venkatesaramani, Tong Wu, Minzhe Guo, Joss Wang, Andrew Estornell, and Chao Yan, for their generous support and encouragement.

Finally, I would like to thank my family for their unconditional support and love during these years, particularly to my wife, Pei Tung, who has been a great companion, and who has been with me through the highs and the lows in the past years.

Liang Tong

Washington University in Saint Louis

May 2021

I dedicate this work to Lord, my wife, and my parents.

ABSTRACT OF THE DISSERTATION

Towards Deploying Robust Machine Learning Systems

by

Liang Tong

Doctor of Philosophy in Computer Science

Washington University in St. Louis, 2021

Professor Yevgeniy Vorobeychik, Chair

Machine learning (ML) has come to be widely used in a broad array of settings, including important security applications such as network intrusion, fraud, and malware detection, as well as other high-stakes settings, such as autonomous driving. A general approach is to extract a set of features, or numerical attributes, of entities in question, collect a training data set of labeled examples (for example, indicating which instances are malicious and which are benign), learn a model which labels previously unseen instances presented in terms of their extracted features, and then investigate alerts raised by instances predicted as malicious. Despite the striking success of ML in security applications, security issues emerge from the full pipeline of ML-based detection systems. First, ML models are often susceptible to adversarial examples, in which an adversary makes changes to the input (such as malware) to avoid being detected. Second, using detection systems in practice is dealing with an overwhelming number of alerts that are triggered by normal behavior (the so-called false positives), obscuring alerts resulting from actual malicious activities. Third, adversaries can target a broad array of ML-based detection systems to maximize impact, which is often ignored by individual ML system designers.

In this thesis, I focus on studying the security problems of deploying robust machine learning systems in adversarial settings. To conduct systematic research on this topic, my study

is based on four components. First, I study the problem of systematizing adversarial evaluation. Concretely, I propose a fine-grained robustness evaluation framework for face recognition systems. Second, I investigate robust machine learning against decision-time attacks. Specifically, I propose a framework for validating models of ML evasion attacks, and evaluate the efficacy of conventional robust machine learning models against realizable attacks in PDF malware detection. My work shows that the key to robustness is the conserved features, and I propose a systematic algorithm to identify these. Additionally, I study robustness against non-salient adversarial examples in image classification and propose cognitive modeling of suspiciousness of adversarial examples. Third, I study the robust alert prioritization problem—often a necessary step in the detection pipeline. I propose a novel approach for computing a policy for prioritizing alerts using adversarial reinforcement learning. Last, I investigate robust decentralized learning, and I develop a game-theoretic model for robust linear regression involving multiple learners and a single adversary.

Chapter 1

Introduction

1.1 Motivation and Challenges

Machine learning (ML) has come to be widely used in a broad array of settings, including important security applications such as network intrusion, fraud, and malware detection, as well as other high-stakes settings, such as autonomous driving. A general approach is to extract a set of features, or numerical attributes, of entities in question, collect a training data set of labeled examples (for example, indicating which instances are malicious and which are benign), learn a model which labels previously unseen instances presented in terms of their extracted features, and then investigate alerts raised by instances predicted as malicious. Success of ML is particularly striking: in malware detection, ML-based static detection of malicious entities can achieve 99% accuracy [102, 103] while in traffic sign classification the accuracy exceeds 91% [25].

Nevertheless, the learning systems can potentially be subverted by adversarial manipulations, which exposes applications that use machine learning techniques to a new class of vulnerabilities. This challenge, in turn, motivates our proposed research of designing robust machine learning systems in practice. We elaborate on the challenging issues from the following three aspects.

Vulnerability to Adversarial Examples. Recent research has shown that machine learning approaches, and especially classifier learning, are susceptible to *adversarial examples*. That is, a classifier can be fooled by adding small perturbations to the original examples. A fundamental reason for such vulnerabilities is that classification learning algorithms generally assume that the distribution of training and test (or production) data is similar. This assumption is violated in security applications, where malicious entities correspond to attackers who can, and do, take deliberate action to evade defensive measures. For example, in the case of malware detection, an adversary can modify malware code so that the resulting malware is categorized as benign by ML, but still successfully executes the malicious payload [103, 126]. An even a broader class of adversarial examples features attacks that manipulate an object, such as a human face, so that face recognition pipeline misclassifies he/she as another person [97].

Overwhelming Number of Alerts. One of the core problems in security is *detection* of malicious behavior, with examples including detection of malicious software, emails, websites, and network traffic. There is a vast literature on machine-learning based detection approaches [11, 76, 101]. Despite best efforts, however, false positives are inevitable. Moreover, one cannot, in general, reduce the rate of false alarms without missing some real attacks as a result. Under the pressure of practical considerations such as liability and accountability, these systems are often configured to produce a large number of alerts in order to be sufficiently sensitive to capture most attacks. As a consequence, cybersecurity professionals are routinely

inundated with alerts and must sift through these overwhelmingly uninteresting logs to identify alerts that should be prioritized for closer inspection.

A considerable literature has emerged attempting to reduce the number of false alerts without significantly affecting the ability to detect malicious behavior [42, 48, 93]. Most of these attempt to add meta-reasoning on top of detection systems that capture broader system state, combining related alerts, escalating priority based on correlated observations, or using alert correlation to dismiss false alarms [114]. Nevertheless, despite significant advances, there are typically still vastly more alerts than time to investigate them. With this state of affairs, alert prioritization approaches have emerged, but rely predominantly on predefined heuristics, such as sorting alerts by suspiciousness score or by potential associated risk [2]. However, any policy that *deterministically* orders alerts potentially opens the door for determined attackers who can simply choose attacks that are rarely investigated, thereby evading detection. Therefore, how to balance the fundamental trade-off between false alert and attack detection rate forms another obstacle for robust machine learning systems deployed in practice.

Decentralization of ML Systems. Increasing use of machine learning in adversarial settings has motivated a series of efforts investigating the extent to which learning approaches can be subverted by malicious parties. An important class of such attacks involves adversaries changing their behaviors, or features of the environment, to effect an incorrect prediction. Most previous efforts study this problem as an interaction between a single learner and a single attacker [10, 20, 59, 132]. However, in reality attackers often target a broad array of potential victim organizations. For example, they craft generic spam templates and generic malware, and then disseminate these widely to maximize impact. The resulting ecology of attack targets reflects not a single learner, but many such learners, all making autonomous decisions about how to detect malicious content, although these decisions often rely on similar

training datasets. However, such setting is typically ignored when ML systems are deployed, which forms the third challenge towards robust machine learning system in practice.

1.2 Overview of The Thesis

In response to the challenges described above, I make several contributions to *designing robust machine learning systems under adversarial environments* in this thesis. My contributions fall into four directions: *systematizing adversarial evaluation* (Chapter 3), *robust machine learning against adversarial examples* (Chapter 4, 5, and 6), *robust alert prioritization* (Chapter 7), and *robust decentralized learning* (Chapter 8).

The remainder of this thesis is organized as follows. In Chapter 2, I present the background knowledge and related work of this thesis. Chapter 3 presents a framework for fine-grained robustness evaluation of face recognition systems, which enables to assess different levels of robustness under various adversarial circumstances. Chapter 4, 5, and 6 focus on adversarial defense against decision-time attacks. Specifically, I first present a general framework for validating the efficacy of conventional robust ML against real attacks in Chapter 4, and show that robust ML systems can fail to defend against realizable attacks in the context of PDF malware detection. In Chapter 5, I present a refinement of robust ML by utilizing conserved features and show that augmenting robust ML with such features can significantly improve performance. Afterward, I investigate robustness against non-salient adversarial examples in image classification in Chapter 6, and propose a simple formalization of an important aspect of what makes adversarial perturbations unsuspicious based on the notion of cognitive salience. Chapter 7 focuses on deciding which of a large number of alerts to choose for further investigation—often a necessary step in the detection pipeline. In this chapter, I present a novel game-theoretic model and principled computational approach for

robust alert prioritization. Chapter 8 investigates robust decentralized learning, in which I present a game-theoretic approach for adversarial regression involving multiple learners and a single attacker. Chapter 9 concludes and discusses potential future directions.

Chapter 2

Background and Related Work

2.1 Machine Learning in Security

2.1.1 Malware Detection

In the (supervised) machine learning literature, it is common to consider the problem abstractly. We are given a training dataset $\mathsf{D} = \{(\boldsymbol{x}_i, y_i)\}$, where $\boldsymbol{x}_i \in \mathsf{X} \subseteq \mathbb{R}^n$ are numeric feature vectors in some feature space X and $y_i \in \mathsf{L}$ are labels in a label space L. Each data point (or example) in D is assumed to be generated i.i.d. according to some unknown distribution P. We are also given a hypothesis (model) space, H, and our goal is to identify (*learn*) a good model $h_{\boldsymbol{\theta}} \in \mathsf{H}$ parameterized by $\boldsymbol{\theta}$ in the sense that it yields a small expected error on new examples drawn from P. In practice, since P is unknown, one typically aims to find $h_{\boldsymbol{\theta}} \in \mathsf{H}$ which (approximately) minimizes empirical error on training data D.

In malware detection—as in others—one is not given numerical features; instead, we start with a collection of entities, such as executables, along with associated labels (we assume

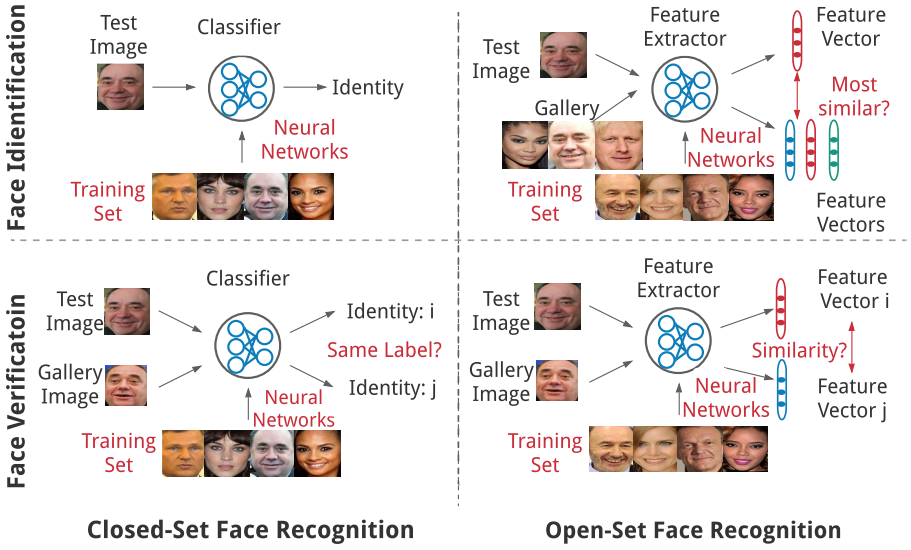

Figure 2.1: Closed-set and open-set face recognition systems.

henceforth that these are available, as we focus here on supervised learning problems). We must then *design a collection of feature extractors*, where each feature extractor computes a numerical value of a corresponding feature from an input entity. For example, we extract a "size" feature by computing the size of an executable. Applying feature extractors to each entity in our dataset, and adding associated object labels, allow us to generate a dataset D to fit the conventional ML framework.

Generally, the label space is binary: either a file is benign (which we can code as -1), or malicious (which we can code as $+1$). In addition, several prior efforts presented techniques for defining *feature extractors* (commonly known simply as features) for PDF malware detection [102, 103]. Applying such feature extractors to a PDF file dataset transforms this dataset into one comprised of numerical feature vectors and associated binary labels. The goal is to predict whether previously unseen PDFs (simulated by holding out a portion of our dataset as *test data*) are correctly labeled as malicious or benign.

2.1.2 Face Recognition

In computer vision tasks such as face recognition, the raw feature extractor can be a camera that captures entities and translate these into images with pixels in the digital space. The images are subsequently fed into deep convolutional neural networks (CNNs) to extract higher level features such as shapes and edges. Generally, deep face recognition systems aim to solve the following two tasks: 1) *Face identification*, which returns the predicted identity of a test face image; 2) *Face verification*, which indicates whether a test face image (also called probe face image) and the face image stored in the gallery belong to the same identity. Based on whether all testing identities are predefined in the training set, face recognition systems can be further categorized into *closed-set systems* and *open-set systems* [65], as illustrated in Fig. 2.1.

In closed-set face recognition tasks, all the testing samples' identities are enrolled in the training set. Specifically, a face identification task is equivalent to a *multi-class classification* problem by using the standard softmax loss function in the training phase [105, 106, 109]. And a face verification task is a natural extension of face identification by first performing the classification twice (one for the test image and the other for the gallery) and then comparing the predicted identities to see if they are identical.

In contrast, there are usually no overlaps between identities in the training and testing set for open-set tasks. In this setting, a face verification task is essentially a *metric learning* problem, which aims to maximize *intra-class distance* and minimize *inter-class distance* under a chosen metric space by two steps [22, 65, 66, 88, 95, 120]. First, we train a feature extractor that maps a face image into a discriminative feature space by using a carefully designed loss function; Then, we measure the distance between feature vectors of the test and gallery face images to see if it is above a verification threshold. As an extension of face verification,

the face identification task requires additional steps to compare the distances between the feature vectors of the test image and each gallery image, and then choose the gallery's identity corresponding to the shortest distance.

2.2 Decision-Time Attacks on Machine Learning Systems

Recent studies have shown that ML-based techniques are often susceptible to *adversarial examples*, in which an adversary makes changes to the input of a machine learning model in order to cause an incorrect prediction at the *decision time*. Based on the space in which such attacks are performed, decision-time attacks can be further categorized into *realizable attacks* and *feature-space attacks*, as detailed below.

2.2.1 Realizable Attacks

Realizable attacks are attacks that entail modifying the actual object in order to fool a machine learning model that subsequently takes its digital representation as input. An early realizable attack on machine learning was devised by Fogla et al. [27, 28], who developed an attack on anomaly-based intrusion detection systems. Šrndic and Laskov [103] present a case study of an evasion attack on a state-of-the-art PDF malware classifier, PDFRate. Xu et al. [126] propose EvadeML, a fully realizable attack on PDF malware classifiers which generates evasion instances by using genetic programming to modify PDF source directly, using a sandbox to ensure that malicious functionality is preserved. Grosse et al. [34] develop a method for generating evasion attacks against a deep learning-based Android malware classifier, using a gradient-based approach which is also a form of iterative improvement heuristics. This particular attack can be viewed as realizable, even though it wasn't implemented and evaluated in actual malware, since the attack space is significantly restricted to only add

features that do not interfere with others already present. Similarly, MalGAN, an evasion attack based on generative adversarial networks developed by Hu and Tan [46], only adds features from benign to malicious malware, and we treat it as a realizable attack (since it's not difficult to implement). Additionally, several recent approaches attempt to generate adversarial examples against computer vision systems in physical space, such as adding stickers to a stop sign to cause misclassification [25], or wearing printed glass frames to fool face recognition [97].

Next, we introduce realizable attacks on two representative domains, PDF malware detection and face recognition, respectively.

Realizable Attacks on PDF Malware Detection. In PDF malware detection, abstractly, one is given a learned model $h_{\boldsymbol{\theta}}(\boldsymbol{x})$ (e.g., a SVM or neural network) which returns a label $y = h_{\boldsymbol{\theta}}(\boldsymbol{x})$ (e.g., malicious or benign) for an arbitrary feature vector $\boldsymbol{x} \in \mathsf{X}$ (e.g., extracted from a PDF file). The attacker additionally starts with an entity $\boldsymbol{e}$ (such as a malicious PDF file), from which we can extract a feature vector $\phi(\boldsymbol{e})$. The attacker then transforms $\boldsymbol{e}$ into another entity, $\boldsymbol{e}'$, with an associated feature vector $\boldsymbol{x}' = \phi(\boldsymbol{e}')$ so as to accomplish two goals: first, that $h_{\boldsymbol{\theta}}(\boldsymbol{x}')$ returns an erroneous label (for example in PDF malware detection, labels $\boldsymbol{e}'$ as benign based on its extracted features $\phi(\boldsymbol{e}')$), and second, that $\boldsymbol{e}'$ preserves the functionality of the original entity $\boldsymbol{e}$—which, in our example of PDF malware detection, entails preserving malicious functionality of $\boldsymbol{e}$. The realizable attack as just described is presumed to transform the *entity itself*, such as the malicious PDF file, albeit accounting for the effect of such transformation on the extracted features $\boldsymbol{x}' = \phi(\boldsymbol{e}')$. The process by which such realizable attacks can be successfully accomplished is quite non-trivial, and typically warrants independent research contributions (e.g., [126]).

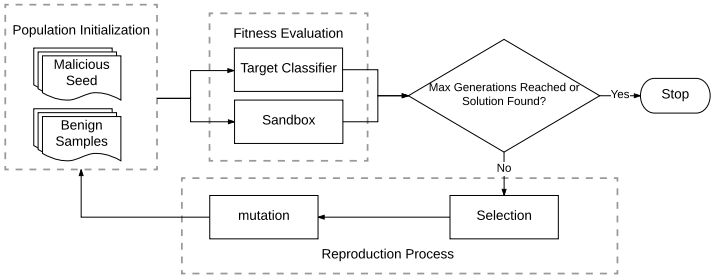

Figure 2.2: EvadeML [126], a realizable attack on PDF Malware Classifiers.

Figure 2.2 illustrate EvadeML [126], a realizable attack on PDF malware detectors which allows insertion, deletion, and swapping of objects, and is consequently a stronger attack than most other realizable attacks in the literature, which typically only allow insertion to ensure that malicious functionality is preserved. EvadeML assumes that the adversary has black-box access to the classifier and can only get classification scores of PDF files, and was shown to effectively evade state-of-the-art PDF malware detectors. It employs genetic programming (GP) to search the space of possible PDF instances to find ones that evade the classifier while maintaining malicious features.

Realizable Attacks on Face Recognition. In the context of face recognition, of particular interest are attacks in the physical world (henceforth, *physical attacks*). Generally, physical attacks have three characteristics [122]. First, the attackers directly modify the actual entity rather than digital features. Second, the attacks can mislead state-of-the-art face recognition systems. Third, the attacks have low suspiciousness (*i.e.*, by adding objects similar to common "noise" on a small part of human faces). For example, an attacker can fool a face recognition system by wearing an adversarial eyeglass frame [97], a standard face accessory in the real world.

In this dissertation, we focus on *the digital representation of physical attacks* (henceforth, *physically realizable attacks*). Specifically, physically realizable attacks are digital attacks that can produce adversarial perturbations with low suspiciousness, and these perturbations

Figure 2.3: Sticker attack: an example of physically realizable attacks on face recognition systems. Left: original input image. Middle: adversarial sticker on the face. Right: predicted identity. In practice, the adversarial stickers can be printed and put on human faces.

can be realized in the physical world by using techniques such as 3-D printing (*e.g.*, Fig. 2.3 illustrates one example of such attacks on face recognition systems). Compared to physical attacks, physically realizable attacks can evaluate robustness of face recognition systems more efficiently: on the one hand, realizable attacks allow us to iteratively modify digital images directly so the evaluation can significantly speedup compared to modifying real-world objects and then photographing them; on the other hand, robustness to physically realizable attacks provides the lower bound of robustness to physical attacks, as the former has fewer constraints and larger solution space.

Formally, physically realizable attacks on face recognition can be performed by solving the following general form of an optimization problem (*e.g.*, for closed-set identification task as described above):

$$\arg\max_{\boldsymbol{\delta}} \ell(h_{\boldsymbol{\theta}}(\boldsymbol{x} + M\boldsymbol{\delta}), y) \qquad s.t.\ \boldsymbol{\delta} \in \Delta, \tag{2.1}$$

where h is the target face recognition model parameterized by $\boldsymbol{\theta}$, ℓ is the adversary's utility function (*e.g.*, the loss function used to train h), $\boldsymbol{x}$ is the original input face image, y is the associated identity, $\boldsymbol{\delta}$ is the adversarial perturbation, and Δ is the feasible space of the perturbation. Here, M denotes the mask matrix that constrains the area of perturbation; it has the same dimension as $\boldsymbol{\delta}$ and contains 1s where perturbation is allowed, and 0s where there is no perturbation.

2.2.2 Feature-Space Attack Models

As implementing realiazable attacks requires domain knowledge and considerable amount of engineering work, it is natural to short-circuit the complexity involved, and work directly in the *feature space*, as is conventional in the machine learning literature [4, 7, 13, 20, 33, 69, 130]. Moreover, a series of efforts explore attacks in the context of image classification by deep neural networks [13, 33, 82, 85, 108]. These approaches commonly generate adversarial perturbations within a bounded ℓ_p norm so that the perturbations are imperceptible, although Gilmer et al. [30] question the common threat models used in these works.

In this case, the attacker is *modeled* as starting with a malicious feature vector $\boldsymbol{x}$ (*not the malicious entity* $\boldsymbol{e}$), and *directly modifying the features* to produce another feature vector $\boldsymbol{x}' \in \mathsf{X}$, so as to yield erroneous predictions, i.e., $y' = h_{\boldsymbol{\theta}}(\boldsymbol{x}')$ (for example, being mislabeled as benign). Crucially, since we are no longer appealing to original entities, we must abstract away the notion of preserving (malicious) functionality. This is done through the use of a cost function, $c(\boldsymbol{x}, \boldsymbol{x}')$, whereby the attacker is penalized for greater modifications to the given feature vector $\boldsymbol{x}$, commonly measured using an ℓ_p norm difference between the original malicious instance and the modified feature vector [7, 58]. We term these the *feature-space attack models*.

The problem of identifying an adversarial example in feature space for $\boldsymbol{x}$ can be captured by the following optimization problem (or its variants):

$$\max_{\boldsymbol{\delta} \in \Delta(\epsilon)} \mathcal{L}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x} + \boldsymbol{\delta}), y\right) \tag{2.2}$$

where $\mathcal{L}(\cdot)$ is the loss function used to train the classifier $h_{\boldsymbol{\theta}}$ and $\Delta(\epsilon)$ is the feasible perturbation space which is commonly represented as a ℓ_p ball: $\Delta(\epsilon) = \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_p \leq \epsilon\}$. A

number of approaches have been proposed to solve the optimization problem above. For classifiers with real-valued features, we can use gradient based approaches [7, 71][1]. When the features are binary, the optimization problem in Eq. (2.2) can be solved by using *Corrdinate Greedy* (alternatively known as iterative improvement) [58] which optimizes one randomly chosen coordinate of the feature vector at a time, until a local optimum is reached. To improve the quality of the resulting solution (considering that $\mathcal{L}(\cdot)$ is typically non-concave), the above process can be repated from several random starting points [58, 71].

2.3 Robust Learning

A large number of approaches have been proposed for defending against adversarial examples (e.g., [7, 9, 10, 86, 87, 91, 115, 121, 124]). While many have been shown inadequate [4, 13], the four generally effective approaches are: (a) game-theoretic reasoning, (b) robust optimization (a special case of (a) where the game is zero-sum), (c) adversarial training (an approach for obtaining approximate (a) or (b) solutions [58, 71, 115]), and (d) randomized smoothing [16, 57]. Game-theoretic methods in general, and robust optimization in particular, are not general-purpose, as solving these directly requires special structure, such as a continuous feature space and differentiability [7, 9, 10], and often additional structure of the learning model, such as linearity [124] or neural network architecture and activation functions [91, 121]. Finally, to date *all have used the mathematical feature-space attack model at their core*.

Next, we describe two categorizations of defense that have proved both sufficiently scalable and effective even against adaptive attacks: *adversarial training* [16, 33, 71, 108] and *randomized smoothing* [16, 57].

[1]Generally, images are preprocessed such that pixels are divided by 255 for computational convenience in training and testing. Consequently, a feasible pixel value should lie in [0,1] and is treated as real-valued.

2.3.1 Adversarial Training

The basic idea of adversarial training is to produce adversarial examples and incorporate these into the training process. Formally, adversarial training aims to solve the following robust learning problem:

$$\min_{\boldsymbol{\theta}} \frac{1}{|\mathsf{D}|} \sum_{\boldsymbol{x},y \in D} \max_{\|\boldsymbol{\delta}\|_p \leq \epsilon} \mathcal{L}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta}), y\right) \tag{2.3}$$

where D is the training dataset. In practice, this problem is commonly solved by iteratively using the following two steps [71]: 1) use any attack to produce adversarial examples in feature space of the training data; 2) use any optimizer to minimize the loss of those adversarial examples. It has been shown that adversarial training can significantly boost the adversarial robustness of a classifier against ℓ_p attacks, and it can be scaled to neural networks with complex architectures. Note that adversarial training is a general approach that can be also incorporated with realizable attacks. That is, we can first use realizable attacks to produce adversarial entities, then translate these into feature space, add them to the training data, and re-train the classifier.

2.3.2 Randomized Smoothing

The second class of methods for robust learning considers adding random perturbations to inputs at both training and test time. The basic idea is to construct a new smoothed classifier $g_{\boldsymbol{\theta}}(\cdot)$ from a base classifier $h_{\boldsymbol{\theta}}(\cdot)$ as follows: first, the base classifier $h_{\boldsymbol{\theta}}(\cdot)$ is trained with *Gaussian data augmentation* with variance $\boldsymbol{\sigma}^2$; then, for any input $\boldsymbol{x}$ at test time, the smoothed classifier $g_{\boldsymbol{\theta}}(\cdot)$ returns the class that has the highest probability measure for the

base classifier $h_{\boldsymbol{\theta}}(\cdot)$ when inputs are perturbed with isotropic Gaussian noise:

$$\begin{aligned} g_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \arg\max_{c} P(h_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\eta}) = c) \\ \text{where } \boldsymbol{\eta} &\sim \mathcal{N}\left(\mathbf{0}, \sigma^2 \boldsymbol{I}\right) \end{aligned} . \tag{2.4}$$

It has been shown that randomized smoothing can provide *certified robustness* to adversarial perturbations for ℓ_2-norm-bounded attacks [16, 57].

2.4 Reinforcement Learning

2.4.1 Deep Reinforcement Learning

Reinforcement learning has received significant attention in recent years, which is in large part due to the emergence of deep reinforcement learning. Deep reinforcement learning combines classic reinforcement learning approaches, such as *Q-learning* [119], with deep neural networks. Classic Q-learning is a model-free reinforcement learning approach, which is guaranteed to find an optimal policy for any finite Markov decision process [118]. However, to do so, it needs to learn and store an exact representation of the action-value function, which is infeasible for a problem with large action or state spaces. Notable early successes combining reinforcement learning with neural networks include *TD-Gammon*, a backgammon program that achieved a level of play that was comparable to top human players in 1992 [110]. More recently, Mnih et al. introduced the model-free *Deep Q-Learning algorithm* (DQN), which achieved human-level performance in playing a number of Atari video games, using purely visual input from the games [78, 79]. However, the actions spaces in all of these games were small and discrete. Lillicrap et al. adapted the idea of Deep Q-Learning to continuous action spaces by introducing an algorithm, called *Deep Deterministic Policy Gradient* (DDPG) [61].

DDPG is a model-free actor-critic algorithm, whose robustness is demonstrated on a variety of continuous control tasks. [117], A3C [77], Distributional DQN [5], and Noisy DQN [29]), which had been proposed by the deep reinforcement learning community since the publication of DQN, across 57 Atari games [40]. Further, they integrated these improvements into a single agent, called *Rainbow*, and demonstrated its state-of-the-art performance on common benchmarks.

2.4.2 Multi-Agent Reinforcement Learning

Single-agent reinforcement learning approaches can train only one agent at a time, which means that in a multi-agent setting, they must treat other agents as part of the environment. As a result, they often provide policies that are not robust—especially in a non-cooperative setting such as ours—since they cannot consider the possibility that other agents respond by learning and updating their own policies. Multi-agent reinforcement learning approaches attempt to provide more robust policies by training multiple adaptive agents together.

Littman proposed a framework for multi-agent reinforcement learning that models the competition between two agents as a zero-sum Markov game [64]. To solve this game, the author introduced a Q-learning-like algorithm, called *minimax-Q*, which is guaranteed to converge to optimal policies for both players. However, the minimax-Q algorithm assumes that the game is zero-sum (i.e., the player's rewards are exact opposites of each other) and every step of the training involves exhaustive searches over the action spaces, which limits the applicability of the algorithm. A number of follow up efforts have proposed more general solutions. For example, Hu and Wellman extended Littman's framework to general-sum stochastic games [44]. They propose an algorithm that is based on each agent learning two action-value functions (one for itself and one for its opponent), which is guaranteed to converge to a Nash equilibrium under certain conditions. To relax some of these conditions,

Littman introduced *Friend-or-Foe Q-learning*, in which each agent is told to treat each other agent either as a "friend" or as a "foe" [63]. Later, Hu and Wellman proposed the *NashQ algorithm*, which generalizes single-agent Q-learning to stochastic games with many agents by using an equilibrium operator instead of expected utility maximization [45].

While the above approaches have the advantage of providing certain convergence guarantees, they assume that action-value functions are represented exactly, which is infeasible for scenarios with large action or state spaces. Deep multi-agent reinforcement-learning provides a more scalable approach by representing action-value functions using deep neural networks. For example, Lowe et al. proposed an adaptation of actor-critic reinforcement-learning methods to multi-agent settings [70]. In the proposed approach, each agent learns a collection of different sub-policies, and for each episode, each agent randomly selects sub-policy from this collection. Lanctot et al. introduced an algorithm, called *policy-space response oracles*, which maintains a set of policies for each agent, but it does not incorporate actor-critic methods, and it was evaluated in settings with relatively small discrete action spaces.

2.5 Alert Management and Prioritization

A multitude of research efforts have studied the problem of reducing the number of alerts without significantly reducing the probability of attack detection [48]. One of the most common approaches is *alert correlation* and *clustering*, which attempt to group related alerts together, thereby reducing the set of messages that are presented [93]. In distributed systems, *collaborative intrusion detection systems* may be deployed, which include several monitoring components and correlate alerts among the monitors to create a holistic view [114]. Since the number of alerts may be too high even after correlation, research efforts have also investigated the prioritization of alerts. For example, Alsubhi et al. introduced a fuzzy-logic based alert

management system, called *FuzMet*, which uses several metrics and fuzzy logic to score and prioritize alerts [2]. However, these approaches do not consider the possibility of an attacker adapting to the prioritization.

Prior work has successfully applied game theory to a variety of security problems, ranging from physical security [3] to network security and privacy [73]. Our proposed work is most closely related to *alert-prioritization games*. Laszka et al. introduced *GAIN*, the first game-theoretic model for alert prioritization, which they solved with the help of a greedy heuristic [56]. The performance of this approach is limited by its restrictive assumptions about the defender's decision making. In particular, this approach assumes that the defender's policy is a strict prioritization that investigates all higher-priority alerts before investigating any lower-priority ones, and the prioritization is chosen before observing the actual number of alerts. Moreover, the model considers only a single time slot, which further limits its usefulness. Yan et al. improved upon GAIN by allowing the defender to specify a maximum budget that may be spent on each alert types, thereby relaxing the strict prioritization of GAIN [127]. However, this improved approach, which we denoted RIO in our experiments, still assumes that the prioritization is chosen before observing any alerts and considers only a single time slot. Schlenker et al. introduced a similar model, called *Cyber-alert Allocation Game*, which further simplifies the problem by assuming that the number of false alerts is fixed and known by both parties in advance [94].

Part I

Systematizing Adversarial Evaluation of Machine Learning Systems

Chapter 3

Fine-Grained Robustness Evaluation for Face Recognition Systems

The previous chapter has shown that machine learning techniques are often susceptible to adversarial examples in different domains and settings. Therefore, there are pressing needs for methods to systematically and comprehensively evaluate robustness of machine learning systems in adversarial settings, which in turn provide insights for designing robust machine learning models. In this chapter, we present FaceSec, a framework for fine-grained robustness evaluation of face recognition systems. FaceSec evaluation is performed along four dimensions of adversarial modeling: the nature of perturbation (*e.g.*, pixel-level or face accessories), the attacker's system knowledge (about training data and learning architecture), goals (dodging or impersonation), and capability (tailored to individual inputs or across sets of these). We use FaceSec to study five face recognition systems in both closed-set and open-set settings, and to evaluate the state-of-the-art approach for defending against physically realizable attacks on these. We find that accurate knowledge of neural architecture is significantly more important than knowledge of the training data in black-box attacks.

Moreover, we observe that open-set face recognition systems are more vulnerable than closed-set systems under different types of attacks. The efficacy of attacks for other threat model variations, however, appears highly dependent on both the nature of perturbation and the neural network architecture. For example, attacks that involve adversarial face masks are usually more potent, even against adversarially trained models, and the ArcFace architecture tends to be more robust than the others.

3.1 Overview

Face recognition has received much attention [50, 65, 88, 95, 116, 120] in recent years. Empowered by deep convolutional neural networks (CNNs), it has become widely used in various areas, including security-sensitive applications, such as airport check-in, online financial transactions, and mobile device login.

Despite its widespread success in computer vision applications, recent studies have found that deep face recognition models are vulnerable to *adversarial examples* in both *digital space* [24, 71, 128] and *physical space* [97]. The former directly modifies an input face image by adding imperceptible perturbations to mislead face recognition (henceforth, *digital attacks*). The latter is characterized by adding adversarial perturbations that can be realized on *physical objects* (*e.g.*, wearing an adversarial eyeglass frame [97]), which are subsequently captured by a camera and then fed into a face recognition model to fool prediction (henceforth, *physically realizable attacks*). As such, the aforementioned domains, especially critical domains such as security or finance, are subjected to risks of opening the backdoor for the attackers. For example, in face recognition supported financial/banking services, an illegal user may bypass biometric verification and steal money from victims' accounts. Therefore, there exists a vital need for methods that can comprehensively and systematically evaluate the robustness of

face recognition systems in adversarial settings, which in turn can shed light on the design of robust models for downstream face recognition tasks.

The main challenges of comprehensive evaluation of the robustness of face recognition lie in dealing with the diversity of face recognition systems and adversarial environments. First, different face recognition systems consist of various key components (*e.g.*, training data and neural architecture); such diversity results in different performance and robustness. To enable comprehensive and systematic evaluations, it is crucial to assess the robustness of every individual or a combination of face recognition components in adversarial settings. Second, adversarial example attacks can vary by the nature of perturbations (*e.g.*, pixel-level or physical space), an attacker's goal, knowledge, and capability. For a given face recognition system, its robustness against a specific type of attack may not generalize to other kinds [122].

In spite of recent advances in adversarial attacks [24, 97, 128] that demonstrate the vulnerability of face recognition systems, most existing methods fail to address the aforementioned challenges due to the following reasons. First, current efforts appeal to either *white-box attacks* or *black-box attacks* to obtain a lower bound or upper bound of robustness. These bounds indicate the vulnerability of face recognition systems in adversarial settings but lack the understanding of how each component of face recognition contributes to such vulnerability. Second, while most existing approaches focus on a specific type of attack (*e.g.*, digital attacks that incur imperceptible noise [24, 128]), they fail to explore the different levels of robustness in response to various attacks (*e.g.*, physically realizable attacks).

To bridge this gap, we propose FaceSec, a fine-grained robustness evaluation framework for face recognition systems. FaceSec incorporates four dimensions in evaluation: the nature of adversarial perturbations (pixel-level or face accessories), the attacker's accurate knowledge about the target face recognition system (training data and neural architecture), goals

(dodging or impersonation), and capability (individual or universal attacks). Specifically, we implement both digital and physically realizable attacks in FaceSec. We leverage the PGD attack [71], the state-of-the-art digital attack paradigm, and the eyeglass frame attack [97] as the representative of physically realizable attacks. Additionally, we propose two novel physically realizable attacks: one involves pixel-level adversarial stickers on human faces, and the other adds color grids on face masks. Moreover, to facilitate universal attacks that produce *image-agnostic* perturbations, we propose a systematic approach that works on top of the attack paradigms described above. We perform a comprehensive evaluation on five publicly available face recognition systems in various settings to demonstrate the efficacy of FaceSec.

3.2 Methodology

In this section, we introduce FaceSec for fine-grained robustness evaluation of face recognition systems. Our goal is twofold: 1) identify vulnerability/robustness of each essential component that comprises a face recognition system, and 2) assess robustness in a variety of adversarial settings. Fig. 3.1 illustrates an overview of FaceSec. Let $S = f(h; D)$ be a face recognition system with a neural architecture h that is trained on a training set D by an algorithm f (*e.g.*, stochastic gradient descent), FaceSec evaluates the robustness of S via a quadruplet:

$$\mathsf{Robustness} = \mathsf{Evaluate}(S, < P, K, G, C >), \tag{3.1}$$

where $< P, K, G, C >$ represents an attacker who tries to produce adversarial examples to fool S. P is the perturbation type, such as perturbations produced by pixel-level digital attacks and physically realizable attacks. K denotes the attacker's knowledge on the target system S, *i.e.*, the information about which sub-components of S are leaked to the attacker.

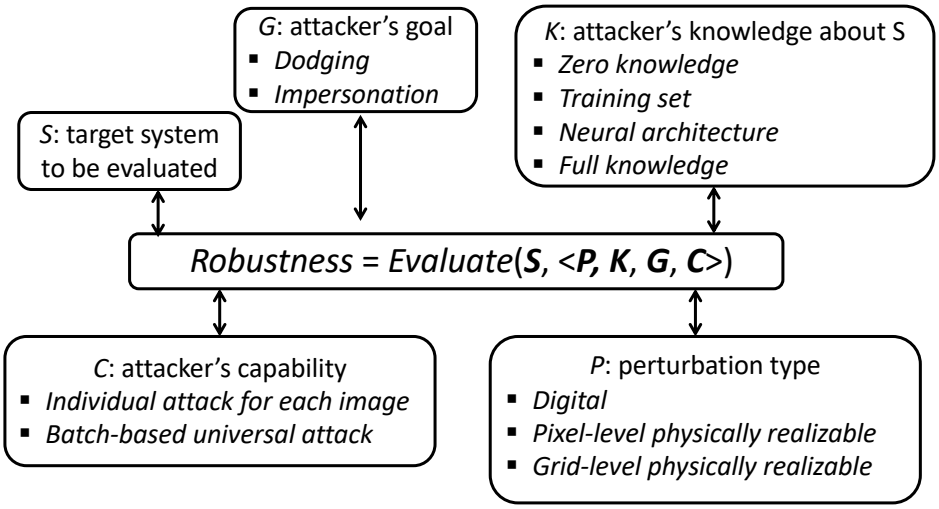

Figure 3.1: An overview of FaceSec.

G is the goal of the attacker, such as the circumvention of detection and the misrecognition as a target identity. C represents the attacker's capability. For example, an attacker can either individually perturb each input face image, or produce universal perturbations for images batch-wise. Next, we will describe each element of FaceSec in details.

3.2.1 Perturbation Type (P)

In FaceSec, we consider three categories of attacks with different perturbation types: *digital attack*, *pixel-level physically realizable attack*, and *grid-level physically realizable attack*, as shown in Fig. 3.2.

Digital Attack. Digital attack produces small perturbations on the entire input face image. We use the ℓ_∞-norm version of the PGD attack [71] as the representative of this category.

Pixel-level Physically Realizable Attack. This category of attack features pixel-level perturbations that can be realized in the physical world (*e.g.*, by printing them on glossy photo papers). In this case, the attacker adds large pixel-level perturbations on a small area

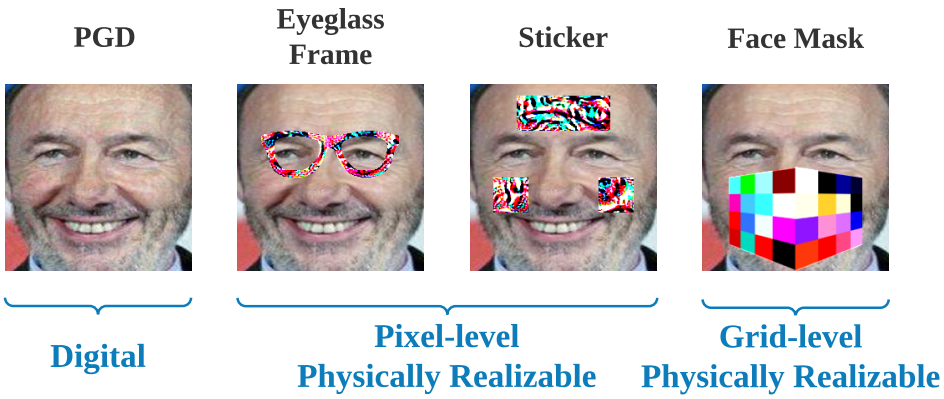

Figure 3.2: Perturbation types in FaceSec.

Table 3.1: Optimization formulations of grid-level face mask attacks.

Target System	Attacker's Goal	Formulation
Closed-set	Dodging	$\max_{\boldsymbol{\delta}} \ell(S(\boldsymbol{x} + M \cdot \mathcal{T}(\boldsymbol{\delta})), y)$
Closed-set	Impersonation	$\min_{\boldsymbol{\delta}} \ell(S(\boldsymbol{x} + M \cdot \mathcal{T}(\boldsymbol{\delta})), y_t)$
Open-set	Dodging	$\max_{\boldsymbol{\delta}} d(S(\boldsymbol{x} + M \cdot \mathcal{T}(\boldsymbol{\delta})), S(\boldsymbol{x}^*))$
Open-set	Impersonation	$\min_{\boldsymbol{\delta}} d(S(\boldsymbol{x} + M \cdot \mathcal{T}(\boldsymbol{\delta})), S(\boldsymbol{x}_t^*))$

of the input image (*e.g.*, face accessories). In FaceSec, we use two attacks of this category: *eyeglass frame attack* [97] and *sticker attack*. The former allows large perturbations within an eyeglass frame, and it can successfully mislead VGG-based face recognition systems [88]. We propose the latter to produce pixel-level perturbations that are added on less important face areas than the eyeglass frame, *i.e.*, the two cheeks and forehead of human faces, as illustrated in Fig. 2.3 and 3.2. Typically, the stickers are rectangular occlusions, which cover a total of about 20% area of an input face image.

Grid-level Physically Realizable Attack. In practice, pixel-level perturbations are not printable on face accessories made of *coarse* materials, such as face masks using cloths and non-woven fabrics. To address this issue, we propose the grid-level physically realizable face mask attack, which adds a color grid on face masks, as shown in Fig. 3.2.

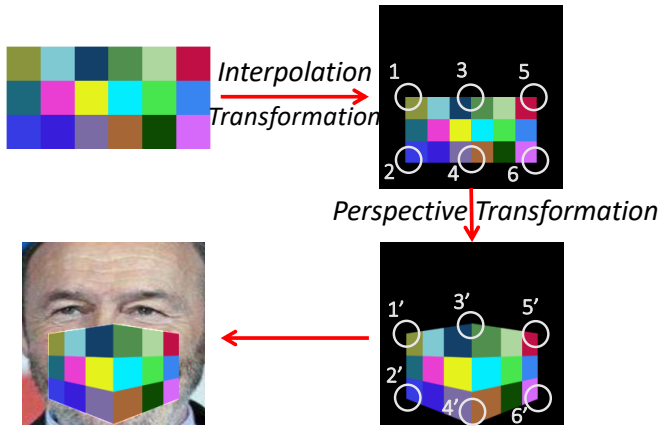

Figure 3.3: Transformations for the grid-level face mask attack.

Formulation. The optimization formulations of the proposed grid-level face mask attacks under different settings are presented in Table 3.1. Here, S is the target face recognition model, $\boldsymbol{x}$ is the original input face image, and $\boldsymbol{\delta}$ is the adversarial perturbation. M denotes the mask matrix that constrains the area of perturbation; it has the same dimension as $\boldsymbol{\delta}$ and contains 1s where perturbation is allowed, and 0s where there is no perturbation. For closed-set systems, ℓ denotes the softmax cross-entropy loss function, y is the identity of $\boldsymbol{x}$, and y_t is the target identity for impersonation attacks. For open-set settings, d is the cosine distance (using one to minus cosine similarity), $\boldsymbol{x}^*$ is the gallery image of $\boldsymbol{x}$, and $\boldsymbol{x}_t^*$ is the target gallery image for impersonation. $\mathcal{T}$ represents a set of transformations that convert the color matrix $\boldsymbol{\delta}$ to a face mask with a color grid in digital space. Specifically, $\mathcal{T}$ contains two transformations: *interpolation transformation* and *perspective transformation*, which are detailed below.

Interpolation Transformation. The interpolation transform starts from a $a \times b$ color matrix $\boldsymbol{\delta}$ and uses the following two steps to scale $\boldsymbol{\delta}$ into a face image, as illustrated in Fig 3.3: First, it resizes the color matrix from $a \times b$ to a rectangle $\boldsymbol{\delta}'$ with $c \times d$ pixels, so as to reflect the size of a face mask in a face image in digital space while preserving the layout of the color grids represented by $\boldsymbol{\delta}$. Specifically, in FaceSec, each input face image has 224×224 pixels. Let $(a, b) = (8, 16)$ and $(c, d) = (80, 160)$. Then, we put the face mask $\boldsymbol{\delta}'$ into a background image, such that the pixels in the rectangular area have the same value with $\boldsymbol{\delta}'$, and those outside the face mask area have values of 0s.

Perspective Transformation. Once the rectangle $\boldsymbol{\delta}'$ is embedded into a background image, we use a 2-D alignment that relies on the perspective transformation by the following steps. First, we divide $\boldsymbol{\delta}'$ into a left half part $\boldsymbol{\delta}'_L$ and a right half part $\boldsymbol{\delta}'_R$; each is rectangular with four corners. Then, we apply the perspective transformation to project each part to be with aligned coordinates, such that the new coordinates align with the position when a face mask is put on a human face, as shown in Fig 3.3. Let $\boldsymbol{\delta}''_L$ and $\boldsymbol{\delta}''_R$ be the left and right part of the aligned face mask, the perspective transformation aims to find a 3×3 matrix $N_k (k \in \{L, R\})$ for each part such that the coordinates satisfy

$$\boldsymbol{\delta}''_k(x, y) = \boldsymbol{\delta}'_k(u, v), \quad k \in \{L, R\},$$

where

$$u = \frac{N_k(1,1)x + N_k(1,2)y + N_k(1,3)}{N_k(3,1)x + N_k(3,2)y + N_k(3,3)},$$

and

$$v = \frac{N_k(2,1)x + N_k(2,2)y + N_k(2,3)}{N_k(3,1)x + N_k(3,2)y + N_k(3,3)}.$$

Finally, we merge $\boldsymbol{\delta}''_L$ and $\boldsymbol{\delta}''_R$ to obtain the aligned grid-level face mask.

Algorithm 1 Computing adversarial face mask.

Input: Target system S;
Input face image $\boldsymbol{x}$ and its identity y;
The number of iterations T;
Step size α;
Momentum parameter μ.

Output: The color grid matrix of adversarial face mask $\boldsymbol{\delta}_T$.

Initialize the color grid $\boldsymbol{\delta}_0 := \mathbf{0}$, momentum $\boldsymbol{g}_0 := \mathbf{0}$;
Use interpolation and perspective transformations to convert $\boldsymbol{\delta}_0$: $\boldsymbol{\delta}_0^{''} := \mathcal{T}(\boldsymbol{\delta}_0)$;
for each $t \in [0, T-1]$ **do**
  $\boldsymbol{g}_{t+1} := \mu \cdot \boldsymbol{g}_t + \frac{\nabla_{\boldsymbol{\delta}_t} \ell(S(\boldsymbol{x}+M\cdot\boldsymbol{\delta}_t^{''}),y)}{||\ell(S(\boldsymbol{x}+M\cdot\boldsymbol{\delta}_t^{''}),y)||_1}$;
  $\boldsymbol{\delta}_{t+1} := \boldsymbol{\delta}_t + \alpha \cdot \text{sign}(\boldsymbol{g}_{t+1})$;
  $\boldsymbol{\delta}_{t+1}^{''} := \mathcal{T}(\boldsymbol{\delta}_{t+1})$;
  Clip $\boldsymbol{\delta}_{t+1}^{''}$ such that pixel values of $\boldsymbol{x} + M \cdot \boldsymbol{\delta}_{t+1}^{''}$ are in $[0, 255/255]$;
end for
return $\boldsymbol{\delta}_T$.

Computing Adversarial Face Masks. The algorithm for computing the color grid for adversarial face mask attack is outlined in Algorithm 1. Here, we use the dodging attack on closed-set systems as an example. The algorithms for other settings are similar. Note that $\boldsymbol{\delta}_T$ is the resulting color grid, and the corresponding adversarial example is $\boldsymbol{x} + M \cdot \mathcal{T}(\boldsymbol{\delta}_T)$.

3.2.2 Attacker's System Knowledge (K)

The key components of a face recognition system S are the training set D and neural architecture h. It is natural to ask how do these two components contribute to the robustness against adversarial attacks. From the attackers' perspective, we propose several evaluation scenarios in FaceSec, which represent adversarial attacks performed under different knowledge levels on D and h.

Zero Knowledge. Both D and h are invisible to the attacker, *i.e.*, $K = \emptyset$. This is the weakest adversarial setting, as no critical information of S is leaked. Thus, it provides an

upper bound for robustness evaluation on S. In this scenario, the attacks are referred to as *black-box attacks*, where the attacker needs no internal details of S to compromise it.

There are two general ways towards black-box attacks, *query-based attack* [14, 83] and *transfer-based attack* [84]. We employ the latter because the former attack requires a large number of online probes to repeatedly estimate the loss gradients of S on adversarial examples, which is less practical than fully offline attacks when access to prediction decisions is unavailable. The latter method is built upon the *transferability* of adversarial examples [23, 84]. Specifically, an attacker first collects a sufficient of training samples and builds a surrogate training set D'. Then, a surrogate system S' is constructed by training a surrogate neural architecture h' on D' for the same task as S, *i.e.*, $S' = f(h'; D')$. Afterward, the attacker obtains a set of adversarial examples by performing *white-box attacks* on the surrogate system S', which constitutes the transferable adversarial examples for evaluating the robustness of S.

Training Set. This scenario enables the assessment of the robustness of the training set of S in adversarial settings. Here, only the training set D is visible to the attacker, *i.e.*, $K = \{D\}$. Without knowing h, an attacker constructs a surrogate system S' by training a surrogate neural architecture h' on D, *i.e.,* $S' = f(h'; D)$. Then, the attacker performs the transfer-based attack aforementioned on S' and evaluates S by using the transferred adversarial examples.

Neural Architecture. Similarly, the attacker may only know the neural architecture h of S but has no access to the training set D, *i.e.*, $K = \{h\}$. This enables us to evaluate the robustness of the neural architecture h of S. Without knowing D, the attacker can build its surrogate system $S' = f(h; D')$ and conduct the transfer-based attack to evaluate S.

Full Knowledge. In the worst case, the attacker can have an accurate knowledge of both the training set D and neural architecture h (*i.e.*, $K = \{D, h\}$). Thus, it provides a lower

bound for robustness evaluation on S. In this scenario, the attacker can fully reproduce S in an offline setting and then performs *white-box attacks* on S.

The evaluation method described above is based on the assumption that the adversarial examples in response to a surrogate system S' can always mislead the target system S. However, there is no theoretical guarantee, and recent studies show that some transferred adversarial examples can only fool the target system S with a low success rate [67].

To boost the transferability of adversarial examples produced on the surrogate system, we leverage two techniques: *momentum-based attack* [23] and *ensemble-based attack* [23, 67]. First, inspired by the momentum-based attack, we integrate the *momentum term* into the iterative process of the white-box attacks on the surrogate system S' to stabilize the update directions and avoid the local optima. Thus, the resulting adversarial examples are more transferable. Second, when the neural architecture h of the target system S is unavailable, we construct the surrogate system S' using an ensemble of models with different neural architectures rather than a single model, *i.e.*, $h' = \{h'_i\}_{i=1}^{k}$, where $\{h'_i\}_{i=1}^{k}$ is an ensemble of k models. Specifically, we aggregate the output logits of $h_i (i \leq k)$ in a similar way to [23]. The rationale behind this is that if an adversarial example can fool multiple models, it is more likely to mislead other models.

3.2.3 Attacker's Goal (G)

In addition to the attacker's system knowledge about S, adversarial attacks can differ in specific goals. In FaceSec, we are interested in the following two types of attacks with different goals:

Dodging/Non-targeted. In a dodging attack, an attacker aims to have his/her face misidentified as another arbitrary face. *e.g.*, the attacker can be a terrorist who wants to

Table 3.2: Optimization formulations by the attacker's goal.

Target System	Attacker's Goal	Formulation
Closed-set	Dodging	$\max_{\boldsymbol{\delta}} \ell(S(\boldsymbol{x}+M\boldsymbol{\delta}), y), \quad s.t.\ \|\|\boldsymbol{\delta}\|\|_p \leq \epsilon$
Closed-set	Impersonation	$\min_{\boldsymbol{\delta}} \ell(S(\boldsymbol{x}+M\boldsymbol{\delta}), y_t), \quad s.t.\ \|\|\boldsymbol{\delta}\|\|_p \leq \epsilon$
Open-set	Dodging	$\max_{\boldsymbol{\delta}} d(S(\boldsymbol{x}+M\boldsymbol{\delta}), S(\boldsymbol{x}^*)), \quad s.t.\ \|\|\boldsymbol{\delta}\|\|_p \leq \epsilon$
Open-set	Impersonation	$\min_{\boldsymbol{\delta}} d(S(\boldsymbol{x}+M\boldsymbol{\delta}), S(\boldsymbol{x}_t^*)), \quad s.t.\ \|\|\boldsymbol{\delta}\|\|_p \leq \epsilon$

bypass a face recognition system for biometric security checking. As the dodging attack has no specific identity as which it aims to predict an input face image, it is also called the *non-targeted attack*.

Impersonation/Targeted. In an impersonation/targeted attack, an attacker seeks to produce an adversarial example that is misrecognized as a target identity. For example, the attacker may try to camouflage his/her face to be identified as an authorized user of a laptop, which uses face recognition for authentication.

In FaceSec, we formulate the dodging attack and impersonation attack as constrained optimization problems, corresponding to different face recognition systems and the attacker's goals, as shown in Table 3.2. Here, ℓ denotes the softmax cross-entropy loss used in closed-set systems, d represents the distance metric for open-set systems (*e.g.*, the cosine distance obtained using one to minus cosine similarity), $(\boldsymbol{x}, y)$ is the input face image and the associated identity, $\boldsymbol{\delta}$ is the adversarial perturbation, S represents a face recognition system which is built on either a single model or an ensemble of models with different neural architectures, M denotes the mask matrix that constrains the area of perturbation (similar to Eq. (2.2)), ϵ is the ℓ_p-norm bound of $\boldsymbol{\delta}$. For closed-set systems, we use y_t to represent the target identity of impersonation attacks. For open-set systems, we use $\boldsymbol{x}^*$ to denote the gallery face image that belongs to the identity as $\boldsymbol{x}$, and $\boldsymbol{x}_t^*$ as the gallery image for the target identity of impersonation.

Note that the formulations listed in Table 3.2 work for both digital attacks and physically realizable attacks: For the former, we use a small value of ϵ and let M be an all-one matrix to ensure imperceptible perturbations on the entire image. For the latter, we use a large ϵ and let M to constrain $\boldsymbol{\delta}$ in a small area of $\boldsymbol{x}$.

3.2.4 Attacker's Capability (C)

In practice, even when the attackers share the same system knowledge and goal, their capabilities can still be different due to the time and/or budget constraints, such as the budget for printing adversarial eyeglass frames [97]. Thus, in FaceSec, we consider two types of attacks corresponding to different attacker's capabilities: *individual attack* and *universal attack*.

Individual Attack. The attacker has a strong capability with enough time and budget to produce a specific perturbation for each input face image. In this case, the optimization formulations are the same as those shown in Table 3.2.

Universal Attack. The attacker has a time/budget constraint such that he/she is only able to generate a *face-agnostic* perturbation that fools a face recognition system on a batch of face images instead of every input.

One common way to compute a universal perturbation is to sequentially find the *minimum* perturbation of each data point in the batch and then aggregate these perturbations [81]. However, this method requires orders of magnitude running time: it processes only one image at each iteration, so a large number of iterations are needed to obtain a satisfactory universal perturbation. Moreover, it only focuses on digital attacks and cannot be generalized to physically realizable attacks, which seek large perturbations in a restricted area rather than the minimum perturbations.

Algorithm 2 Finding universal perturbations.

Input: Target system S;
Input face image batch $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$;
The number of iterations T;
Step size α;
Momentum parameter μ.
Output: The universal perturbation $\boldsymbol{\delta}_T$ for $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$.

1: Initialize $\boldsymbol{\delta}_0 := \mathbf{0}$, $\boldsymbol{g}_0 := \mathbf{0}$;
2: **for** each $t \in [0, T-1]$ **do**
3: **for** each $i \in [1, N]$ **do**
4: $\ell_{i,t} := \ell(S(\boldsymbol{x}_i + M \cdot \boldsymbol{\delta}_t), y_i)$;
5: **end for**
6: $\ell_t = \min\{\ell_{i,t}\}_{i=1}^N$;
7: $\boldsymbol{g}_{t+1} := \mu \cdot \boldsymbol{g}_t + \frac{\nabla_{\boldsymbol{\delta}_t} \ell_t}{||\ell_t||_1}$;
8: $\boldsymbol{\delta}_{t+1} := \boldsymbol{\delta}_t + \alpha \cdot \text{sign}(\boldsymbol{g}_{t+1})$;
9: Clip $\boldsymbol{\delta}_{t+1}$ such that pixel values of $\boldsymbol{x} + M \cdot \boldsymbol{\delta}_{t+1}$ are in $[0, 255/255]$;
10: **end for**
11: **return** $\boldsymbol{\delta}_T$.

To address these issues, we formulate the universal attack as a *maxmin optimization* as follows (using the dodging attack on closed-set systems as an example):

$$\max_{\boldsymbol{\delta}} \min\{\ell(S(\boldsymbol{x}_i + M\boldsymbol{\delta}), y_i)\}_{i=1}^N, \quad s.t.\ ||\boldsymbol{\delta}||_p \leq \epsilon, \tag{3.2}$$

where $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$ is a batch of input images that share the universal perturbation $\boldsymbol{\delta}$. Compared to [81], our approach has several advantages: First, we can significantly improve the efficiency by processing images batchwise. Second, our formulation can explicitly control the universality of the perturbation by setting different values of N. Third, our method can be generalized to both digital attacks and physically realizable attacks. Details of our algorithm for solving the optimization problem in Eq. (3.2) is presented in Algorithm 2. Here, we use the dodging attack on closed-set systems as an example. The algorithms for other settings are similar. Note that in practice, the pseudocode from Line 3 to Line 6 in Algorithm 2 can be executed in a paralleled manner by using GPUs. Therefore, compared to traditional methods that

Table 3.3: Optimization formulations of universal dodging attacks.

Target System	Perturbation Type	Formulation
Closed-set	Pixel-level	$\max_{\boldsymbol{\delta}} \min\{\ell(S(\boldsymbol{x}_i + M\boldsymbol{\delta}), y_i)\}_{i=1}^{N}, \quad s.t.\ ||\boldsymbol{\delta}||_p \leq \epsilon$
Closed-set	Grid-level	$\max_{\boldsymbol{\delta}} \min\{\ell(S(\boldsymbol{x}_i + M \cdot \mathcal{T}(\boldsymbol{\delta})), y_i)\}_{i=1}^{N}$
Open-set	Pixel-level	$\max_{\boldsymbol{\delta}} \min\{d(S(\boldsymbol{x_i} + M\boldsymbol{\delta}), S(\boldsymbol{x}_i^*))\}_{i=1}^{N}, \quad s.t.\ ||\boldsymbol{\delta}||_p \leq \epsilon$
Open-set	Grid-level	$\max_{\boldsymbol{\delta}} \min\{d(S(\boldsymbol{x_i} + M \cdot \mathcal{T}(\boldsymbol{\delta})), S(\boldsymbol{x}_i^*))\}_{i=1}^{N}$

iterate every data point to find a universal perturbation [81], our approach can achieve a significant speedup.

The formulations of universal perturbations in different settings are presented in Table 3.3. In FaceSec, we mainly focus on universal dodging attacks. Effective universal impersonation attack is still an open problem, and we leave it for future work.

3.3 Experimental Results

In this section, we evaluate a variety of face recognition systems using FaceSec on both closed-set and open-set tasks under different adversarial settings.

3.3.1 Experimental Setup

Datasets. For closed-set systems, we use a subset of the VGGFace2 dataset [12]. Specifically, we select 100 classes, each of which has 181 face images. For open-set systems, we employ the VGGFace2, MS-Celeb-1M [36], CASIA-WebFace [129] datasets for training surrogate models, and the LFW dataset [47] for testing.

Neural Architectures. The face recognition systems with five different neural networks are evaluated in our experiments: VGGFace [88], InceptionResNet [107], IResNet18 [60], IResNet50 [60], and IResNet101 [60].

Table 3.4: Open-set face recognition systems in our experiments.

Target Model	**Training Set**	**Neural Architecture**	**Loss**
VGGFace [88]	VGGFace [88]	VGGFace [88]	Triplet [88]
FaceNet [26]	CASIA-WebFace [129]	InceptionResNet [107]	Triplet [95]
ArcFace18 [90]	MS-Celeb-1M [36]	IResNet18 [60]	ArcFace [22]
ArcFace50 [90]	MS-Celeb-1M [36]	IResNet50 [60]	ArcFace [22]
ArcFace101 [90]	MS-Celeb-1M [36]	IResNet101 [60]	ArcFace [22]

Attack Models. We perform both digital attacks and physically realizable attacks in our evaluation. For digital attacks, we choose the PGD attack [71] as the representative. For physical realizable attacks, we use the three attacks introduced in Section 3.2: the eyeglass frame attack, the sticker attacks, and the grid-level face mask attacks.

Defense Baselines. Two defense strategies are used in our experiments. (1) Rectangular occlusion attacks (henceforth, DOA) [122]: the state-of-the-art adversarially robust training scheme for face recognition; (2) Randomized Smoothing (henceforth, RS) [16]: the provably robust classification against ℓ_2 attacks. The defense strategies are evaluated when each face recognition system is trained on non-adversarial data.

Evaluation Metric. We use *attack success rate* = 1 - accuracy as the evaluation metric. Specifically, a higher attack success rate indicates that a face recognition system is more fragile in adversarial settings, while a lower rate shows higher robustness against adversarial attacks.

Implementation. For open-set face recognition, we directly applied five publicly available pre-trained face recognition models using different datasets and neural architectures, as summarized in Table 3.4. At prediction stage, we used 100 photos randomly selected from frontal images in the LFW dataset [47], each of which is aligned by using MTCNN [131] and corresponds to one identity. And we used another 100 photos of the same identities as the test gallery. We computed the cosine similarity between the feature vectors of the test and

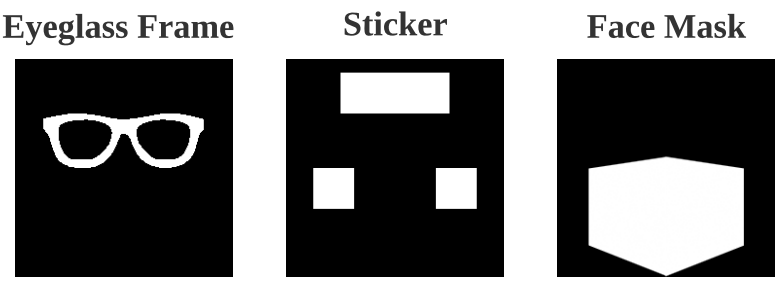

Figure 3.4: Mask matrices for physically realizable attacks in FaceSec.

gallery photos. If the score is above a threshold corresponding to a False Acceptance Rate of 0.001, then the test photo is predicted to have the same identity as the gallery photo.

For closed-set face recognition, we randomly split each class of the VGGFace2 subset into three parts: 150 for training, 30 for validation, and 1 for testing. To train closed-set models, we used standard transfer learning with the open-set models listed in Table 3.4. Specifically, we initialized each closed-set model with the corresponding open-set model, and then added a final fully connected layer, which contains 100 neurons. Unless otherwise specified, each model was trained for 60 epochs with a training batch size of 64. We used the Adam optimizer [53] with an initial learning rate of 0.0001, then dropped the learning rate by 0.1 at the 20th and 35th epochs.

For each physically realizable attack in FaceSec, we used 255/255 as the ℓ_∞ norm bound for perturbations allowed, and ran each attack for 200 iterations. For the PGD attack [71], we used an ℓ_∞ bound 8/255 and 40 iterations. The dimension of the color grid for face mask attacks is set to 16×8. The mask matrices that constrain the areas of perturbations for physically realizable attacks are visualized in Fig. 3.4.

Table 3.5: Attack success rate of dodging attacks on closed-set face recognition systems by the attacker's system knowledge. Z represents zero knowledge, T is training set, A is neural architecture, and F represents full knowledge.

Target System	**Attack Type**	**Attacker's System Knowledge**			
		Z	**T**	**A**	**F**
VGGFace	PGD	0.40	0.51	0.93	0.94
	Eyeglass Frame	0.23	0.28	0.70	0.99
	Sticker	0.05	0.06	0.47	0.98
	Face Mask	0.26	0.32	0.63	1.00
FaceNet	PGD	0.83	0.83	1.00	1.00
	Eyeglass Frame	0.13	0.16	0.90	1.00
	Sticker	0.01	0.01	0.92	1.00
	Face Mask	0.30	0.42	0.83	1.00
ArcFace18	PGD	0.87	0.92	0.97	1.00
	Eyeglass Frame	0.06	0.06	0.44	1.00
	Sticker	0.01	0.01	0.37	1.00
	Face Mask	0.27	0.33	0.71	1.00
ArcFace50	PGD	0.87	0.90	0.81	0.99
	Eyeglass Frame	0.09	0.12	0.44	0.99
	Sticker	0.00	0.01	0.14	0.94
	Face Mask	0.29	0.36	0.67	0.99
ArcFace101	PGD	0.81	0.78	0.86	0.96
	Eyeglass Frame	0.03	0.03	0.26	0.98
	Sticker	0.04	0.04	0.08	0.95
	Face Mask	0.26	0.36	0.54	0.99

3.3.2 Robustness of Face Recognition Components

We begin by using FaceSec to assess the robustness of face recognition components in various adversarial settings. For a given target face recognition system S and a perturbation type P, we evaluate the training set D and neural architecture h of S with the four evaluation scenarios presented in Section 3.2.2. Specifically, when h is invisible to the attacker, we construct the surrogate system S' by ensembling the models built on the other four neural architectures shown in Table 3.4. In the scenarios where the attacker has no access to D, we build the surrogate training set D' with another VGGFace2 subset that has the same

Table 3.6: Attack success rate of dodging attacks on open-set face recognition systems with zero knowledge.

Target Model	**Attack Type**			
	PGD	**Sticker**	**Eyeglass Frame**	**Face Mask**
VGGFace	0.26	0.56	0.79	0.67
FaceNet	0.55	0.13	0.54	0.62

classes as D in closed-set settings, and use the other four training sets listed in Table 3.4 for open-set tasks. We present the experimental results for dodging attacks on closed-set face recognition systems in Table 3.5, and the results for zero-knowledge dodging attacks on open-set VGGFace and FaceNet in Table 3.6. The other results can be found in Appendix A.1. Additionally, we evaluate the efficacy of using *momentum* and *ensemble* methods to improve transferability of adversarial examples, which is detailed in Appendix A.2.

It can be seen from Table 3.5 that: *the neural architecture is significantly more fragile than the training set in most adversarial settings.* For example, when only the neural architecture is exposed to the attacker, the sticker attack has a high success rate of 0.92 on FaceNet. In contrast, when the attacker only knows the training set, the attack success rate significantly drops to 0.01. In addition, by comparing each row of Table 3.5 that corresponds to the same target system, we observe that *digital attacks (PGD) are considerably more potent than their physically realizable counterparts on closed-set systems, while grid-level perturbations on face masks are noticeably more effective than pixel-level physically realizable perturbations (i.e., the eyeglass frame attack and the sticker attack).* Moreover, by comparing the zero knowledge attacks in Table 3.5 and 3.6, we find that *open-set face recognition systems are more vulnerable than closed-set systems* such that nearly all perturbation types of attacks (even the black-box sticker attack that often fails in closed-set) tend to be more likely to successfully transfer across different open-set systems (*i.e.*, these are more susceptible to black-box attacks), which should raise more concerns about their security.

Table 3.7: Attack success rate of dodging attacks on closed-set face recognition systems by the universality of adversarial examples. Here, N represents the batch size of face images that share a universal perturbation.

Target System	**Attack Type**	**Attacker's Capability**			
		N=1	**N=5**	**N=10**	**N=20**
VGGFace	PGD	0.94	0.86	0.31	0.15
	Eyeglass Frame	0.99	0.91	0.52	0.23
	Sticker	0.98	0.66	0.34	0.09
	Face Mask	1.00	1.00	0.88	0.56
FaceNet	PGD	1.00	1.00	0.80	0.21
	Eyeglass Frame	1.00	1.00	1.00	0.62
	Sticker	1.00	1.00	0.98	0.61
	Face Mask	1.00	1.00	1.00	0.91
ArcFace18	PGD	1.00	1.00	0.64	0.08
	Eyeglass Frame	1.00	0.96	0.44	0.08
	Sticker	1.00	0.56	0.09	0.00
	Face Mask	0.99	0.92	0.90	0.67
ArcFace50	PGD	1.00	0.80	0.37	0.05
	Eyeglass Frame	0.99	0.81	0.38	0.07
	Sticker	0.91	0.28	0.06	0.00
	Face Mask	0.99	0.98	0.81	0.72
ArcFace101	PGD	0.96	0.91	0.24	0.03
	Eyeglass Frame	0.98	0.71	0.19	0.02
	Sticker	0.93	0.15	0.03	0.00
	Face Mask	0.99	0.92	0.90	0.67

3.3.3 Robustness Under Universal Attacks

Next, we use FaceSec to evaluate the robustness of face recognition systems with various extents of adversarial universality by setting the parameter N in Eq. (3.2) to different values. For a given N, we split the testing set into mini-batches of size N, and produce a specific perturbation for each batch. Note that when $N = 1$, a universal attack is reduced to an individual attack. Table 3.7 shows the experimental results for universal dodging attacks on closed-set systems. The other results are presented in Appendix A.3.

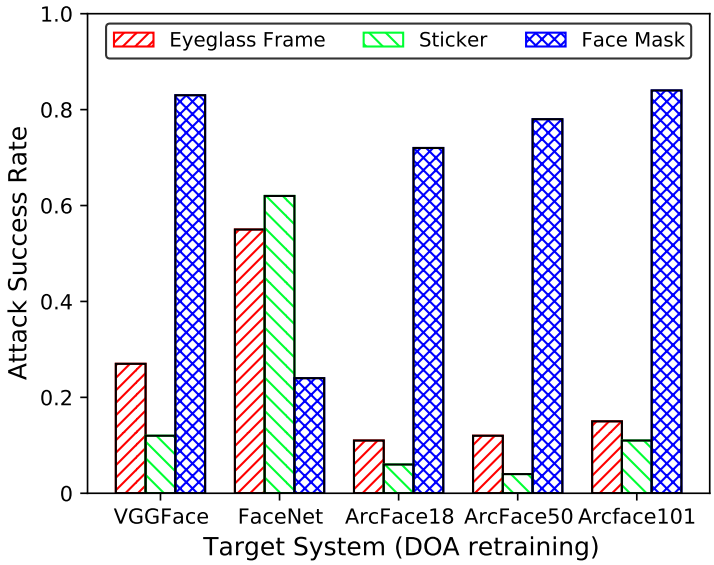

Figure 3.5: Attack success rate of dodging physically realizable attacks on closed-set systems with DOA retraining.

Our first observation is that *face recognition systems are significantly more vulnerable to the universal face masks than other types of universal perturbations.* Under a large extent of universality (*e.g.*, when $N = 20$), face mask attacks remain > 0.5 success rates. Particularly noteworthy is the universal face mask attacks on FaceNet, which can achieve a rate as high as 0.91. In contrast, other universal attacks can have relatively low success rates (*e.g.*, 0.08 for eyeglass frame attack on ArcFace18). The second observation is that *the robustness of a face recognition system against different types of universal perturbations is highly dependent on its neural architecture.* For example, the ArcFace101 architecture is more robust than the others in most settings, while FaceNet tends to be the most fragile one.

3.3.4 Is "Robust" Face Recognition Really Robust?

While numerous approaches have been proposed for making deep neural networks more robust to adversarial examples, only a few [122] focus on defending against physically realizable

attacks on face recognition systems. These defense approaches have achieved good performance for certain types of realizable attacks and neural architectures, but their effectiveness for other types of attacks and face recognition systems remains unknown. In this section, we apply FaceSec to evaluate the state-of-the-art defense paradigms. Specifically, we first use DOA [122], a method that defends closed-set VGGFace against eyeglass frame attacks [97] to re-train each closed-set system. We then evaluate the refined systems using the three physically realizable attacks included in FaceSec. Fig. 3.5 shows the experimental results for dodging attacks.

Our first observation is that *the state-of-the-art defense approach, DOA, fails to defend against the grid-level perturbations on face masks for most neural architectures*. Specifically, face mask attacks can achieve > 0.7 success rates on four out of the five face recognition systems refined by DOA. Moreover, we observe that *adversarial robustness against one type of perturbation can not be generalized to other types*. For example, while VGGface-DOA exhibits a relatively high level of robustness (more than a 70% accuracy) against pixel-level perturbations (*i.e.*, stickers and eyeglass frames), it is very vulnerable to grid-level perturbations (*i.e.*, face masks). In contrast, using DOA on FaceNet can successfully defend face mask perturbations with the attack success rate significantly dropping from 1.0 to 0.24, but it's considerably less effective against eyeglass frames and stickers. In summary, these results show that the effectiveness of defense is highly dependent on the nature of perturbation and neural architectures, which in turn, indicates that it is critical to consider different types of attacks and neural architectures when evaluating a defense method for face recognition systems.

3.4 Conclusion

In this chapter, we present FaceSec, a fine-grained robustness evaluation framework for face recognition systems. FaceSec incorporates four evaluation dimensions and can work on both face identification and verification of open-set and closed-set systems. To our best knowledge, FaceSec is the first-of-its-kind platform that supports evaluating the risks of different components of face recognition systems from multiple dimensions and under various adversarial settings. The comprehensive and systematic evaluations on five state-of-the-art face recognition systems demonstrate that FaceSec can greatly help understand the robustness of the systems against both digital and physically realizable attacks. We envision that FaceSec can serve as a useful framework to advance future research of adversarial learning on face recognition.

Part II

Robust Learning against Decision-Time Attacks

Chapter 4

How Robust Is Robust ML?

Previous chapters have shown that ML models are often susceptible to *decision-time attacks*[2], in which an adversary makes changes to the input (such as malware) in order to avoid being detected. A conventional approach to evaluate ML robustness to such attacks, as well as to design robust ML, is by considering simplified *feature-space* models of attacks, where the attacker changes ML features directly to effect evasion, while minimizing or constraining the magnitude of this change. In this chapter, We investigate the effectiveness of this approach to designing robust ML in the face of attacks that can be realized in actual malware (*realizable attacks*). We first propose a general methodological framework for evaluating the validity of mathematical models of ML evasion attacks. We then demonstrate that in the context of structure-based PDF malware detection, such techniques appear to have limited effectiveness, but they are effective with content-based detectors. In other words, the widely used feature space attack models can be inadequate as a means for ML defense.

[2]In binary classification, such attacks are also called evasion attacks. We make these two terms interchangeable in this chapter.

4.1 Overview

As shown in previous studies, ML-based techniques are often susceptible to *adversarial examples*, an important special case of which are *decision-time attacks*, or *evasion attacks* in the case of binary classification. In a prototypical case of a decision-time attack, an adversary modifies malware code so that the resulting malware is categorized as benign by ML, but still successfully executes the malicious payload [28, 34, 72, 103, 126]. An even broader class of adversarial examples features attacks that manipulate an object, such as a stop sign, so that a computer vision pipeline misclassifies it as another object (such as a speed limit sign) [25, 33, 97].

In response, a host of methods emerged for making ML robust to adversarial examples, the most potent of which are those based on game-theoretic approaches, robust optimization (including certified robustness), and adversarial retraining [10, 33, 58, 71, 91, 121, 124, 132]. A fundamental ingredient in all of these are *feature-space models of attacks*. Specifically, the attacker is assumed to directly modify values of features, with either a constraint or a penalty on the aggregate feature change measured in terms of an l_p norm.

Such feature-space models of attacks are clearly abstractions of reality. First, arbitrary modifications of feature values may not be *realizable*. For example, adding a benign object to a malicious PDF (with no other changes) necessarily increases its size, and so setting the associated feature to 1 (from 0) and simultaneously reducing file size may not be practically feasible. Second, the key goal for an adversary is to create a target malicious effect, such as to execute a malicious payload. Limiting feature modifications to be small in some l_p norm clearly need not capture this: one can insert many no-ops (resulting in a large change according to an l_p norm) with no impact on malicious functionality, and conversely, minimal changes (such as removing a Javascript tag) may break malicious functionality. Nevertheless,

an implicit assumption in robust ML approaches is that the feature-space models capture reality sufficiently to yield ML models that are robust even to realizable attacks. *The goal of our work in this chapter is to evaluate the validity of this implicit assumption* in the context of PDF malware detection.

Our first contribution is a general methodological framework for evaluating the validity of mathematical models of ML evasion attacks. At the core of the framework is a conceptual model of defense and attack based on a Stackelberg game [10], where we assume to have an attack oracle that can be queried to obtain an attack for a given defense. The second ingredient is iterative adversarial retraining that can make use of an arbitrary learning algorithm and automated attack, enabling general applicability of the framework, and a fair comparison in validation (since the defensive approach is the same whether the attack is realizable or in feature space). The third feature of the framework is an evaluation measure which quantifies ML robustness by pitting ML against a *realizable attack*, which must avoid being detected *and* preserve malicious functionality as validated using a sandbox [19, 35].

Our second contribution is to evaluate feature-space evasion attack models in the context of PDF malware detection, using EvadeML as a realizable attack [126]. Specifically, we consider four ML-based approaches for PDF malware detection: two based on features that capture PDF file structure (SL2013 [102] and Hidost [104]), and two based on PDF file content (two Mimicus variants of PDFRate [100, 103]). In all cases, we show that successful defense against a given realizable attack is feasible (by retraining with this attack). In the case of structure-based detectors, we demonstrate that adversarial retraining in the feature space does not lead to adequate robustness against realizable attacks. In contrast, adversarial retraining in the feature space is effective in the case of content-based detectors. In other words, the nature of the feature space can matter a great deal.

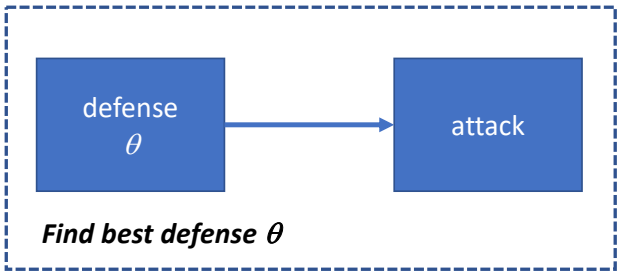

Figure 4.1: A conceptual model of how an attack (either realizable or using a feature-space model) can be used in improving ML security. Let defense be parameterized by θ, and an attack *reacting* to the particular defense θ (e.g., attacker evades the learned ML model $h(x)$). We wish to choose the best defense θ against such a reactive attacker, as captured by our attack model.

4.2 A Framework for Validating Models of ML Evasion Attacks

Our main goal is to evaluate whether robust ML approaches that make use of feature-space models of evasion attacks are, indeed, robust against *real*—realizable—attacks.

We start with a conceptual model of defense and attack illustrated in Figure 4.1. We can view this conceptual model as a Stackelberg game between ML ("defender"), who first chooses a defense θ (in our case, the learned model $h(x)$) and the attacker, who finds an optimal attack that *reacts* to the particular defense θ. An *attack model* captures how the attacker changes behavior in response to the defense θ. The defender's goal is to choose the best defense θ against such a reactive attacker, as captured by the attack model. Indeed, this is a common way to model the adversarial evasion problem in prior literature [10, 59, 115]. This model has two useful features. First, the attack is treated as an oracle in the sense that it returns an attack for an arbitrary defense θ. This allows us, in principle, to design a defense against an arbitrary evasion attack, making no distinction between feature-space attack models and realizable attacks. Second, we can separately consider *defense* against a

specific attack (for example, a feature-space attack), and evaluation, which can use another attack (e.g., a realizable attack).

To be more precise, let $O(h; \mathsf{D})$ be an arbitrary attack which returns evasions given a dataset D and a classifier h, and let $u(h; O(h; \mathsf{D}))$ be the measure that the defender wishes to optimize (for example, accuracy on data *after* evasions). Then defense against the attack $O(h; \mathsf{D})$ amounts to solving the following optimization problem:

$$\max_h u(h; O(h; \mathsf{D})). \tag{4.1}$$

In practice, we need a means for approximately solving the optimization problem in Eq. ((4.1)) for an arbitrary attack. To this end, we make use of *iterative retraining*, an approach previously proposed for hardening classifiers against evasion attacks [51, 58]. In particular, we use a variant of iterative retraining with provable guarantees [58], which is outlined as follows:

1. Start with the initial classifier.
2. Execute the *evasion attack* for each malicious instance in training data to generate a new feature vector.
3. Add all new data points to training data (removing any duplicates), and retrain the classifier.
4. Terminate after either a fixed number of iterations, or when no new evasions can be added.

Now, we describe our approach to validation.

Consider a model of an evasion attack, $\tilde{O}(h; \mathsf{D})$ (e.g., a feature-space attack model), which is a proxy for a "real" (realizable) attack, $O(h; \mathsf{D})$; note that each attack evades a given ML model h. We first find the defense against $\tilde{O}$ using the retraining procedure above; let the resulting robust classifier be $\tilde{h}$. Next, we *evaluate* $\tilde{h}$ by running the target realizable attack $O(\tilde{h}; \mathsf{D})$. Finally, we create a *baseline* h^*, which is a robust classifier against a target realizable attack O. We then evaluate how well $\tilde{h}$ performs, compared to h^*, against the target attack. For example, if we find that $\tilde{h}$ is ineffective against the target attack, we say that $\tilde{O}$ is a poor attack proxy, whereas if it remains robust, we view $\tilde{O}$ as a good proxy for the target attack O.

Putting everything together, we propose the following framework for validating the effectiveness of ML evasion models. Choose the ML algorithm that we wish to make robust. Next, consider a model of an evasion attack, $\tilde{O}(h)$ (e.g., a feature-space attack model), which is a proxy for a "real" (realizable) attack, $O(h)$; note that each attack evades a given ML model h. Let $u(h; O)$ be a measure of robustness of an ML model h *against the realizable attack*. Now,

1. Perform iterative retraining, using the model, $\tilde{O}(h)$; let $\tilde{h}$ be the resulting "hardened" ML model;

2. Perform iterative retraining, using the realizable attack, $O(h)$; let h^* be the ML model hardened against this attack;

3. The effectiveness of $\tilde{O}(h)$ *relative to* $O(h)$ (for which it is a proxy) is $\max\{u(h^*; O) - u(\tilde{h}; O), 0\}$.

Note that we generally expect that ML hardened using the realizable attack will be more robust against this attack than ML hardened using some other proxy (e.g., feature-space) attack. However, this is not always the case, and we do not require it; we simply use $u(h^*; O)$

as a fair baseline, and claim the model to be effective as long as it's nearly as good as this baseline, and certainly when it's better.

In the sequel, we use our framework to evaluate robustness of conventional feature-space approaches for hardening ML when they are confronted with a realizable attack.

4.3 Experimental Methodology

We use malicious PDF detection as a case study to investigate robustness of ML hardened using feature-space models of evasion attacks. We now describe our experimental methodology. We start with some background on PDF structure, and proceed to describe the specific ML-based detectors, evasion attacks (both realizable, and feature-space), datasets, and evaluation metrics used in our experiments.

4.3.1 PDF Document Structure

The Portable Document Format (PDF) is an open standard format used to present content and layout on different platforms. A PDF file structure consists of four parts: *header*, *body*, *cross-reference table* (CRT), and *trailer*. The header contains information such as the magic number and format version. The body is the most important element of a PDF file, which comprises multiple PDF objects that constitute the content of the file. These objects can be one of the eight basic types: Boolean, Numeric, String, Null, Name, Array, Dictionary, and Stream. They can be referenced from other objects via indirect references. There are other types of objects, such as JavaScript which contains executable JavaScript code. The CRT indexes objects in the body, while the trailer points to the CRT.

Table 4.1: Target classifiers.

Classifier	Feature type	Number of features
SL2013	Binary	6,087
Hidost	Binary	961
PDFRate-R	Real-valued	135
PDFRate-B	Binary	135

The relations between objects with cross-references can be described as a directed graph that presents their logical structure by using edges representing reference relations and nodes representing different objects. As an object can be referred to by its child node, the resulting logical structure is a directed cyclic graph. To eliminate the redundant references, the logical structure can be reduced to a structural tree with the breadth-first search procedure.

4.3.2 Target Classifiers

Several PDF malware classifiers have been proposed [19, 100, 102, 104]. For our study, we selected SL2013 [102], Hidost [104] and two variants of PDFRate [100] (termed PDFRate-R and PDFRate-B respectively), displayed in Table 4.1. SL2013 and its revised version, Hidost, are *structure-based* PDF classifiers, which use the logical structure of a PDF document to construct and extract features used in detecting malicious PDFs. PDFRate, on the other hand, is a *content-based* classifier, which constructs features based on *metadata* and *content* information in the PDF file to distinguish benign and malicious instances. Evasion attacks on both SL2013 and PDFRate classifiers, particularly of the realizable kind, have been developed in recent literature [102, 103, 104, 126], providing a natural evaluation framework for our purposes.

Structure-Based Classifiers. In this chapter, we use SL2013 and Hidost as the representatives of structure-based classifiers.

SL2013: SL2013 is a well-documented and open-source machine learning system using Support Vector Machines (SVM) with a radial basis function (RBF) kernel, and was shown to have state-of-the-art performance [102]. It employs structural properties of PDF files to discriminate between malicious and benign PDFs. Specifically, SL2013 uses the presence of particular *structural paths* as binary features to present PDF files in feature space. A structural path of an object is a sequence of edges in the reduced (tree) *logical structure*, starting from the catalog dictionary and ending at this object. Therefore, the structural path reveals the shortest reference path to an object. SL2013 uses 6,087 most common structural paths among 658,763 PDF files as a uniform set for classification.

Hidost: Hidost is an updated version of SL2013. It inherits all the characteristics of SL2013 and employs *structural path consolidation* (SPC), a technique to consolidate features which have the same or similar semantic meaning in a PDF. As the semantically equivalent structural paths are merged, Hidost reduces polymorphic paths and still preserves the semantics of logical structure, so as to improve evasion-robustness of SL2013 [104].

In our work, we employ the 961 features identified in the latest version of Hidost.

PDFRate: A Content-Based Classifier. The original PDFRate classifier uses a random forest algorithm, and employs PDF *metadata* and *content* features. The metadata features include the size of a file, author name, and creation date, while content-based features include position and counts of specific keywords. All features were manually defined by Smutz and Stavrou [100].

PDFRate uses a total of 202 features, but only 135 of these are publicly documented [99]. Consequently, in our work we employ the Mimicus implementation of PDFRate which was shown to be a close approximation [103]. Mimicus trained a surrogate SVM classifier with the documented 135 features and the same dataset as PDFRate, using both the SVM and

random forest classifiers, both performing comparably. We use the SVM implementation in our experiments to enable more direct comparisons with the structure-based classifiers that also use SVM. An important aspect of Mimicus is *feature standardization* on extracted data points performed by subtracting the mean of the feature value and dividing by standard deviation, transforming all features to be real-valued and zero-mean (henceforth, PDFRate-R). This surrogate was shown to have $\sim 99\%$ accuracy on the test data [100]. In addition, we construct a *binarized* variant of PDFRate (henceforth, PDFRate-B), where each feature is transformed into a binary feature by assigning 0 whenever the feature value is 0, and assigning 1 whenever the feature value is non-zero.

4.3.3 Realizable Evasion Attacks

The primary realizable attack in our study is EvadeML [126], which allows insertion, deletion, and swapping of objects, and is consequently a stronger attack than most other realizable attacks in the literature, which typically only allow insertion to ensure that malicious functionality is preserved. EvadeML assumes that the adversary has black-box access to the classifier and can only get classification scores of PDF files, and was shown to effectively evade both SL2013 and PDFRate [126]. It employs genetic programming (GP) to search the space of possible PDF instances to find ones that evade the classifier while maintaining malicious features. First, an initial population is produced by randomly manipulating a malicious PDF repeatedly. The manipulation is either a deletion, an insertion, or a swap operation on PDF objects. A deletion operation deletes a target object from the seed malicious PDF file. An insertion operation inserts an object from external benign PDF files (provided exogenously) after the target object. A swap operation replaces the entry of the target object with that of another object in the external benign PDFs. After the population is initialized, each variant is assessed by the Cuckoo sandbox [35] and the target classifier to evaluate its fitness.

The sandbox is used to determine if a variant preserves malicious behavior, such as API or network anomalies. The target classifier provides a classification score for each variant. If the score is above a threshold, then the variant is classified as malicious. Otherwise, it is classified as benign. If a variant is classified as benign but displays malicious behavior, or if GP reaches the maximum number of generations, then GP terminates with the variant achieving the best fitness score and the corresponding mutation trace is stored in a pool for future population initialization. Otherwise, a subset of the population is selected for the next generation based on their fitness evaluation. Afterward, the variants selected are randomly manipulated to generate the next generation of the population. EvadeML was used to evade SL2013 in [126]. The reported results show that it can automatically find evasive variants for all 500 selected malicious test seeds.

In this chapter, we set the GP parameters in EvadeML as the same as in the experiments by Xu et al. [126]. The population size in each generation is 48. The maximum number of generations is 20. The mutation rate for each PDF object is 0.1. The mutation traces that lead to successful evasion and promising variants are stored and applied in our experiments. The fitness threshold of a classifier is 0. We use the same external benign PDF files as Xu et al. [126] to provide ingredients for insertion and swap operations.

4.3.4 Feature-Space Evasion Model

In typical realizable attacks, including EvadeML, a consideration is not merely to move to the benign side of the classifier decision boundary, but to appear as benign as possible. This naturally translates into the following multi-objective optimization in feature space:

$$\underset{x}{\text{minimize}} \quad Q(x) = f(x) + \lambda c(x_M, x), \tag{4.2}$$

where $f(x)$ is the score of a feature vector x, with the actual classifier (such as SVM) $g(x) = \text{sgn}(f(x))$, x_M the malicious seed, x an evasion instance, $c(x_M, x)$ the cost of transforming x_M into x, and λ a parameter which determines the feature transformation cost. We use l_2 norm distance between x_M and x as the cost function: $c(x_M, x) = \sum_i |x_i - x_{M,i}|^2$. Since in most of our experiments features are binary, the choice of l_2 norm (as opposed to another l_p norm) is not critical.

As the optimization problem in Eq. (4.2) is non-convex and variables are binary in three of the four cases we consider, we use a stochastic local search method designed for combinatorial search domains, *Coordinate Greedy* (alternatively known as iterative improvement), to compute a local optimum (the binary nature of the features is why we eschew gradient-based approaches) [43, 58]. In this method, we optimize one randomly chosen coordinate of the feature vector at a time, until a local optimum is reached. To improve the quality of the resulting solution, we repeat this process from several random starting points. This approach has been shown to be extremely effective for computing evasion instances in binary domains [58].

4.3.5 Datasets

The dataset we use is from the *Contagio Archive*[3]. We use 5,586 malicious and 4,476 benign PDF files for training, and another 5,276 malicious and 4,459 benign files as the non-adversarial test dataset. The training and test datasets also contain 500 seeds selected by Xu et al. [126], with 400 in the training data and 100 in the test dataset. These seeds are filtered from 10,980 PDF malware samples and are suitable for evaluation since they are detected with reliable malware signatures by the Cuckoo sandbox [35]. We randomly select

[3]Available at the following URL: `http://contagiodump.blogspot.com/2013/03/16800-clean-and-11960-malicious-files.html`.

40 seeds from the training data as the retraining seeds and use the 100 seeds in the test data as the test seeds.

4.3.6 Implementation of Iterative Adversarial Retraining

We made a small modification to the general iterative retraining approach described in Section 4.2 when it uses EvadeML as the realizable attack $O(h; \mathsf{D})$. Specifically, we used only 40 malicious seeds to EvadeML to generate evasions, to reduce running time and make the experiment more consistent with realistic settings where a large proportion of malicious data is not adapting to the classifier. As shown below, this set of 40 instances was sufficient to generate a model robust to evasions from held out 100 malicious seed PDFs.

We distribute both retraining and adversarial test tasks on two servers (Intel(R) Xeon(R) CPU E5-2695 v4 @ 2.10GHz, 18 cores and 64 GB memory, running Ubuntu 16.04). For retraining using EvadeML as the attack, we assign each server 20 seeds; each seed is processed by EvadeML to produce the adversarial evasion instances. We then add the 40 examples obtained to the training data, retrain the classifier, and then split the seeds between the two servers in the next iteration. In the evaluation phase, we assign each server 50 seeds from the 100 test instances, and each seed is further used to evade the classifier by using EvadeML.

4.3.7 Evaluation Metrics

We evaluate performance in two ways: 1) evaluation of evasion robustness (which is central to our specific inquiry), and 2) traditional evaluation. To evaluate robustness, we compute the proportion of 100 malicious test seed PDFs for which EvadeML successfully evades the classifier; this is our metric of *evasion robustness*, evaluated with respect to EvadeML. Thus, evasion robustness of 0% means that the classifier is successfully evaded in every instance, while

evasion robustness of 100% means that evasion fails every time. Our traditional evaluation metric uses test data of malicious and benign PDFs, where no evasions are attempted. On this data, we compute the ROC (receiver operating characteristic) curve and the corresponding AUC (area under the curve).

4.4 Efficacy of Feature-Space Attack Models

We now undertake our first task: evaluation of the effectiveness of robust ML obtained by using the abstract feature-space models of attack. We compare to a baseline classifier obtained by retraining with the most potent attack on our menu, EvadeML (which, in addition to inserting content, as done by other attacks [46, 72, 103], also allows the attacker to delete and swap PDF objects). We can think of our baseline as assuming that the defender knows that EvadeML is employed by the attacker, along with its hyperparameters. Throughout this and next section, we also use EvadeML to evaluate the effectiveness of classifiers hardened using a feature-space model, in comparison with the above baseline.

4.4.1 Structure-Based PDF Malware Classification

Our first case study uses a state-of-the-art PDF malware classifier which engineers features based on PDF *structure*. Indeed, we evaluate two versions of this classifier: an earlier version, which we call *SL2013*, and a more recent version, which we call *Hidost*. The experiments by Xu et al. [126] demonstrate that SL2013 can be successfully evaded. Since Hidost was a recent redesign attempting in part to address its vulnerability to mimicry attacks by significantly reducing the feature space, no data exists on its vulnerability to evasion attacks. Below we demonstrate that Hidost is also vulnerable to evasion attacks (indeed, more so than SL2013).

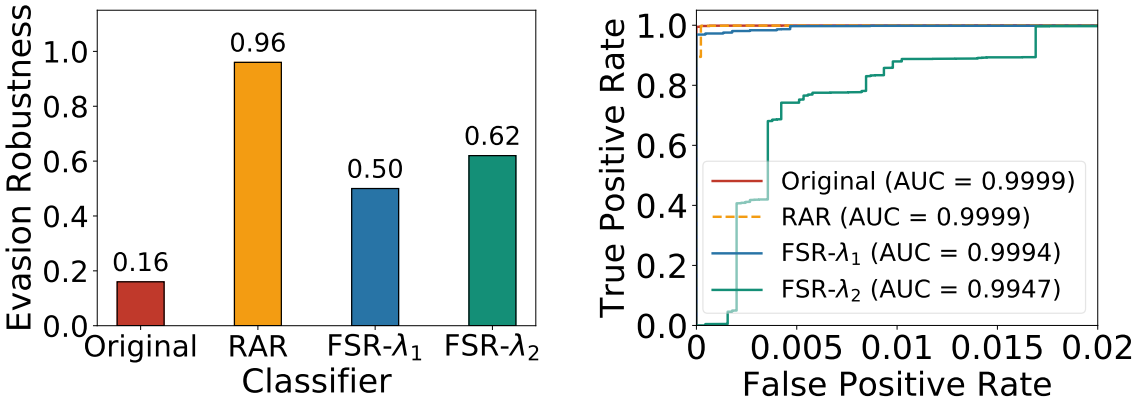

Figure 4.2: Evasion robustness under EvadeML test (left) and performance on non-adversarial data (right) of different classifiers for SL2013.

From the perspective of defense, we show that it is possible to harden both SL2013 and Hidost against a powerful realizable EvadeML attack by simply retraining with this attack (*RAR*, for *realizable-attack retraining*, henceforth refers to a model hardened using EvadeML). This serves as a baseline we use to evaluate the efficacy of a retraining defense with a feature-space attack model (henceforth, FSR for *feature-space retraining*). We then show that for both SL2013 and Hidost, FSR significantly underperforms RAR.

In our experiments, we empirically set the *RBF* parameters for training both SL2013 and Hidost to $C = 12$ and $\gamma = 0.0025$.

SL2013. We start our case study with SL2013. We first study adversarial retraining with realizable attacks, we then proceed to evaluate feature-space retraining.

Retraining with a Powerful Realizable Attack First, we replicated the EvadeML attack on the original SL2013; the classifier achieves only a 16% evasion robustness.[4] Next, to create a baseline, we conduct experiments in which EvadeML is employed to retrain SL2013. The

[4]This result differs from the experiments in [126] which show a 0% evasion robustness. We found a flaw in the implementation of feature extraction in EvadeML which causes evaluation to be performed using the wrong feature vectors. This bug has been fixed in the GitHub version of EvadeML.

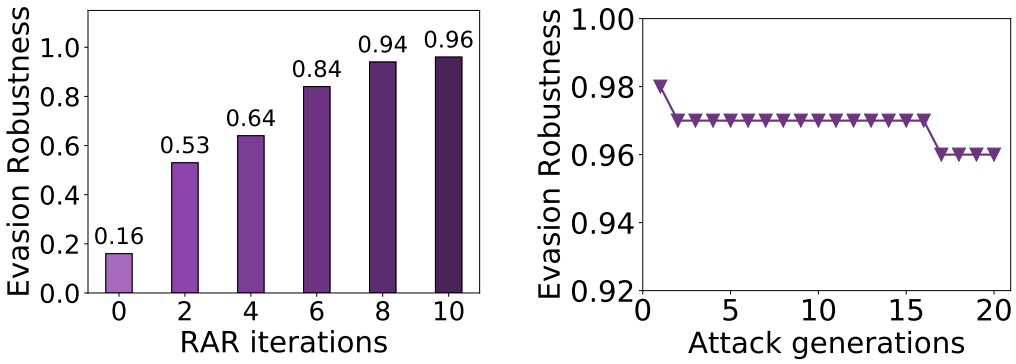

Figure 4.3: Evasion robustness with retraining iterations (left) and generations of the EvadeML attack test (right).

process terminated after 10 iterations at which point no evasive variants of the 40 retraining seeds could be generated. We observe (Figure 4.2 (left)) that the retrained classifier (RAR) obtained by this approach achieves a 96% evasion robustness. Moreover, RAR is essentially as accurate as the baseline SL2013 on non-adversarial data (Figure 4.2 (right)). Thus, it is clearly possible to be highly robust to this evasion attack without significantly compromising effectiveness on data not featuring explicit evasion attacks.

Figure 4.3 (left) shows the gradual improvement of evasion robustness over the 10 retraining iterations. This plot demonstrates non-trivial effectiveness of EvadeML: the first few iterations are clearly insufficient, as re-running EvadeML creates many new evasions that cannot be correctly detected by the classifier. Only after 6 iterations does EvadeML optimization loop begin to show significant signs of failing. Figure 4.3 (right) shows how increasing the number of generations in EvadeML attacks affects robustness of the RAR classifier. At this point, we can see that increasing the capability of the attack has minimal impact.

Feature-Space Retraining. Next, we experimentally evaluate the effectiveness of retraining with a feature-space model of evasion attacks in obtaining robust ML in the face of the

EvadeML realizable attack. We consider the setting with $\lambda = 0.05$ and $\lambda = 0.005$ in Eq. (4.2) (henceforth, FSR-λ_1 and FSR-λ_2).

The robustness results are shown in Figure 4.2 (left). Compared to the SL2013 baseline, feature-space retraining (FSR) boosts evasion robustness from 16% to 62%. Crucially, *the robustness of the resulting classifier is far below the classifier achieved by RAR.* This illustrates that defense that relies on feature-space models of adversarial examples may not in fact lead to robustness when it is faced with a real attack.

We again consider performance of FSR classifier on non-adversarial test data (Figure 4.2 (right)). We can see that robustness boosting again does not much degrade performance, with AUC remaining above 99%. However, we do see a substantial degradation as we move from $\lambda = 0.05$ to 0.005; thus, as we increase adversarial power in the feature-space model, while we do obtain a slightly more robust model, we incur a nontrivial hit in performance on non-adversarial data.

Hidost. We now repeat our experiments above with another structure-based classifier, Hidost. We set the retraining parameter $\lambda = 0.005$, which appears to strike a reasonable balance between robustness and accuracy on non-adversarial data. As before, we first evaluated the robustness of the original Hidost [104] by EvadeML. The result shows a 2% robustness—remarkably, significantly worse than SL2013.

Evasion robustness of Hidost, as well as improvements achieved by RAR and FSR, are shown in Figure 4.4 (left), and the results are consistent with our observations for SL2013. First, by retraining with the realizable attack, evasion robustness is boosted to 98%, a rather dramatic improvement, and clear demonstration that successful defense is possible. In contrast, FSR achieves a 70% evasion robustness, a significant boost over the original Hidost, to be sure, but far below the evasion robustness of RAR.

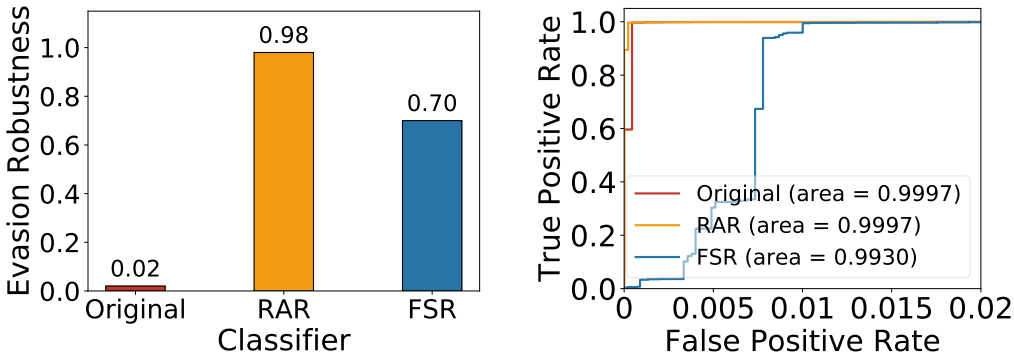

Figure 4.4: Evasion robustness under EvadeML test (left) and performance on non-adversarial data (right) of different classifiers for Hidost.

Evaluating these classifiers on non-adversarial test data in terms of ROC curves (Figure 4.4 (right)), we can observe that RAR achieves comparable accuracy ($> 99.9\%$ AUC) with the original Hidost classifier on non-adversarial data, and provides even better *True Positive Rate (TPR)* when *False Positive Rate (FPR)* is close to zero. On the other hand, FSR achieves $> 99\%$ AUC, but yields a significant degradation of TPR when FPR< 0.01.

4.4.2 Content-Based PDF Malware Classification

Our next case study concerns another two PDF malware classifiers which use features based on PDF file content, rather than logical structure. We trained both real-valued and binarized PDFRate (henceforth, PDFRate-R and PDFRate-B) on the same dataset as SL2013 and Hidost, and achieved $> 99.9\%$ AUC for both classifiers on test data. In our experiments, we empirically set the SVM *RBF* parameters for training to $C = 10$ and $\gamma = 0.01$. In our evaluation of ML robustness, we again set the feature-space model parameter λ to be 0.005.

PDFRate with Real-Valued Features. We begin with the variant of PDFRate—PDFRate-R—which has been constructed in previous evaluations and shown comparable in performance to the original implementation [103]. We again begin by replicating the EvadeML evasion

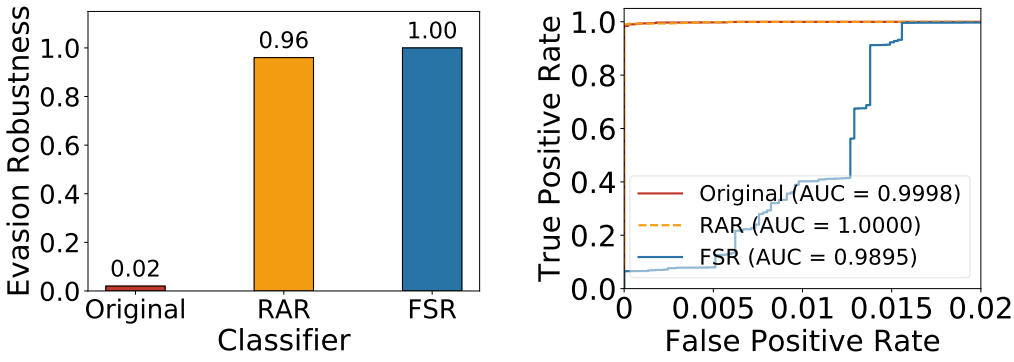

Figure 4.5: Evasion robustness under EvadeML test (left) and performance on non-adversarial data (right) of different classifiers for PDFRate-R.

robustness evaluation of the baseline classifier. As expected, we find the classifier quite vulnerable, with only 2% evasion robustness.

Next, we retrain PDFRate-R with EvadeML for 10 iterations (RAR baseline), and perform feature-space retraining using the conventional feature space model above. Our results are shown in Figure 4.5 (left). Observe that while RAR indeed achieves a highly robust classifier (96% robustness), FSR actually performs *even better*, with 100% robustness.

Comparing RAR and FSR performance on non-adversarial data (Figure 4.5 (right)), we observe that the high robustness of FSR does incur a cost: while RAR remains exceptionally effective (>99.99% AUC), FSR achieves AUC slightly lower than 99%, although most significantly, the degradation is rather pronounced for low FPR regions (below 0.015).

PDFRate with Binarized Features. One of our great surprises is the robustness of the binarized PDFRate: despite the fact that the real-valued PDFRate is quite vulnerable, *the same classifier using binary features was 100% robust to EvadeML* (Figure 4.6 (left)). Consequently, this will serve as our robust baseline (equivalently, RAR would terminate with no iterations). Feature-space retrained PDFRate-B also exhibits 100% evasion robustness, although it does require a number of iterations to converge.

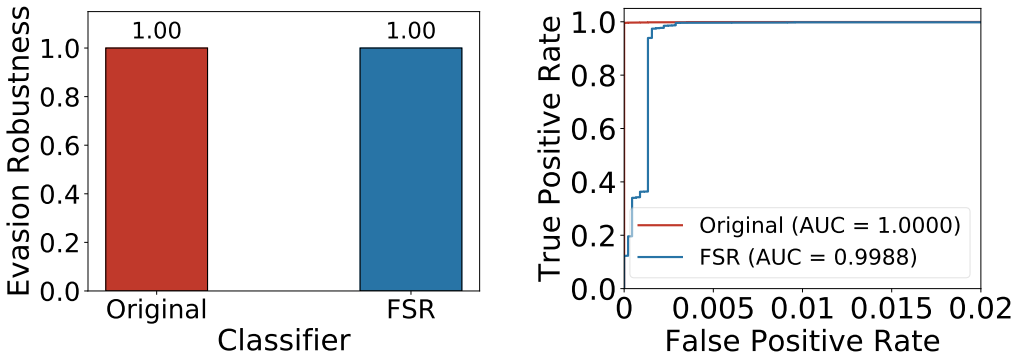

Figure 4.6: Evasion robustness under EvadeML test (left) and performance on non-adversarial data (right) of different classifiers for PDFRate-B.

Considering now the performance of PDFRate-B and FSR on non-adversarial test data (Figure 4.6 (right)), we can make two interesting observations. First, the baseline PDFRate-B is remarkably good even on this data; in a sense, it appears to hit the sweet spot of adversarial robustness and non-adversarial performance. Second, FSR retrained classifier is competitive in terms of AUC ($\sim 99.9\%$), but is observably worse than the baseline classifier for very low false positive rates.

4.4.3 Discussion

While we observed some value of using feature-space methods for boosting classifier robustness to evasion attacks, it falls far short of the robustness we know we can achieve by using a realizable attack oracle. Despite the advantages—in some cases substantial—in terms of running time, our results suggest that relying on feature space models may not be satisfactory in practice. There are two general reasons for the observed gap. First, synthetically generated adversarial instances may in actuality not preserve malicious functionality, since they need not abide by the system-level attack constraints. This would introduce noise and potentially bias into the retraining process. Second, realistic adversarial instances may not be generated, as they do not possess a sufficiently high objective value according to Eq. ((4.2)), in part

because better solutions according to this evasion model may not abide by realistic attack constraints.

4.5 Conclusion

We undertook an extensive exploration of the extent to which robust ML that uses the conventional feature-space models of evasion attacks remains robust to "real" attacks that can be implemented in actual malware and preserve malicious functionality (what we call realizable attacks). Our intriguing observation is that defense based on feature-space models can fail to achieve satisfactory robustness. This in itself raises some doubts about the nearly universal focus on such models as a means for ML defense, and suggests that practical usefulness of such approaches cannot be taken for granted. However, we also show that changing the nature of the feature space can make a difference: robust ML with feature-space models is quite robust in content-based detection (which uses content, rather than structural paths, as features).

It is natural to wonder how our approach and results are applied to other domains. In computer vision, the analog of realizable malware attacks are physical attacks, whereby the physical environment is modified, rather than the digital object, such as an image. Here, the corresponding foundational question is whether common robust ML methods based on small-l_p attacks successfully protect against physical attacks. The notion of conserved features can also be seen as more generally applicable. For example, in a bag-of-words representation for spam filtering, these could correspond to the existence of URL or file attachments, and in SQL injection attacks, these may refer to the existence of specific SQL commands, such as `Select`.

The main limitation of our study is in the specific choices we had to make to ensure that it is tractable. We chose a particular defensive paradigm—iterative retraining. As we have argued, it is the only paradigm that can fit every case that we investigate; for example, there is no other general approach for learning a robust SVM with non-linear kernels. However, it is possible that approaches based on robust optimization, if they were developed, can improve performance by taking advantage of the special structure of this problem. We implemented a particular class of feature-space attacks, using l_2 norm to measure the attacker's cost of feature manipulations, and stochastic local search to compute evasions. It is possible that better attack algorithms for generating attacks over binary domains will be developed, and, indeed, some alternatives exist. However, prior work suggests that this approach yields attacks that are close to optimal [58], with the use of random restarts playing a crucial role. Finally, our study was specific to PDF malware detection. However, our framework is quite general, and could be used in the future to consider other similar questions, such as the effectiveness of robust deep learning against physical attacks.

Chapter 5

Defending against Realizable Attacks in PDF Malware Detection

In Chapter 4, we have shown that robust ML that uses feature-space attack models can fail to defend against realizable attacks in the context of PDF malware detection. In spite of the above limitations, the feature space methods have tremendous appeal due to their relative speed and amenability to analysis. In this chapter, we present and evaluate a surprisingly simple "fix" to enable feature-space model not merely to become competitive with problem space approaches, but to actually exhibit better robustness. Specifically, we show that augmenting the feature space models with *conserved* features (those that cannot be unilaterally modified without compromising malicious functionality) significantly improves performance. Moreover, we show that feature space models enable generalized robustness when faced with a variety of realizable attacks, as compared to classifiers which are tuned to be robust to a specific realizable attack.

5.1 Overview

Thus far, we had observed that ML hardened with the standard mathematically convenient feature-space evasion attack model may in some cases not yield satisfactory robustness against real attacks. The key issue is that feature-space models are entirely disembodied from the domain. This is crucial to enable us to have mathematical formulations of attacks, but clearly has limitations. The key question is whether we can devise a simple way of anchoring feature-space attacks in the application domain to allow us to meaningfully and minimally constrain abstract attacks to reflect some of the constraints that real attacks face. Next, we propose a refinement of the feature-space model that aims to do just that.

Specifically, in this chapter we introduce the idea of *conserved features*, which we define to be *features, the unilateral modification of which compromises malicious functionality.* We develop this idea specifically for *binary features*, as this notion is particularly crisp in such a case (e.g., such features tend to correspond to the existence of particular objects in PDF).

To develop intuition about the nature of conserved features, consider SL2013, which employs structural paths as features to discriminate between malicious and benign PDFs. On the one hand, the structural paths like `/Type` are unessential to preserve malicious behaviors, and we do not expect them to be conserved. On the other hand, as the shellcode which triggers malicious functionality is embedded in certain PDF objects, those corresponding structural paths are likely to be conserved in each variant crafted from the same malicious seed (e.g., `/OpenAction/JS`). In addition, structural paths that facilitate embedded script in PDF files also can be conserved features as removing them can break the script (e.g., `/Names` and `/Pages`). This further illustrates that conserved features are not necessarily optimal for statistically distinguishing benign and malicious instances (indeed, these may be

common to both); rather, they serve to anchor the feature-space attack model in the domain by connecting features to malicious functionality.

Our first contribution is a method for boosting robustness of feature-space models without compromising their mathematical convenience (crucial for most approaches for robust ML). The key idea is to identify *conserved features*, that is, features that cannot be unilaterally modified without compromising malicious functionality. We exhibit such features in our setting, show that they cannot be identified with traditional statistical methods, and develop an algorithm for automatically extracting them. Finally, we show that by simply constraining that these features remain unmodified in adversarial training, feature-space approaches become effective even for robust structure-based PDF malware detection.

Our second contribution is to explore the extent to which ML robustness is *generalizable* to multiple *distinct* realizable attacks. Specifically, we expose both a robust classifier that was retrained by using a realizable attack (EvadeML), and a model hardened using a feature-space attack (accounting for conserved features), to a series of realizable attacks. Our results reveal a stark difference between the two: ML models hardened using EvadeML are quite fragile; in contrast, ML models hardened using feature-space attacks exhibit uniformly high robustness to the other attacks. Remarkably, we demonstrate that ML models hardened using feature-space attacks remain robust *even against realizable attacks that defeat conserved features*.

5.2 Identifying Conserved Features

We now describe a systematic automated procedure for identifying conserved features in the context of PDF malware detection. We first introduce how to identify conserved features of

SL2013 [102], and then describe how to generalize the approach to extract conserved features of other classifiers which are employed in our case study in chapter 4.

The key to identifying the conserved features of a malicious PDF is to discriminate them from non-conserved ones. Since merely applying statistical approaches on training data is insufficient to discriminate between these two classes of features, as demonstrated above, we need a qualitatively different approach which relies on the nature of evasions (as implemented in EvadeML [126]) and the sandbox (which determines whether malicious functionality is preserved) to identify features that are conserved.

We use a modified version of pdfrw [74][5] to parse the objects of PDF file and repack them to produce a new PDF file. We use Cuckoo [35] as the sandbox to evaluate malicious functionality. In the discussion below, we define x_i to be the malicious file, S_i the conserved feature set of x_i, and O_i the set of its non-conserved features. Initially, $\mathsf{S}_i = \mathsf{O}_i = \emptyset$.

At the high level, our first step is to sequentially delete each object of a malicious file and eliminate non-conserved features by evaluating the existence of a malware signature in a sandbox for each resulting PDF, which provides a preliminary set of conserved features. Then, we replace the object of each corresponding structural path in the resulting preliminary set with an external benign object and assess the corresponding functionality, which allows us to further prune non-conserved features. Next, we describe these procedures in detail.

5.2.1 Structural Path Deletion

In the first step, we filter out non-conserved features by deleting each object and its corresponding structural path, and then checking whether this eliminates malicious functionality (and should therefore be conserved). First, we obtain all the structural paths (objects) by

[5]The modified version is available at `https://github.com/mzweilin/pdfrw`.

parsing a PDF file. These objects are organized as a tree-topology and are sequentially deleted. Each time an object is removed, we produce a resulting PDF file by repacking the remaining objects. Then, we employ the sandbox to detect malicious functionality of the PDF after the object deletion. If any malware signature is captured, the corresponding structural path of the object is deleted as a non-conserved feature, and added to O_i. On the other hand, if no malware signature is detected, the corresponding feature is added in S_i as a *possibly* conserved feature.

One important challenge in this process is that features are not necessarily independent. Thus, in addition to identifying S_i and O_i, we explore *interdependence* between features by deleting objects. As the logic structure of a PDF file is with a tree-topology, the presence of some structural path depends on the presence of other structural paths whose object refers to the object of the prior one. We define that a structural path is a dependent of another if unilateral deleting the object associated with the latter causes a flip from 1 to 0 on the feature value of the former. For any feature j of x_i, we denote the set of features that depend on j by D_i^j. Note that for a given structural path (feature), there could be multiple corresponding PDF objects. In such a case, these objects are deleted simultaneously, so as the corresponding feature value is shifted from 1 to 0.

5.2.2 Structural Path Replacement

In the second step, we subtract the remaining non-conserved features in the preliminary S_i and move them to O_i. Similar to the prior step, we first obtain all the structural paths and objects of the malicious PDF file. Then for each object of the PDF that is in S_i, we replace it with an external object from a benign PDF file and produce the resulting PDF, which is further evaluated in the sandbox. If the sandbox detects any malware signature, then the corresponding structural path of the object replaced is moved from S_i to O_i. Otherwise, the

structural path is a conserved feature since both deletion and replacement of the corresponding object removes the malicious functionality of the PDF file. Note that in the case of multiple corresponding and identical objects of a structural path, all of these objects are replaced simultaneously.

After structural path deletion and replacement, for each malicious PDF file x_i, we can get its conserved feature set S_i, non-conserved feature set O_i, and dependent feature set D_j for any feature $j \in \mathsf{S}_i \cup \mathsf{O}_i$, which could be further leveraged to design evasion-robust classifiers.

5.2.3 Obtaining a Uniform Conserved Feature Set

The systematic approach discussed above provides a conserved feature set for each malicious seed to retrain a classifier. Our goal, however, is to identify a single set of conserved features which is *independent* of the specific malicious PDF seed file. We now develop an approach for transforming a collection of S_i, O_i, and D_i^j for a set of malicious seeds i into a *uniform* set of conserved features.

Obtaining a uniform set of conserved features faces two challenges: 1) minimizing conflicts among different conserved features, as a conserved feature for one malicious instance could be a non-conserved feature for another, and 2) abiding by feature interdependence if a conserved feature should be further eliminated.

To address these challenges, we propose a *Forward Elimination* algorithm to compute the uniform conserved feature set for a set of malicious seeds $\{x_1, x_2, ..., x_n\}$, given the conserved feature sets, non-conserved feature sets and dependent sets for each seed. As Algorithm 3 shows, we first obtain a union of the conserved feature sets. Then, we explore the contradiction of each feature in the union with the others, by comparing the total number of the feature being selected as a non-conserved feature and conserved feature. If the former one is greater

Algorithm 3 Forward Elimination for uniform conserved feature set.

Input:
The set of conserved features for $x_i (i \in [1, n])$, S_i;
The set of non-conserved features for $x_i (i \in [1, n])$, O_i;
The set of dependent features for $j \in \mathsf{S}_i \cup \mathsf{O}_i$, D_i^j;
Output:
The uniform conserved feature set for $\{x_1, x_2, ..., x_n\}$, S;

$\mathsf{S} \leftarrow \bigcup_{i=1}^{n} \mathsf{S}_i$;
$\mathsf{S}' \leftarrow \mathsf{S}$;
$\mathsf{Q} \leftarrow \emptyset$;
$\mathsf{D}^j = \bigcup_{i=1}^{n} \mathsf{D}_i^j$;
for each $j \in \mathsf{S}'$ **do**
  if $j \notin \mathsf{Q}$ **then**
    if $\sum_{i=1}^{n} \mathbb{1}_{j \in \mathsf{O}_i} \geq \beta \cdot \sum_{i=1}^{n} \mathbb{1}_{j \in \mathsf{S}_i}$ **then**
      $\mathsf{S} \leftarrow \mathsf{S} \setminus (\{j\} \cup \mathsf{D}^j)$;
      $\mathsf{Q} \leftarrow \mathsf{Q} \cup (\{j\} \cup \mathsf{D}^j)$;
    end if
  end if
end for
return S;

than β times the latter one, then this feature, together with its dependents, are eliminated from the union. Otherwise, the feature is added to the uniform feature set. We use β as a parameter to adjust the balance between conserved and non-conserved features. Typically, $\beta > 1$ as we are inclined to preserve malicious functionality associated with a conserved feature, even it could be a non-conserved feature of another PDF file. We set $\beta = 3$ in our experiments.

5.2.4 Identifying Conserved Features for Other Classifiers

Once we obtain conserved features of SL2013 for each malicious seeds, we can employ these features to identify conserved features for other classifiers using binary features. As our approach relies on the existence of malicious functionality and corresponding features, such

a relation is not obvious for real-valued features; we therefore leave the question of how to define and identify conserved features in real space for future work.

Hidost. Hidost and SL2013 are similar in nature in such a way that they employ structural paths as features. The only difference is that Hidost consolidates features of SL2013 as described in Chapter 4. Therefore, once the conserved features of SL2013 are identified, we can simply apply the *PDF structural path consolidation rules* described in Srndic and Laskov [104] to transform these features to the corresponding conserved features for Hidost.

Binarized PDFRate. We identify the conserved features for PDFRate-B by using the conserved feature set S_i of each seed x_i. For each x_i, we generate $|\mathsf{S}_i|$ PDF files, each of which corresponds to the PDF file when an element (structural path) in S_i is deleted. We then compare PDFRate-B features of these PDFs to the original x_i. If any feature value of x_i is flipped from 1 to 0, then this feature will be added in the conserved feature set of x_i for PDFRate-B. Afterward, we use Algorithm 3 to obtain the uniform conserved feature set. This approach can in fact be used for arbitrary PDF malware detectors over binary features (leveraging conserved structural paths identified using SL2013).

5.3 Classifying Using Only Conserved Features

We begin by exploring the effectiveness of using *only* conserved features for classification. We identified 8 conserved features for SL2013 (out of ∼6000), 7 for Hidost (out of ∼1000), and 4 for PDFRate-B (out of 135); these are detailed in Table 5.1.

We start by considering four natural questions pertaining to conserved features: 1) are they sufficient to make a classifier robust to evasions, 2) do they effectively discriminate between benign and malicious instances, 3) can they be identified using standard statistical methods

Table 5.1: Conserved features and their relevance to JavaScript.

Classifier	**Conserved features**	Involve JS?
SL2013	/Names	No
	/Names/JavaScript	Yes
	/Names/JavaScript/Names	Yes
	/Names/JavaScript/Names/JS	Yes
	/OpenAction	No
	/OpenAction/JS	Yes
	/OpenAction/S	No
	/Pages	No
Hidost	/Names	No
	/Names/JavaScript	Yes
	/Names/JavaScript/Names	Yes
	/Names/JavaScript/Names/JS	Yes
	/OpenAction	No
	/OpenAction/JS	Yes
	/Pages	No
PDFRate-B	count_box_other	No
	count_javascript	Yes
	count_js	Yes
	count_page	No

(such as sparse regularization), and 4) are they just detecting the presence of JavaScript in PDF?

We explore these for SL2013. Specifically, we trained a classifier using *only* the 8 conserved features (*CF* henceforth). As we can see in Figure 5.1 (left), this classifier is 100% robust to EvadeML attacks, appearing to resolve the first question. However, we emphasize that conserved features alone need not capture the full spectrum of adversarial behavior and constraints. Indeed, in Section 5.5 we show that classifiers based solely on conserved features can also be evaded, particularly if attacks are *specifically designed to evade them.* Rather, as we show presently, they provide a *sufficient anchoring* in the problem domain for feature-space attack models to succeed.

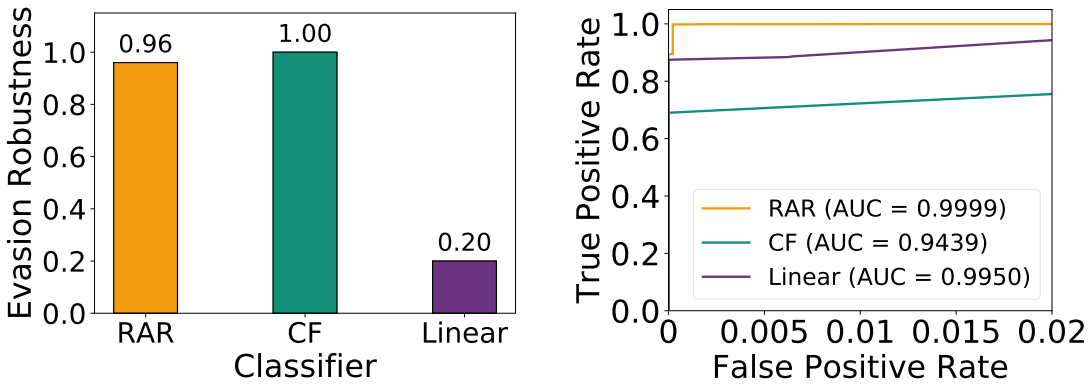

Figure 5.1: Classifying with conserved features: comparing evasion robustness (left) and ROC curves (right).

To address question (2), consider Figure 5.1 (right): clearly, if we desire a low false positive rate, using only conserved features for classification yields subpar performance on non-adversarial data.

To address the third question, we learn a linear SVM classifier for SL2013 with l_1 regularization (henceforth, Linear) where we empirically adjust the SVM parameter C to perform feature reduction until the number of the features is also 8; we find that only 3 of these are conserved features.[6] As we can see in Figure 5.1 (left), this classifier exhibits poor robustness; thus, statistical methods are insufficient to identify good conserved features.

To address the fourth question, we create a classifier using only one boolean feature which identifies the presence of JavaScript in a PDF file (henceforth, we refer to this feature as *JS*). We find that this classifier is also robust to EvadeML. On non-adversarial data, JS achieves FPR of 0.04 and FNR of 0.14 (in other words, 4% of the benign files in the non-adversarial dataset use JavaScript, while 14% of malicious instances use alternative attacks to

[6]The sparse versions of Hidost includes only 3 of the conserved features, while sparse PDFRate-B includes only 1. In another experiment, we adjusted C until all conserved features were selected. In this case, SL2013 requires 510 features, Hidost needs 154, and PDFRate-B needs 83.

Javascript).[7] To create an apples-to-apples comparison with the CF classifier, we empirically adjust the classification threshold of CF until we get the same FPR with JS. The resulting CF classifier exhibits FNR of 0.11, considerably better than JS. Nevertheless, it is clear that using either CF (only conserved features), or only JS, is impractical, since both FNR and FPR of these are quite high. Moreover, as we show in Section 5.5, classifiers based only on conserved features can be defeated by other realizable attacks. Next, we show that identification of conserved features is nevertheless crucial in creating highly effective feature-space attack models.

5.4 Feature-Space Model with Conserved Features

As discussed in chapter 4 , the feature-space evasion model in Eq. (4.2) may not sufficiently boost ML robustness. Since conserved features allow us to minimally tie the abstract feature-space representation to malicious functionality, we offer a natural modification of the model in Eq. (4.2), imposing the constraint that conserved features cannot be modified by the attacker. We formally capture this in the new optimization problem in Eq. (5.1), where S is the set of conserved features:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & Q(x) = f(x) + \lambda c(x_M, x), \\ \text{subject to} \quad & x_i = x_{M,i},\ \forall i \in \mathsf{S}. \end{aligned} \tag{5.1}$$

Other than this modification, we use the same *Coordinate Greedy* algorithm with random restarts as before to compute adversarial examples. We adopt the evasion model in Eq. (5.1) to retrain the target classifier using the retraining procedure from Section 4.3. We denote the classifier obtained by the retraining procedure using a feature-space model grounded by

[7] We observe similar results for 5,000 benign PDFs obtained by using Google web searches [103], where 3% of benign files use Javacript.

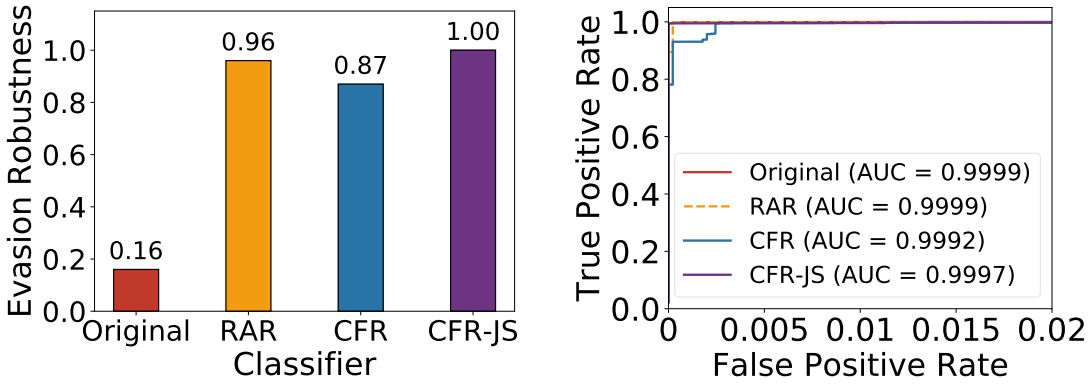

Figure 5.2: Evasion robustness (left) and performance on non-adversarial data (right) of different variants of SL2013.

conserved features by *CFR*. We also study the effectiveness of our automated procedure for identifying conserved features as compared to using a subset that only considers Javascript features (we can think of these as expert-identified conserved features, as this is what an expert would naturally consider). To this end, we repeat the procedure above by replacing the conserved feature set S in Eq. 5.1 with a subset that involves Javascript. The classifier resulting from such restricted adversarial retraining with "expert"-identified conserved features is termed *CFR-JS*.

5.4.1 SL2013

We now evaluate the robustness and effectiveness of the feature space retraining approach, which uses conserved features. We set the parameter $\lambda = 0.005$ as before. The robustness results are presented in Figure 5.2 (left). Observe that *CFR* now significantly improves robustness of the original classifier, with evasion robustness rising from 16% to 87%. Moreover, *CFR-JS* achieves a 100% evasion robustness against EvadeML. These results demonstrate that by leveraging the conserved features, the feature-space evasion models are now quite effective as a means to boost evasion robustness of SL2013.

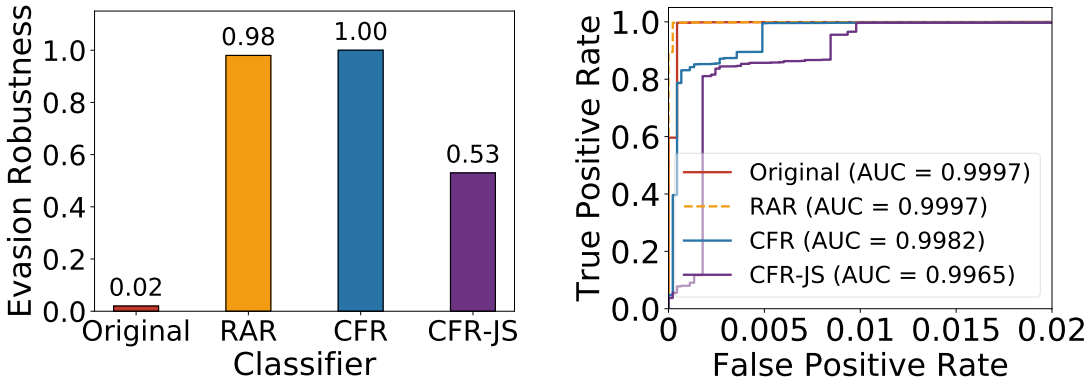

Figure 5.3: Evasion robustness (left) and performance on non-adversarial data (right) of different variants of Hidost.

In Figure 5.2 (right) we evaluate the quality of these classifiers on non-adversarial test data in terms of ROC curves. In all cases, be it original, RAR, CFR, and CFR-JS, AUC is $> 99.9\%$, although we can see a slight degradation of CFR for extremely low false positive rates compared to the others. It is noteworthy that CFR performs much better than FSR (robust ML using a standard feature-space approach, recall Figure 4.2 (right)).

5.4.2 Hidost

Next, we evaluate the effectiveness of CFR for Hidost. The results are shown in Figure 5.3 (left) and are largely consistent with SL2013. In particular, CFR boosts evasion robustness from 2% to 100% (slightly better than RAR), well above conventional FSR (recall Figure 4.4 (left)). In contrast, CFR-JS only boosts robustness to 53%, showing that our algorithmic approach can in some cases offer a considerable advantage to expert-chosen conserved features.

Evaluating the performance of CFR and CFR-JS on non-adversarial test data in terms of ROC curves in Figure 5.3 (right), we find that the CFR classifier can achieve $\sim 99.8\%$ AUC. This is somewhat worse than RAR, particularly for very low false positive rates, but

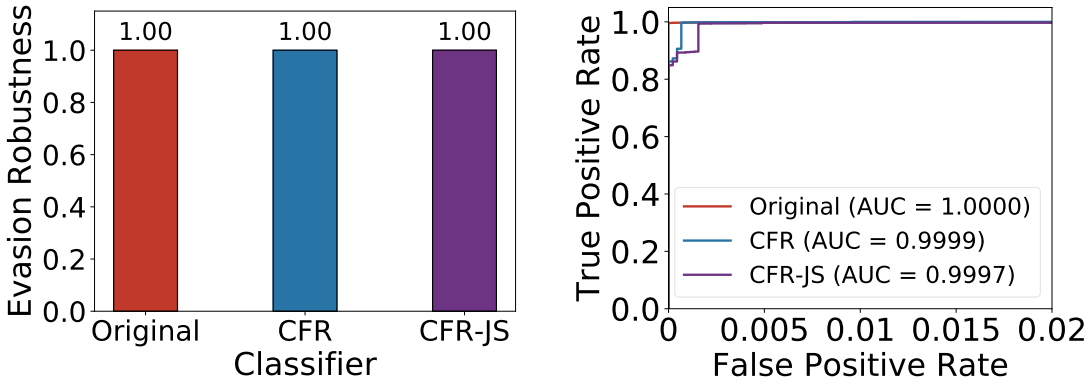

Figure 5.4: Evasion robustness (left) and performance on non-adversarial data (right) of different variants of PDFRate-B.

better than CFR-JS—again, in this case using the full batch of conserved features exhibits a significant advantage over solely looking for Javascript.

5.4.3 Binarized PDFRate

Finally, we evaluate the effectiveness of the CFR variants of PDFRate-B. We observe that both the *CFR* and *CFR-JS* classifiers in the PDFRate-B family achieve 100% evasion robustness against EvadeML (Figure 5.4 (left)), just as the RAR and FSR counterparts had.

However, a close look at Figure 5.4 (right) demonstrates that CFR and CFR-JS achieve far better performance on non-adversarial data, with >99.9% AUC, where improvements are particularly significant for small false positive rates compared to FSR (recall Figure 4.6 (right)). Moreover, in this experiment, CFR achieves slightly higher TPR than CFR-JS for low FPR regions (below 0.003). The main takeaway here is that although the feature-space approach already yields high robustness in this setting, introducing conserved features significantly mitigates its degradation in performance on non-adversarial data.

5.5 Additional Realizable Evasion Attacks

So far we used EvadeML [126] as the primary realizable attack in our experiments. This choice is defensible, as EvadeML explores a significantly larger attack space than many other evasion methods (e.g., Mimicry [103]), allowing deletions and swaps, in addition to insertions. Nevertheless, it is natural to wonder whether classifiers robust to EvadeML remain robust to other classes of evasion attacks. A particularly intriguing question is how the classifiers hardened against EvadeML fare in comparison with classifiers hardened against feature-space models, when faced with different realizable attacks.

To answer these questions, we consider *five* additional realizable attacks: *Mimicry* [103], which was one of the first realizable attacks on PDF malware detectors, *Mimicry+*, an enhanced variant of Mimicry, *MalGAN* [46], which uses Generative Adversarial Networks (GANs) to create evasion attacks (but only targets binary classifiers), *Reverse Mimicry* [72], which inserts malicious payloads into target benign files, and a new custom attack aimed at defeating PDFRate-B conserved features, and our *Custom Attack*, which targets a feature extraction bug in the Mimicus implementation of PDFRate in order to defeat the corresponding CF classifier. The Mimicry/Mimicy+ attacks are designed specifically for PDFRate, and cannot be usefully applied to SL2013 or Hidost, whereas the Reverse Mimicry attack and our custom attack require *zero knowledge* of target classifiers. In our experiments, W=we use the same 100 malicious seeds employed in Chapter 4 and Section 5.4 as attack files.

5.5.1 Mimicry and Mimicry+ Attacks

We start by considering the Mimicry [103] and Mimicry+ attacks for both real-valued and binarized variants of PDFRate.

Mimicry assumes that an attacker has full knowledge of the features employed by a target classifier. The mimicry attack then manipulates a malicious PDF file so that it mimics a particular selected benign PDF as much as possible. The implementation of Mimicry is simple and independent of any particular classification model.

Our mimicry attack uses the Mimicus [103] implementation, which was shown to successfully evade the PDFRate classifier. To improve its evasion effectiveness, Mimicus chooses 30 different target benign PDF files for each attack file. It then produces one instance in feature space for each target-attack pair by merging the malicious features with the benign ones. The feature space instance is then transformed into a PDF file using a *content injection approach*. The resulting 30 files are evaluated by the target classifier, and only the PDF with the best evasion result is selected, which was submitted to WEPAWET [19] to verify malicious functionality. To make Mimicry consistent with our framework, we employ the Cuckoo sandbox [35] in place of WEPAWET (which was in any case discontinued) to validate maliciousness of the resulting PDF file.

In addition to the original version of Mimicry, we implement an enhanced variation, *Mimicry+*, with two modifications. First, Mimicry+ chooses the 30 most benign PDF files predicted by the target classifier as target files (instead of randomly selecting those, as in Mimicry). Second, for each attack file, all the resulting 30 files are evaluated by the sandbox and only those verified to have malicious functionality are selected to evade the target classifier.

The results are shown in Figures 5.5 and 5.6, and offer two noteworthy findings. First, as can be seen in Figure 5.6, RAR classifiers (hardened specifically against EvadeML, recall that the original PDFRate-B classifier is equivalent to RAR) can be quite vulnerable to the Mimicry+ attack, whereas both FSR and CFR classifiers remain robust. Second, Mimicry+ is indeed a much stronger attack than Mimicry: the original Mimicry fails to significantly degrade

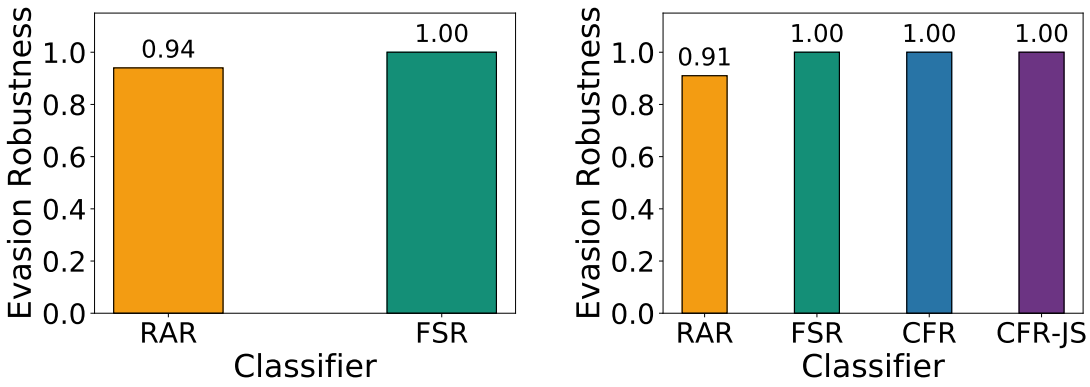

Figure 5.5: Robustness to Mimicry attack. Left: PDFRate-R (note that our notion of CFR is not applicable here). Right: PDFRate-B.

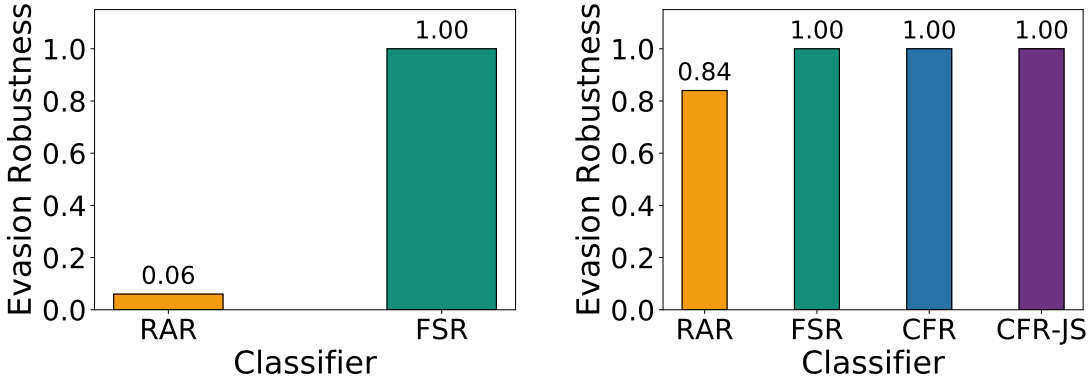

Figure 5.6: Robustness to Mimicry+ attack. Left: PDFRate-R (note that our notion of CFR is not applicable here). Right: PDFRate-B.

RAR performance, whereas Mimicry+ largely evades the RAR variant of PDFRate-R, and is somewhat more potent against PDFRate-B than Mimicry. This demonstrates that besides its mathematical elegance, the abstract feature-space evasion models, once appropriately anchored to the domain, are rather generally robust to evasion attacks.

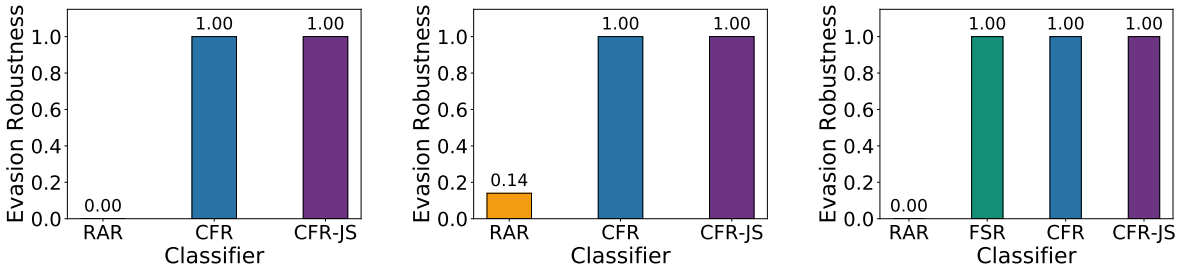

Figure 5.7: Robustness to MalGAN attack. SL2013 (left), Hidost (middle), PDFRate-B (right).

5.5.2 MalGAN Attack

Next, we consider the MalGAN attack [46] on the three classifiers over binary feature space we have previously studied: SL2013, Hidost, and PDFRate-B, with RAR and FSR/CFR versions that have been shown robust to EvadeML.

MalGAN is a Generative Adversarial Network [32] framework to generate malware examples which can evade a black-box malware detector with binary features. It assumes that an attacker knows the features, but has only black-box access to the detector decisions. MalGAN comprises three main components: a generator which transforms malware to its adversarial version, a black-box detector which returns detection results, and a substitute detector which is used to fit the black-box detector and train the generator. The generator and substitute detector are feed-forward neural networks which work together to evade the black-box detector. The results of [46] show that MalGAN is able to decrease the *True Positive Rate* on the generated examples from $> 90\%$ to 0%. We note that strictly speaking, MalGAN variants are not implemented as actual PDF files; however, we still treat it as a realizable attack since it only adds features to a malicious file, which can be implemented (at least in structure-based detection) by adding the associated objects into the PDF file.

The results, shown in Figure 5.7, demonstrate that despite EvadeML being a powerful attack, the RAR approaches which use it for hardening (with resulting classifiers no longer very vulnerable to EvadeML) are *highly* vulnerable to MalGAN, with evasion robustness of 0% in most cases. In contrast, CFR models which use conserved features remain highly robust (100% in all cases), just as we had observed earlier.

5.5.3 Reverse Mimicry Attack

Next, we employ the Reverse Mimicry attack [72] on the EvadeML-robust variants of all the classifier types (SL2013, Hidost, PDFRate-R, and PDFRate-B).

The Reverse Mimicry attack assumes that an attacker has zero knowledge of the target classifier. The basic idea is to inject malicious payloads into target benign files to minimize the structural difference between the resulting examples and targets. Our Reverse Mimicry attack employs the adversarial examples provided by Maiorca et al. [72] which was shown to successfully evade PDF classifiers based on structural analysis. Specifically, we use the 500 PDF files produced by injecting a malicious JavaScript code that does not contain references to other objects to target benign PDF files. We selected the 376 files out of 500 that display malicious behaviors detected by the Cuckoo sandbox.

Figure 5.8 presents the results, which are revealing in several ways. First, we again observe that RAR (hardened specifically against EvadeML) is roundly defeated in most cases. Second, consider the robustness results for the classifier using only the conserved features (CF), we can see that reverse mimicry succeeds in defeating conserved features for a non-trivial proportion of instances. It does so by including Javascript tags in structural paths that are not used as features by SL2013/Hidost (since these classifiers only consider commonly occurring sets of structural paths). Thus, this attack reveals an important vulnerability in the feature

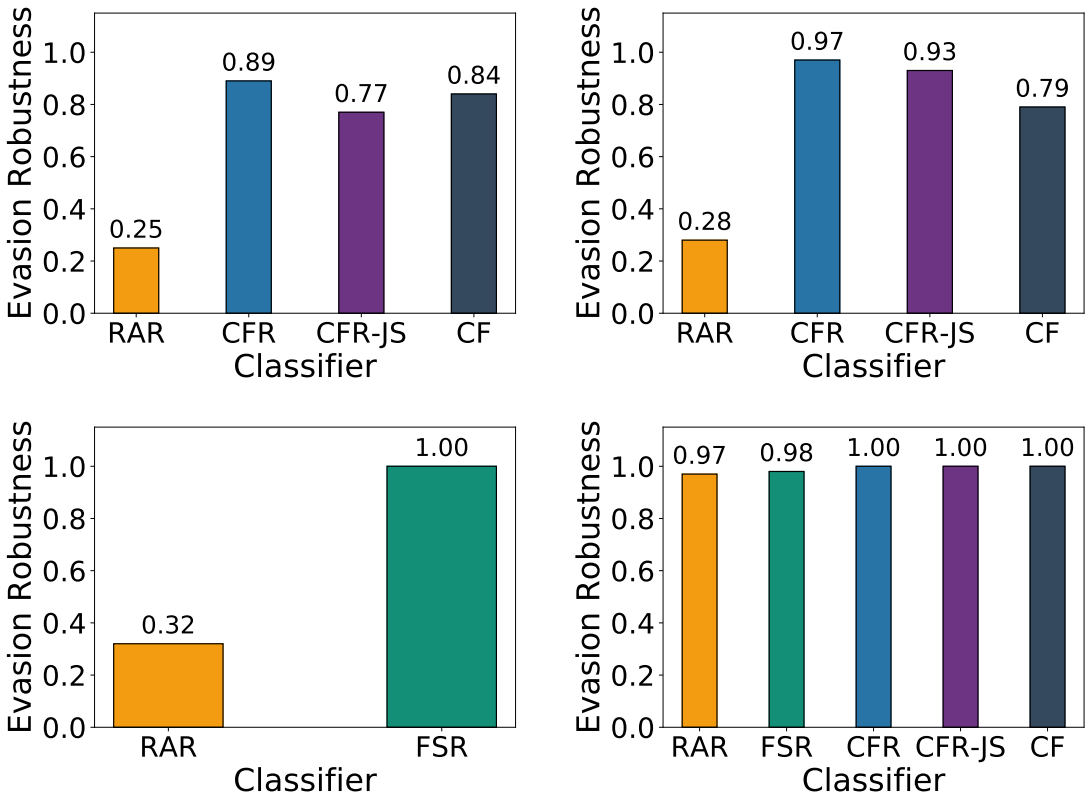

Figure 5.8: Robustness to Reverse Mimicry attack. SL2013 (top left), Hidost (top right), PDFRate-R (bottom left), PDFRate-B (bottom right). Note that our notions of CFR and CF for PDFRate-R is not applicable here.

extraction approach employed by these classifiers; indeed, it suggests that structure-based classifiers may be *inherently* difficult to harden. Remarkably, CFR remains more robust than CF despite these vulnerabilities. The case of Hidost is particularly stark: CFR is nearly 20% more robust than CF!

5.5.4 The Custom Attack

Our final custom attack exploits a feature extraction vulnerability in the Mimicus implementation of PDFRate. Normally, the characters used in the Name objects of a PDF file are limited to a specific set. Since PDF specification version 1.2, a lexical convention has

Table 5.2: Transformation of entry names in the custom attack.

Entry	Hexadecimal Representation
/Action	/#41#63#74#69#6f#6e
/Filter	/#46#69#6c#74#65#72
/Length	/#4c#65#6e#67#74#68
/JavaScript	/#4a#61#76#61#53#63#72#69#70#74
/JS	/#4a#53
/S	/#53
/Type	/#54#79#70#65

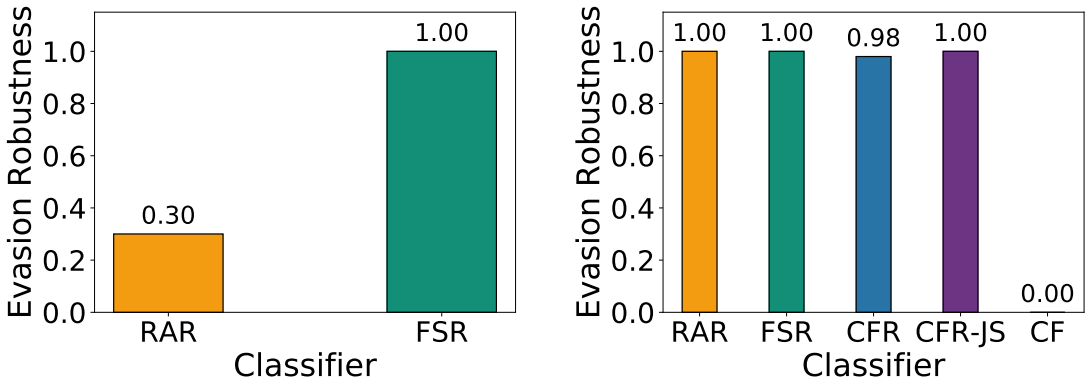

Figure 5.9: Robustness to the custom attack. Left: PDFRate-R (note that our notions of CFR and CF are not applicable here). Right: PDFRate-B.

been added to represent a character with its hexadecimal ANSI-code, e.g., #xx. Such a modification enables us to create an arbitrary string in the form of #xx#xx#xx. In our implementation, we replaced a set of entries in the attack PDF files with their hexadecimal representations (see Table 5.2). These features were selected with the goal to obfuscate tags crucial to the code execution in PDF, which are frequently used for feature extraction. With this technique, the scanner would not be able to detect malicious code without dynamically reconstructing the PDF structure. While it is theoretically possible to replace all the ASCII text inside the document, we chose not to do that due to the concern on the expansion of file size.

The results are shown in Figure 5.9. We find that after this attack, CF robustness is 0. We also observe that the robustness of RAR classifier for PDFRate-R also drops, although to 0.3 rather than 0. Significantly, the FSR classifiers for both PDFRate-R and PDFRate-B remain 100% robust, and the CFR variant of PDFRate-B has nearly perfect robustness (0.98) against this attack. Our latter observation is particularly remarkable: although the conserved features are roundly defeated by this attack, the use of these as a part of a holistic retraining approach yields a classifier that remains robust. Thus, not only is it possible to construct a robust malware classifier without unduly relying on conserved features, but we can accomplish this through iterative retraining in feature space.

5.6 Conclusion

In this chapter, we presented a refined version of the feature-space model that makes use of conserved features (which we can identify automatically), and showed that where feature-space defense previously failed, it now succeeds. Our finding may well be the most intriguing: feature-space approaches exhibit generalized robustness, in that the resulting robust ML (after appropriate refinement using conserved features) exhibits robustness to multiple realizable attacks. This contrasts with defense that is hardened using a *specific* realizable attack—even one quite powerful on the surface (EvadeML)—which can fail dramatically when faced with a different attack. These findings demonstrate the power of effective mathematical abstractions in security.

There are several limitations of our method that can offer further opportunities for future work. One example is the fact that we only define conserved features when these are binary; it may be that finding meaningful conserved features in continuous feature spaces is inherently more difficult. Another issue is the surprising finding that sufficient anchoring of feature-space

defense in the domain using conserved features allows us to achieve robustness, *even when conserved features can be circumvented.* It may be that conserved features are ultimately only a part of the solution, and only help if they adequately capture the attack surface in the abstract feature space. The extent to which small variations in the set of identified conserved features matters is also an open question: our evidence is mixed, with "expert"-defined features usually, but not always, sufficient for robustness.

Chapter 6

Defending against Non-Salient Adversarial Examples in Image Classification

In this chapter, we focus on robust learning in image classification. Despite the remarkable success of deep neural networks, significant concerns have emerged about their robustness to adversarial perturbations to inputs. While most attacks aim for being unsuspicious, there is no universal notion of what it means for adversarial examples. Here, we focus on salience as a way to capture whether perturbations are suspicious, and propose a general approach for defending against non-salient attacks. To capture cognitive salience, we split an image into foreground (salient region) and background (the rest), and allow significantly larger adversarial perturbations in the background, while ensuring that cognitive salience of background remains low. We describe how to compute the resulting non-salience-preserving

dual-perturbation attacks on classifiers. We then show that adversarial training with dual-perturbation attacks yields classifiers that are considerably more robust to such attacks, as well as to background perturbations, than state-of-the-art robust learning approaches, without compromising robustness to conventional attacks. In addition, our defense against non-salient attacks results in classifiers that align well with human perception.

6.1 Overview

An observation by [108] that state-of-the-art deep neural networks that exhibit exceptional performance in image classification are fragile in the face of small adversarial perturbations of inputs has received a great deal of attention. A series of approaches for designing adversarial examples followed [13, 33, 108], along with methods for defending against them [71, 87], and then new attacks that defeat prior defenses, and so on. Attacks can be roughly classified along three dimensions: 1) introducing small l_p-norm-bounded perturbations, with the goal of these being imperceptible to humans [71], 2) using non-l_p-based constraints that capture perceptibility (often called *semantic perturbations*) [6], and 3) modifying physical objects, such as stop signs [25], in a way that does not arouse suspicion. One of the most common motivations for the study of adversarial examples is safety and security, such as the potential for attackers to compromise the safety of autonomous vehicles that rely on computer vision [25]. However, while imperceptibility is certainly sufficient for perturbations to be unsuspicious, it is far from necessary, as physical attacks demonstrate. On the other hand, while there are numerous formal definitions that capture whether noise is perceptible [13, 82], what makes adversarial examples suspicious has been largely informal and subjective.

We propose a simple formalization of an important aspect of what makes adversarial perturbations unsuspicious based on the notion of cognitive salience [39, 55]. Specifically, we make

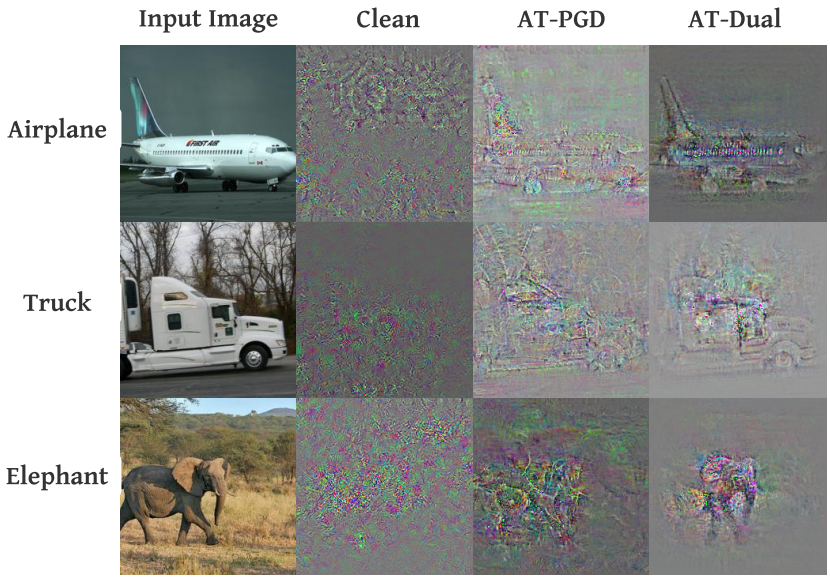

Figure 6.1: Visualization of loss gradient of different classifiers with respect to pixels of *non-adversarial* inputs. *Clean* is the model that takes no defense. *AT-PGD* denotes adversarial training using the PGD attack. *AT-Dual* is our proposed defense. It can be seen that *AT-Dual* aligns significantly better with human perception.

a distinction between image foreground and background based on how much attention a human viewer pays to the different parts of the captured scene. We capture this as a concrete threat model by allowing significantly more noise in the background (which has low salience) than the foreground (which has high salience). In effect, we posit that perturbations in the foreground, when visible, will arouse significantly more suspicion (by being cognitively more salient) than perturbations made in the background.

Our first contribution is a formal model of such *dual-perturbation attacks*, which is a generalization of the l_p-norm-bounded attack models that explicitly aims to ensure that the adversarial perturbation *does not make the background highly salient*. Second, we propose an algorithm for finding adversarial examples using this model, which is an adaptation of the PGD (projected gradient descent) attack [71]. Since our ultimate goal is robustness to non-salient adversarial examples, our third contribution is to embed the dual-perturbation

attacks in the adversarial training loop [71] instead of PGD or other conventional variants; we term the resulting defense *AT-Dual*. We then experimentally evaluate first our approach to designing non-salient adversarial examples, and then the effectiveness of this threat model in making classifiers robust to a broad class of non-salient adversarial example attacks. We show that the proposed approach indeed yields adversarial examples that do not significantly increase the salience of background, despite adding substantially more noise to it. Most significantly, we show that adversarial training which uses our dual-perturbation attack model yields significantly higher robustness to such non-salience-preserving attacks than state-of-the-art adversarial training alternative. Moreover, it maintains comparable robustness to conventional l_p-norm bounded attacks to state-of-the-art approaches. In addition, we show increased robustness of our defense to background perturbation. Finally, as shown in Figure 6.1 which visualizes the loss gradient of *AT-Dual* (proposed approach) with two alternative image classifiers, our approach results in models that align better with human perception than even adversarial training which uses PGD-based adversarial examples.

6.2 Dual-Perturbation Attacks

6.2.1 Motivation

Our threat model is motivated by the *feature integration theory* [111] in cognitive science: regions that have features that are different from their surroundings are more likely to catch a viewer's gaze. Such regions are called *salient regions*, or *foreground*, while the others are called *background*. Accordingly, for a given image, the semantics of the object of interest is more likely to be preserved in the foreground, as it catches more visual attention of a viewer compared to the background. If the foreground of an image is corrupted, then the semantics of the object of interest is broken. In contrast, the same extent of corruption in

Figure 6.2: Semantic distinction between foreground and background. Left: Original image of bears. Middle: Adversarial example with ℓ_∞ bounded perturbations ($\epsilon = 40/255$) on the background, the semantic meaning (bear) is preserved. Right: Adversarial example with ℓ_∞ bounded perturbations ($\epsilon = 40/255$) on the foreground, with more ambiguous semantics.

the background nevertheless preserves the overall semantic meaning of the scene captured (see, e.g., Figure 6.2).

Despite this important cognitive distinction between foreground and background, essentially all of the attacks on deep neural networks for image classification make no such distinction, even though a number of other semantic factors have been considered [6, 80]. Rather, much of the focus has been on adversarial perturbations that are *not noticeable* to a human, but which are applied equally *to the entire image*. However, in security applications, the important issue is not merely that an attack cannot be noticed, but that whatever observed is *not suspicious*. The main goal of the threat model we introduce next is therefore to capture more precisely the notion that an adversarial example is not suspicious by leveraging the cognitive distinction between foreground and background of an image.

6.2.2 Modeling Non-Salient Adversarial Examples

At the high level, our proposed threat model involves producing small (imperceptible) adversarial perturbations in the foreground of an image, and larger perturbations in the

background. This can be done by incorporating state-of-the-art attacks into our method: we can use one attack with small ϵ in the foreground, and another with a large ϵ in the background. Consequently, we term our approach *dual-perturbation attacks*. Note that these clearly generalize the standard small-norm (e.g., PGD) attacks, since we can set the ϵ to be identical in both the foreground and background. However, the key consideration is that after we add the large amount of noise to the background, *we must ensure that we do not thereby make it highly salient to the viewer*. We capture this second objective by including in the optimization problem a *salience* term that decreases with increasing salience of the background.

Formally, the *dual-perturbation* attack solves the following optimization problem:

$$\max_{||\boldsymbol{\delta}\circ\mathcal{F}(\boldsymbol{x})||_p\leq\epsilon_F, ||\boldsymbol{\delta}\circ\mathcal{B}(\boldsymbol{x})||_p\leq\epsilon_B} \mathcal{L}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta}), y\right) + \lambda\cdot\mathcal{S}\left(\boldsymbol{x}+\boldsymbol{\delta}\right), \tag{6.1}$$

where $\mathcal{S}\left(\boldsymbol{x}+\boldsymbol{\delta}\right)$ measure the non-salience of background after adversarial noise $\boldsymbol{\delta}$ has been added, with λ a parameter that explicitly balances the two objectives: maximizing predicted loss on adversarial examples, and limiting background salience so that the adversarial example produced is unsuspicious. Here $\mathcal{F}$ returns the mask matrix constraining the area of the perturbation in the foreground, and $\mathcal{B}$ returns the mask matrix restricting the area of the perturbation in the background, for an input image $\boldsymbol{x}$. $\mathcal{F}(\boldsymbol{x})$ and $\mathcal{B}(\boldsymbol{x})$ have the same dimension as $\boldsymbol{x}$ and contain 1s in the area which can be perturbed and 0s elsewhere. $\circ$ denotes element-wise multiplication for matrices. Hence, we have $\boldsymbol{x} = \mathcal{F}(\boldsymbol{x}) + \mathcal{B}(\boldsymbol{x})$ which indicates that any input image can be decomposed into two independent images: one containing just the foreground, and the other containing the background.

We consider two approaches for modeling the suspiciousness $\mathcal{S}(\boldsymbol{x}+\boldsymbol{\delta})$ of a perturbed image $\boldsymbol{x}+\boldsymbol{\delta}$. Our primary method is to make use of the variance of an image's *Laplacian* as the

measure of salience [89]. The Laplacian approach highlights regions of an image containing rapid intensity changes such as shape edges. Thus, a Laplacian with a larger variance indicates a more focused and sharper image, while a smaller variance suggests a smooth image that is less salient. We adopt this idea and measure background non-salience of a perturbed image by using its negative variance of the background area of its Laplacian, which is defined as

$$\mathcal{S}(\boldsymbol{x}+\boldsymbol{\delta}) = -\mathsf{Var}(\{\mathsf{Lap}(\mathsf{Gray}(\boldsymbol{x}+\boldsymbol{\delta}))_k | \mathcal{B}(\boldsymbol{x})_k \neq 0 \ \}), \tag{6.2}$$

where $\mathsf{Gray}(\cdot)$ converts an RGB image to grayscale, $\mathsf{Lap}(\cdot)$ is the Laplacian filter, and $\mathsf{Var}(\cdot)$ computes the population variance of a set. Our alternative approach leverages a recent study on human fixation prediction [55], which is detailed in Appendix B.1.

6.2.3 Identifying Foreground and Background

Given an input $\boldsymbol{x}$, we aim to compute $\mathcal{F}(\boldsymbol{x})$, the foreground mask and $\mathcal{B}(\boldsymbol{x})$, the background mask. We consider two approaches for this: fixation prediction and segmentation.

Our first method leverages the fixation prediction approach [55] to identify foreground and background. This enables a general approach for foreground-background partition as fixation predictions are not limited to any specific collection of objects. Specifically, we first use DeepGaze II [55] to output predicted pixel-level density of human fixations on an image. We then divide the image into foreground and background by setting a threshold $t = 0.5 \cdot (s_{min}(\boldsymbol{x}) + s_{max}(\boldsymbol{x}))$ for each input image $\boldsymbol{x}$ where (s_{min}, s_{max}) are the minimum and maximum values of human fixation on pixels of $\boldsymbol{x}$. Pixels with larger values than t are grouped into the foreground, and the others are identified as background subsequently.

Our second approach is to make use of semantic segmentation to provide a partition of the foreground and background in pixel level. This can be done in two steps: First, we use

state-of-the-art paradigms for semantic segmentation (e.g., [68]) to identify pixels that belong to each corresponding object, as there might be multiple objects in an image. Next, we identify the pixels that belong to the object of interest as the foreground pixels, and the others as background pixels.

We use both of the above approaches in dual-perturbation attacks when evaluating the robustness of classifiers, as well as designing robust models. Note that we use the same foreground and background partition for a non-adversarial image $\boldsymbol{x}$ and its perturbed version $\boldsymbol{x}+\boldsymbol{\delta}$. Specifically, we first use the above approaches to identify the foreground and background of $\boldsymbol{x}$; we then apply the same partition on $\boldsymbol{x}+\boldsymbol{\delta}$. We purposefully choose not to use the above deep-learning-based partition methods directly on $\boldsymbol{x}+\boldsymbol{\delta}$, as the corresponding neural models are trained on non-robust data, and they could lead to adversarial attacks on the partition.

6.2.4 Computing Dual-Perturbation Attacks

A natural approach for solving the optimization problem shown in Eq. (6.1) is to apply an iterative method, such as the PGD attack. However, the use of this approach poses two challenges in our setting. First, as in the PGD attack, the problem is non-convex, and PGD only converges to a local optimum. We can address this issue by using *random starts*, i.e., by randomly initializing the starting point of the adversarial perturbations, as in [71]. Second, and unlike PGD, the optimization problem in Eq. (6.1) involves *two hard constraints* $||\boldsymbol{\delta} \circ \mathcal{F}(\boldsymbol{x})||_p \leq \epsilon_F$ and $||\boldsymbol{\delta} \circ \mathcal{B}(\boldsymbol{x})||_p \leq \epsilon_B$. Thus, the feasible region of the adversarial perturbation $\boldsymbol{\delta}$ is not an ℓ_p ball, which makes computing the projection $\mathcal{P}_\epsilon$ computationally challenging in high-dimensional settings. To address this challenge, we split the *dual-perturbation* attack into two individual processes in each iteration, one for the adversarial perturbation in the foreground and the other for the background, and then merge these two perturbations when computing the gradients, like a standard PGD attack.

We use the following steps to solve the optimization problem of dual-perturbation attacks:

1. *Initialization.* Start with a random initial starting point $\boldsymbol{\delta}^{(0)}$. To do this, randomly sample a data point $\boldsymbol{\delta}_F^{(0)}$ in ℓ_p ball $\Delta(\epsilon_F)$ and $\boldsymbol{\delta}_B^{(0)}$ in $\Delta(\epsilon_B)$. Then, $\boldsymbol{\delta}^{(0)}$ can be obtained by using $\boldsymbol{\delta}^{(0)} = \boldsymbol{\delta}_F^{(0)} \circ \mathcal{F}(\boldsymbol{x}) + \boldsymbol{\delta}_B^{(0)} \circ \mathcal{B}(\boldsymbol{x})$. This ensures that the initial perturbation is feasible in both foreground and background.

2. *Split.* At the k-th iteration, split the perturbation $\boldsymbol{\delta}^{(k)}$ into $\boldsymbol{\delta}_F^{(k)}$ for foreground and $\boldsymbol{\delta}_B^{(k)}$ for background:

$$\begin{cases} \boldsymbol{\delta}_F^{(k)} = \boldsymbol{\delta}^{(k)} \circ \mathcal{F}(\boldsymbol{x}) \\ \boldsymbol{\delta}_B^{(k)} = \boldsymbol{\delta}^{(k)} \circ \mathcal{B}(\boldsymbol{x}) \end{cases} . \tag{6.3}$$

   Then update the foreground and background perturbations separately using the following rules:

$$\begin{cases} \boldsymbol{\delta}_F^{(k+1)} = \mathcal{P}_\epsilon(\boldsymbol{\delta}_F^{(k)} + \alpha_F \cdot g_F) \\ \boldsymbol{\delta}_B^{(k+1)} = \mathcal{P}_\epsilon(\boldsymbol{\delta}_B^{(k)} + \alpha_B \cdot g_B) \end{cases} \tag{6.4}$$

   where g_F is the update that corresponds to the *normalized steepest descent* constrained in the foreground, and g_B for the background. Specifically,

$$\begin{cases} g_F = \mathcal{G}(\mathcal{F}(\boldsymbol{x}) \circ \nabla_{\boldsymbol{\delta}^{(k)}} \{\mathcal{L}(h_{\boldsymbol{\theta}}(\boldsymbol{x} + \boldsymbol{\delta}^{(k)}), y)) + \lambda \cdot \mathcal{S}\left(\boldsymbol{x} + \boldsymbol{\delta}^{(k)}\right)\} \\ g_B = \mathcal{G}(\mathcal{B}(\boldsymbol{x}) \circ \nabla_{\boldsymbol{\delta}^{(k)}} \{\mathcal{L}(h_{\boldsymbol{\theta}}(\boldsymbol{x} + \boldsymbol{\delta}^{(k)}), y)) + \lambda \cdot \mathcal{S}\left(\boldsymbol{x} + \boldsymbol{\delta}^{(k)}\right)\} \end{cases} \tag{6.5}$$

   where α_F is the step size for foreground, and α_B is the step size for background.

3. *Merge.* At the end of the k-th iteration, merge the perturbations obtained in the last step by using

$$\boldsymbol{\delta}^{(k+1)} = \boldsymbol{\delta}_F^{(k+1)} + \boldsymbol{\delta}_B^{(k+1)}. \tag{6.6}$$

$\boldsymbol{\delta}^{(k+1)}$ is further used to derive the update for the normalized steepest descent at the next iteration.

4. Return to step 2 or terminate after either a fixed number of iterations.

6.3 Defense Approach

Once we are able to compute the dual-perturbation attack, we can incorporate it into conventional adversarial training paradigms for defense, as it has been demonstrated that adversarial training is highly effective in designing classification models that are robust to a given attack. Specifically, we replace the PGD attack in the adversarial training framework proposed by [71], with the proposed dual-perturbation attack. We term this approach *AT-Dual*, which aims to solve the following optimization problem:

$$\min_{\boldsymbol{\theta}} \frac{1}{|D|} \sum_{\boldsymbol{x},y \in D} \max_{\substack{||\boldsymbol{\delta} \circ \mathcal{F}(\boldsymbol{x})||_p \leq \epsilon_F, \\ ||\boldsymbol{\delta} \circ \mathcal{B}(\boldsymbol{x})||_p \leq \epsilon_B}} \mathcal{L}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta}), y\right) + \lambda \cdot \mathcal{S}\left(\boldsymbol{x}+\boldsymbol{\delta}\right). \tag{6.7}$$

Note that *AT-Dual* needs to identify background and foreground for any input when solving the inner maximization problems in Eq. (6.7) at training time. At prediction time, our approaches classify test samples like any standard classifiers, which is independent of the semantic partitions so as to close the backdoors to attacks on object detection approaches [123].

6.4 Experimental Results

6.4.1 Experimental Setup

Datasets. We conducted the experiments on the following three datasets (detailed in Appendix B.2): The first is Segment-6 [17], which are images with 32×32 pixels obtained by pre-processing the Microsoft COCO dataset [62] to make it compatible with image classification tasks. We directly used the semantic segmentation based foreground masks provided in this dataset. Our second dataset is STL-10, a subset that contains images with 96×96 pixels. Our third dataset is ImageNet-10, a 10-class subset of the ImageNet dataset [21]. We cropped all its images to be with 224×224 pixels. For STL-10 and ImageNet-10, we used fixation prediction to identify foreground and background as described in Section 6.2.

Baselines. We consider *PGD* attack as a baseline adversarial model, and *Adversarial Training with PGD Attacks* as a baseline robust classifier. We also consider a classifier trained on non-adversarial data (henceforth, *Clean*). Additionally, we consider *Randomized Smoothing* [16] and defer corresponding results to Appendix B.9.

Evaluation Metrics. We use two standard evaluation metrics for both attacks and defenses: 1) accuracy of prediction on clean test data where no adversarial attacks were attempted. 2) adversarial accuracy, which is accuracy when adversarial inputs are used in place of clean inputs.

Throughout our evaluation, we used both ℓ_2 and ℓ_∞ norms to measure the magnitude of added adversarial perturbations. We only present experimental results of the *Clean* model

and classification models that are trained to be robust to ℓ_2 norm attacks using the ImageNet-10 dataset. The results for ℓ_∞ norm and other datasets are similar and deferred to the Appendix B.

In the following experiments, all classifiers were trained with 20 epochs on a ResNet34 model [38] pre-trained on ImageNet and with a customized final fully connected layer. Specifically, we trained AT-PGD by using 50 steps of ℓ_2 PGD attack with $\epsilon = 2.0$, and AT-Dual by using 50 steps of ℓ_2 dual-perturbation attack with $\{\epsilon_F, \epsilon_B, \lambda\} = \{2.0, 20.0, 0.0\}$ at each training epoch. At test time, we used both ℓ_2 PGD and dual-perturbation attacks with 100 steps to evaluate robustness.

6.4.2 Saliency Analysis of Dual-Perturbation Adversarial Examples

We begin by considering a natural question: is our particular distinction between foreground and background actually consistent with cognitive salience? In fact, this gives rise to two distinct considerations: 1) whether foreground as we identify it is in fact significantly more salient than the background, and 2) if so, whether background becomes significantly more salient *as a result of our dual-perturbation attacks*. We answer both of these questions by appealing to DeepGaze II [55] to compute the *foreground score (FS)* of dual-perturbation examples and using the accuracy of different classifiers on dual-perturbation examples with different background salience. Concretely, DeepGaze II outputs predicted pixel-level density of human fixations on an image with the total density over the entire image summing to 1. Our measure of relative salience of the foreground, the *foreground score (FS)*, is defined as $FS = \sum_{i \in \{k | \mathcal{F}(\boldsymbol{x})_k \neq 0\}} s_i$, where s_i is the saliency score produced by DeepGaze II for pixel i of image $\boldsymbol{x}$. Since foreground, as a fraction of the image, tends to be around 50-60%, a score significantly higher than 0.5 indicates that predicted human fixation is relatively localized to the foreground.

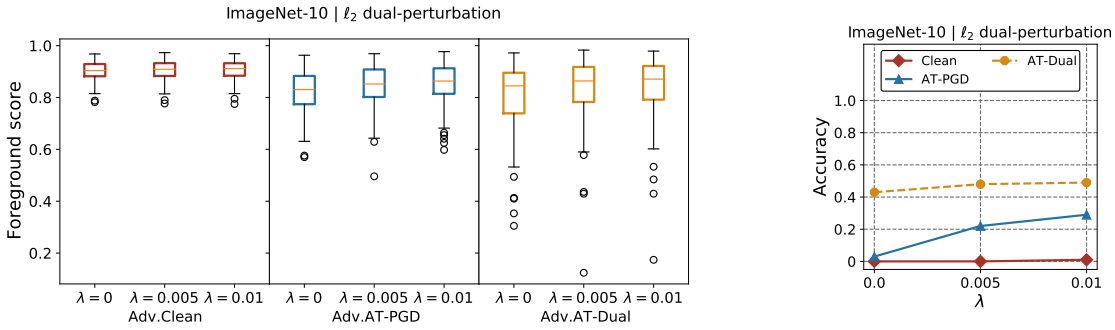

Figure 6.3: Saliency analysis. Dual-perturbation attacks are performed by using $\{\epsilon_F, \epsilon_B\} = \{2.0, 20.0\}$ and a variety of λ displayed in the figure. Left: foreground scores of dual-perturbation examples in response to different classifiers. Right: accuracy of classifiers on dual-perturbation examples with salience control.

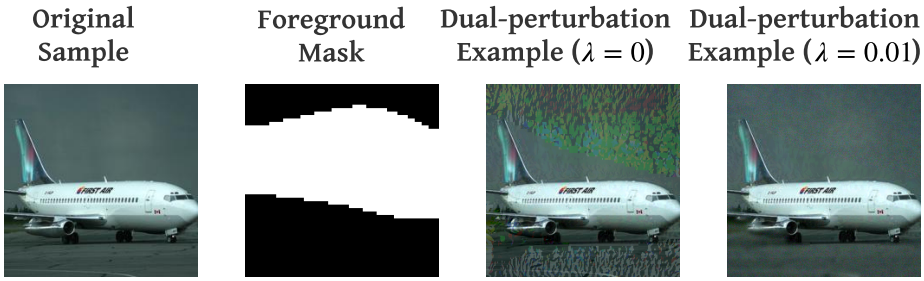

Figure 6.4: An illustration of dual-perturbation attacks. Adversarial examples are with large ℓ_∞ perturbations on the background ($\epsilon_B = 20/255$) and small ℓ_∞ perturbations on the foreground ($\epsilon_F = 4/255$). A parameter λ is used to control background salience explicitly. A larger λ results in less salient background under the same magnitude of perturbation.

Figure 6.3 presents the answer to both of the questions above. First, observe that in Figure 6.3, *FS* (vertical axis) is typically well above 0.5, and in most cases above 0.8, for all attacks. Second, this is true whether we attack the *Clean* model, or either *AT-PGD* or *AT-Dual* robust models. Particularly noteworthy, however, is the impact that the parameter λ has on the *FS*, especially when robust classifiers are employed. Recall that λ reflects the relative importance of salience in generating adversarial examples, with larger values forcing our approach to pay more attention to preserving unsuspiciousness of background relative to foreground. As we increase λ, we note higher *FS*, i.e., lower background salience (again, Figure 6.3, left).

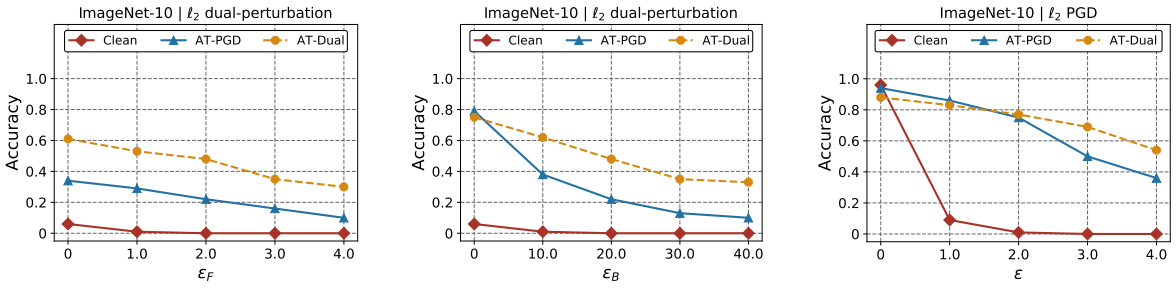

Figure 6.5: Robustness to white-box ℓ_2 attacks on ImageNet-10. Left: dual-perturbation attacks with different foreground distortions. ϵ_B is fixed to be 20.0 and $\lambda = 0.005$. Middle: dual-perturbation attacks with different background distortions. ϵ_F is fixed to be 2.0 and $\lambda = 0.005$. Right: PGD attacks.

Figure 6.4 offers a visual illustration of this effect. As significantly, Figure 6.3 (right) shows that moderately increasing λ does not significantly reduce the effectiveness of the attack on the *Clean* and *AT-Dual* classifiers.

6.4.3 Robustness against Non-Salient Adversarial Examples

Next, we evaluate the effectiveness of dual-perturbation attacks against state-of-the-art robust learning methods, as well as the effectiveness of adversarial training that uses dual-perturbation attacks for generating adversarial examples. We begin by considering white-box attacks, and subsequently evaluate transferability.

The results for white-box attacks are presented in Figure 6.5. First, consider the dual-perturbation attacks (left and middle plots). Note that in all cases these attacks are highly successful against the baseline robust classifier (AT-PGD); indeed, even relatively small levels of foreground noise yield near-zero accuracy when accompanied by sufficiently large background perturbations. For example, when the perturbation to the foreground is $\epsilon_F = 2.0$ and background perturbation is $\epsilon_B = 20.0$, *AT-PGD* achieves robust accuracy around 20%. In contrast, AT-Dual remains significantly more robust, with an improvement of up to 30%

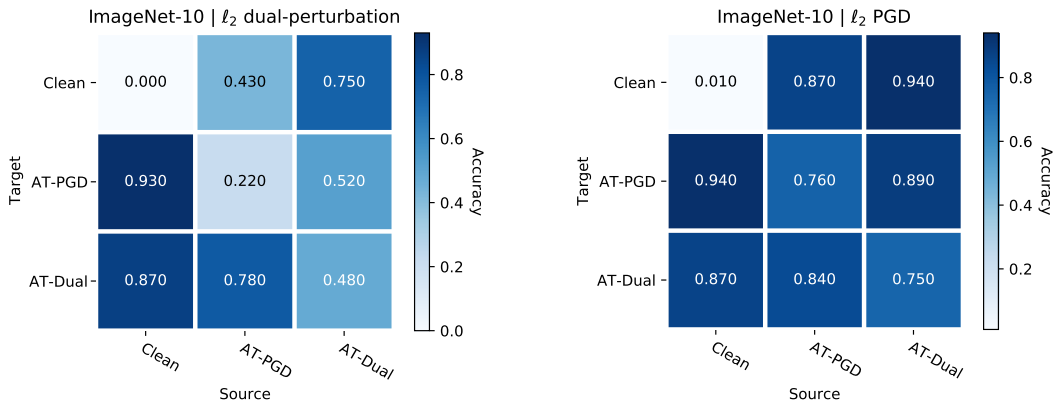

Figure 6.6: Robustness against adversarial examples transferred from other models on ImageNet-10. Left: ℓ_2 dual-perturbation attacks performed by using $\{\epsilon_F, \epsilon_B, \lambda\} = \{2.0, 20.0, 0.005\}$ on different source models. Right: ℓ_2 PGD attacks with $\epsilon = 2.0$ on different source models.

compared to the baseline. Second, consider the standard PGD attacks (right plot). It can be observed that all of the robust models are successful against the ℓ_2 PGD attacks. However, our defense exhibit moderately higher robustness than the baselines under large distortions of PGD attacks, without sacrificing much in accuracy on clean data. For example, when the perturbation of the ℓ_2 PGD attack is above $\epsilon = 3.0$, *AT-Dual* can achieve 20% more accuracy.

Next, we measure the *transferability* of adversarial examples among different classification models. To do this, we first produced adversarial examples by using ℓ_2 PGD attack or dual-perturbation attack on a source model. Then, we used these examples to evaluate the performance of an independent target model, where a higher prediction accuracy means weaker transferability. The results are presented in Figure 6.6. The first observation is that dual-perturbation attacks exhibit significantly better transferability than the conventional PGD attacks (transferability is up to 40% better for dual-perturbation attacks). Second, we can observe that when *AT-Dual* is used as the target (i.e., defending by adversarial training with dual-perturbation examples), these are typically resistant to adversarial examples generated

against either the clean model, or against *AT-PGD*. This observation obtains even when we use PGD to generate adversarial examples.

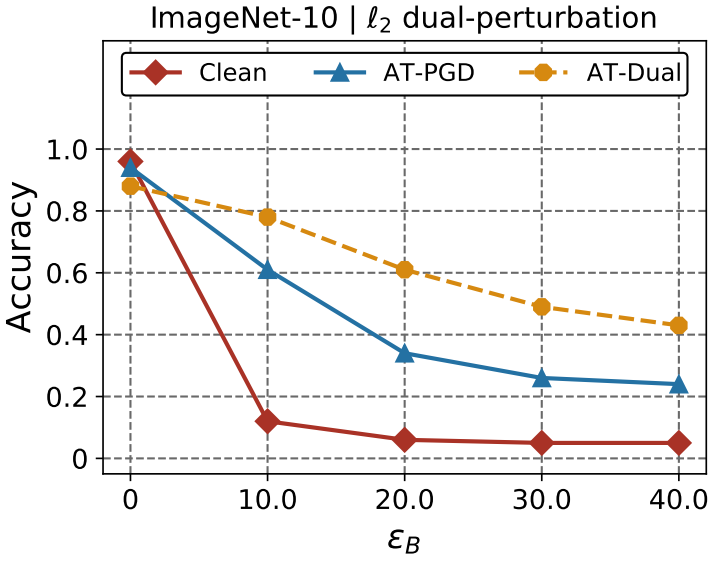

Figure 6.7: Robustness to white-box background attacks on ImageNet-10. ϵ_F is fixed to be 0.0 and $\lambda = 0.005$.

In addition, we evaluate robustness of classifiers against background perturbations. To do this, we fixed $\epsilon_F = 0$, and only background perturbations are allowed to be added to an image. The results are shown in Figure 6.7. We can observe significantly increased robustness of our defense to background perturbation, up to 25% than *AT-PGD* and 50% than the *Clean* classifier.

6.4.4 Generalizability of Defense

It has been observed that models robust against l_p-norm-bounded attacks for one value of p can be fragile when facing attacks with a different norm $l_{p'}$ [98].

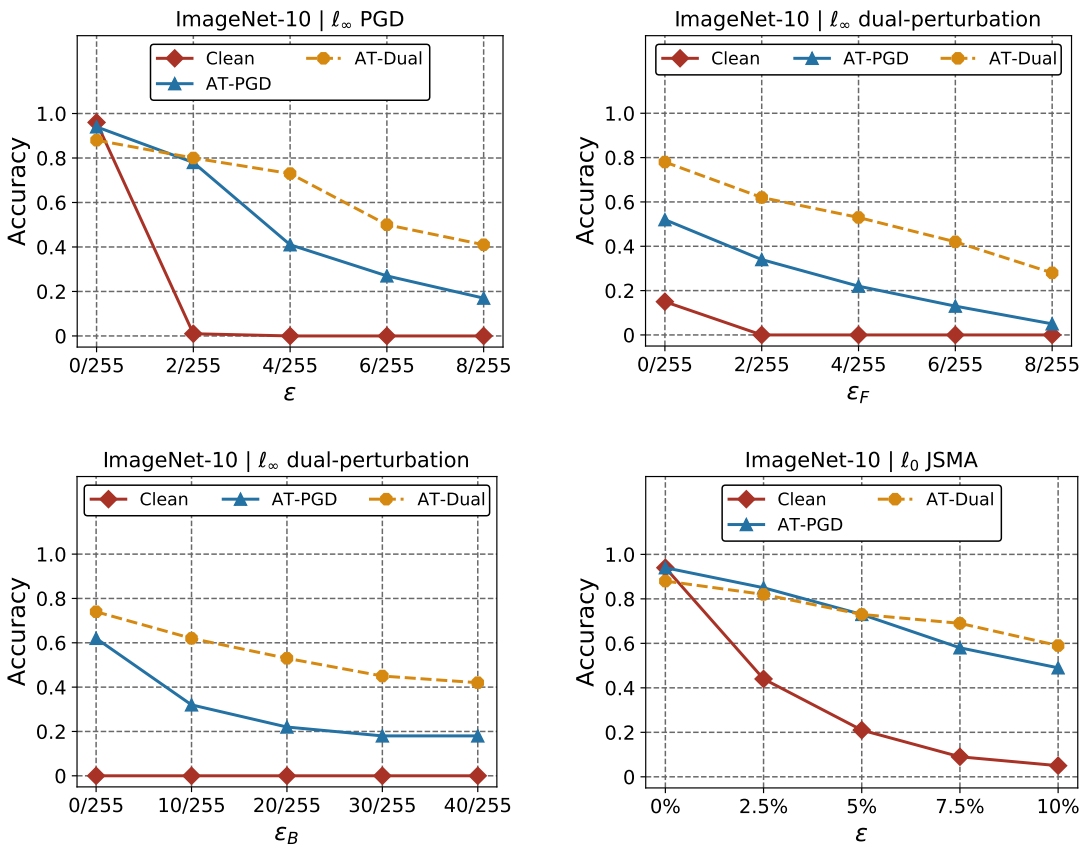

Figure 6.8: Robustness to additional white-box attacks on ImageNet-10. Top left: 20 steps of ℓ_∞ PGD attacks. Top right: 20 steps of ℓ_∞ dual-perturbation attacks with different foreground distortions. ϵ_B is fixed to be $20/255$ and $\lambda = 0.005$. Bottom left: 20 steps of ℓ_∞ dual-perturbation attacks with different background distortions. ϵ_F is fixed to be $4/255$ and $\lambda = 0.005$. Bottom right: ℓ_0 JSMA attacks.

Here, our final goal is to present evidence that the approaches for defense based on dual-perturbation attacks remain relatively robust even when faced with attacks generated using different norms. Here, we show this when our models are trained using the l_2-bounded attacks, and evaluated against other attacks using other norms. The results are presented in Figure 6.8. We consider three alternative attacks: 1) PGD using the l_∞-bounded perturbations, as in [71] (top left in Figure 6.8) 2) dual-perturbation attacks with l_∞-norm bounds (top right and bottom left in Figure 6.8), and 3) JSMA, a l_0-bounded attack [85] (bottom right in Figure 6.8). We additionally considered l_2 attacks, per Carlini and Wagner [13], but find that

all of the robust models, whether based on PGD or dual-perturbation attacks, are successful against these.

Our first observation is that *AT-Dual* is significantly more robust to l_∞-bounded PGD attacks than the adversarial training approach in which adversarial examples are generated using l_2-bounded PGD attacks (Figure 6.8 (top left)). Consequently, training with dual-perturbation attacks already exhibits better ability to generalize to other attacks compared to conventional adversarial training.

The gap between dual-perturbation-based adversarial training and standard adversarial training is even more significant when we consider l_∞ dual-perturbation attacks (top right and bottom left figures of Figure 6.8). Here, we see that robustness of PGD-based adversarially trained model is only marginally better than that of a clean model under large distortions (e.g., when $\epsilon_B \geq 20/255$ in the bottom left plot of Figure 6.8), whereas *AT-Dual* remains relatively robust.

Finally, considering JSMA attacks (see Figure 6.8 (right)), we can observe that both *AT-Dual* and *AT-PGD* remain relatively robust. However, a deeper look at Figure 6.8 (bottom right) reveals that compared to *AT-PGD*, *AT-Dual* exhibit moderately higher robustness than the baselines under large distortions of JSMA attacks. Overall, in all of the cases, the model made robust using dual-perturbation attacks remains quite robust even as we evaluate against a different attack, using a different norm.

6.4.5 Analysis of Defense

Finally, we conduct a qualitative study of adversarial robustness by investigating which pixel-level features are important for different classifiers at prediction time. To do this, we visualize the loss gradient of different classifiers with respect to pixels of the same *non-adversarial*

inputs (as introduced in [113]), shown in Figure 6.1. Our first observation is that the gradients in response to adversarially robust classifiers (AT-PGD and AT-Dual) align well with human perception, while a standard training model (Clean) results in a noisy gradient for the input images. Second, compared to adversarial training with the conventional PGD attack (AT-PGD), the loss gradient of AT-Dual provides significantly better alignment with sharper foreground edges and less noisy background. This indicates that adversarial training with the dual-pertubation attack can extract more perceptual semantics from an input image and is less dependent on the background at prediction time. In other words, our defense approach can extract highly robust and semantically meaningful features, which contribute to its robustness to a variety of attacks.

6.5 Conclusion

In this chapter, we proposed the dual-perturbation attack, a novel threat model that produces *unsuspicious adversarial examples* by leveraging the cognitive distinction between image foreground and background. As we have shown, our attack can defeat all state-of-the-art defenses. By contrast, the proposed defense approaches using our attack model can significantly improve robustness against unsuspicious adversarial examples, with relatively small performance degradation on non-adversarial data. In addition, our defense approaches can achieve comparable to, or better robustness than the alternatives in the face of traditional attacks.

Our threat model and defense motivate several new research questions. The first is whether there are more effective methods to identify foreground of images. Second, can we further improve robustness to dual-perturbation attacks? Finally, while we provide the first principled

approach for quantifying suspiciousness, there may be effective alternative approaches for doing so.

Part III

Robust Detection Pipeline

Chapter 7

Finding Needles in a Moving Haystack: Prioritizing Alerts with Adversarial Reinforcement Learning

In this chapter, we focus on deciding which of a large number of alerts to choose for further investigation—often a necessary step in the detection pipeline. One of the major challenges in using detection systems in practice is in dealing with an overwhelming number of alerts that are triggered by normal behavior (the so-called false positives), obscuring alerts resulting from actual malicious activity. While numerous methods for reducing the scope of this issue have been proposed, ultimately one must still decide how to prioritize which alerts to investigate, and most existing prioritization methods are heuristic, for example, based on suspiciousness or priority scores. We introduce a novel approach for computing a policy for prioritizing alerts using adversarial reinforcement learning. Our approach assumes that the attacker knows the full state of the detection system and the defender's alert prioritization

policy, and will dynamically choose an optimal attack. The first step of our approach is to capture the interaction between the defender and attacker in a game theoretic model. To tackle the computational complexity of solving this game to obtain a dynamic stochastic alert prioritization policy, we propose an adversarial reinforcement learning framework. In this framework, we use neural reinforcement learning to compute best response policies for both the defender and the adversary to an arbitrary stochastic policy of the other. We then use these in a double-oracle framework to obtain an approximate equilibrium of the game, which in turn yields a robust stochastic policy for the defender. Extensive experiments using case studies in fraud and intrusion detection demonstrate that our approach is effective in creating robust alert prioritization policies.

7.1 Overview

Building on the observation of the fundamental trade-off between false alert and attack detection rate, in this chapter, we propose a novel computational approach for robust alert prioritization to address the challenge. Our approach assumes a strong attacker who knows the full state of the detection environment including which alerts have been triggered, which have been investigated in the past, and even the defender's policy. We also assumed that the adversary is capable of finding and utilizing a near optimal attack strategy against the defender policy based on his knowledge of the system and defending policy. To defend against such a strong attacker, we propose to compute the optimal stochastic dynamic defender policy that chooses the alerts to investigate as a function of the observable state, and that is robust to our threat model. At the core of our technical approach is a combination of game theory with *adversarial reinforcement learning (ARL)*. Specifically, we model the problem of robust alert prioritization as a game in which the defender chooses its stochastic and dynamic policy for prioritizing alerts, while the attacker chooses which attacks to execute,

also dynamically with full knowledge of the system state. Our computational approach first uses neural reinforcement learning to compute approximately optimal policies for either player in response to a fixed stochastic policy of their counterpart. It then uses these (approximate) best response *oracles* as a part of a double-oracle framework, which iterates two steps: 1) solve a game involving a restricted set of policies by both players, and 2) augment the policy sets by calling the best response oracle for each player. *Note that our approach is completely orthogonal to methods for reducing the number of false positive alerts, such as alert correlation*, and is meant to be used in combination with these, rather than as an alternative. In particular, we can first apply alert correlation to obtain a reduced set of alerts, and subsequently use our approach for selecting which alerts to investigate. Since alert correlation cannot be overly aggressive in order to ensure that we still capture actual attacks, the number of alerts often still significantly exceeds the investigation budget.

We evaluate our approach experimentally in two application domains: intrusion detection, where we use the Suricata open-source intrusion-detection system (IDS) with a network IDS dataset, and fraud detection, with a detector learned from data using machine learning. In both settings, we show that our approach is significantly more effective than alternatives with respect to our threat model. Furthermore, we demonstrate that our approach remains highly effective, and better than baseline alternatives in nearly all cases, even when certain assumptions of our threat model are violated.

7.2 System Model

As displayed in Figure 7.1, our system is partitioned into four major components: a group of *regular users* (RU), an *adversary* (also called attacker), a *defender*, and an *attack detection environment* (ADE).

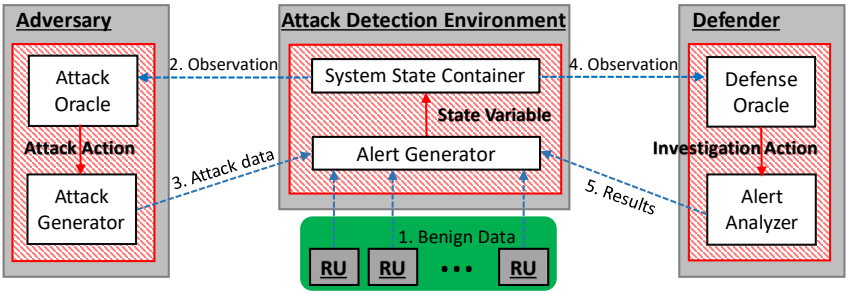

Figure 7.1: System model. The *Attack Oracle* computes the attacker's policy for executing attacks, which is implemented by the *Attack Generator* and then triggers alerts observed by the *Attack Detection Environment*. The *Defense Oracle* computes the defender's alert prioritization policy, which is implemented by the *Alert Analyzer*.

The regular users (RU) are the authorized users of a system. In contrast, the adversary is a sophisticated actor who attacks the target computer system. The attack detection environment (ADE) models the combination of the software artifact that is responsible for monitoring the system (e.g., network traffic, files, emails) and raising alerts for observed suspicious behavior, as well as relevant system state. System state includes attacks that have been executed (unknown to the defender), and alerts that have been investigated (known to both the attacker and defender). Crucially, the alerts triggered in the ADE may correspond either to behavior of the normal users RU, or to malicious behavior (attacks) by the adversary. We divide time into a series of discrete time periods. The defender is limited in how many alerts it can investigate in each time period and must select a small subset of alerts for investigation, while the adversary is limited in how many attacks it executes in each time period. The full system operates as follows for a representative time period (see again the schematic in Figure 7.1):

1. Benign alerts are generated by the ADE.

2. These alerts, and the remaining ADE system state (such as which alerts from past time periods have not yet been investigated, but could be investigated in the future), are observed by the attacker, who executes a collection of attacks.

Table 7.1: Notation summary.

Notation	Interpretation
Constants and functions	
A	Types of attacks
T	Types of alerts
C_t	Cost of investigating an alert of type $t \in T$
B	Defender's budget
E_a	Cost of mounting an attack of type $a \in A$
D	Adversary's budget
$P_{a,t}(n)$	Probability that an attack $a \in A$ raises n alerts of type $t \in T$
$\mathcal{F}_t$	Probability distribution of false alerts of type $t \in T$
L_a	Loss inflicted by an undetected attack $a \in A$
τ	Temporal discounting factor
State variables (Time slot $k \in \mathbb{N}$)	
$N_t^{(k)}$	Number of uninvestigated alerts of type $t \in T$
$M_a^{(k)}$	Indicator of whether an attack of type $a \in A$ was mounted
$S_{a,t}^{(k)}$	Number of alerts of type $t \in T$ raised due to attack $a \in A$
$R_{+1}^{(k)}$	Reward obtained by the defender
Actions, policies, and strategies	
$\boldsymbol{\alpha}_v$	Action of player $v \in \{-1, +1\}$
$\boldsymbol{\pi}_v$	Policy (i.e., pure strategy) of player $v \in \{-1, +1\}$
$\boldsymbol{\sigma}_v$	Mixed strategy of player $v \in \{-1, +1\}$

3. The attacks trigger new alerts. These are arbitrarily mixed into the full collection of alerts, which is then observed by the defender.

4. The defender chooses a subset of alerts to investigate. The ADE state is updated accordingly, and the process repeats in the next time period.

Next, we describe our model of the alert detection environment, our threat model, and our defender model. The full list of notation that we use in the model is presented in Table 7.1.

7.2.1 Attack Detection Environment (ADE) Model

Our model of the attack detection environment (ADE) captures a broad array of detection settings, including credit card fraud, intrusion, and malware detection. In this model, the ADE is composed of two parts: an *alert generator* (such as an intrusion detection system, like Suricata) and *system state*.

An alert generator produces a sequence of alerts in each time period. We aggregate alerts based on a finite predefined set of types T. For example, an alert type may be based on the application layer it was generated for (HTTP, DNS, etc), port number or range, destination IP address, and any other information that's informative for determining the nature and relative priority of alerts. We can also define alert types for meaningful sequences of alerts. Indeed, the notion of alert types is entirely without loss of generality—we can define each type to be a unique sequence of alerts, for example—but in practice it is useful (indeed, crucial for scalability) to aggregate semantically similar alerts.

At the end of each time period the system generates a collection of alert *counts* for each alert type $t \in T$. We assume that normal or benign behavior generates alerts according to a known distribution $\mathcal{F}$, where $\mathcal{F}_t(n)$ is the marginal probability that n alerts of type t are generated. We also refer to this as the distribution of *false alarms*, since if the defender were omniscient, they would never trigger such alerts. Note that in practice it is not difficult to obtain the distribution $\mathcal{F}$. Specifically, we can use past logs of *all* alerts over some time period to learn the distribution $\mathcal{F}$. Since the vast majority of alerts in real systems are in fact false positives, any unidentified true positives in the logs will have a negligible impact.[8]

[8]If we are concerned about these poisoning the data, we can use robust estimation approaches to mitigate the issue [115].

We use three matrices to represent the state of ADE at time period k. The first represents the counts of alerts not yet investigated, grouped by type. Formally, we denote this structure by $\mathbf{N}^{(k)} = \{N_t^{(k)}\}_{t \in T}$, where $N_t^{(k)}$ is the number of alerts of type $t \in T$ that were raised but have not been investigated by the defender. This is observed by *both* the defender and the attacker. The second describes which attacks have been executed by the adversary; formally, $\mathbf{M}^{(k)} = \{M_a^{(k)}\}_{a \in A}$, where $M_a^{(k)}$ is a binary indicator where $M_a^{(k)} = 1$ iff the attack a was executed. This matrix is only observed by the attacker. Finally, we represent which alerts are raised specifically due to each attack. Formally, $\mathbf{S}^{(k)} = \{S_{a,t}^{(k)}\}_{a \in A, t \in T}$, where $S_{a,t}^{(k)}$ represents the number of alerts of type $t \in T$ raised due to attack a. This is also only observed by the attacker.

7.2.2 Threat Model

Adversary's Knowledge. We consider a strong attacker who is capable of observing the current state of the ADE. This obviates the need to make specific (and potentially erroneous) assumptions about information actually available to the attacker about system state; in practice, given the zero-sum nature of the encounter we consider below, having a less informed attacker will only improve the defender's utility. Additionally, the attacker knows the randomized *policy* used by the defender for choosing which alerts to inspect (more on this below), and inspection decisions in previous rounds, but not the inspection decision in the current round (which happens after the attack).

Adversary's Capabilities. In each time period, the adversary can execute multiple actions a from a set of possible (representative) actions A.[9] Each attack action $a \in A$ stochastically triggers alerts according to the probability distribution P, where $P_{a,t}(n)$ is the marginal

[9]In practice, actions in A correspond to equivalence classes of attacks; for example, $a \in A$ could be a representative denial-of-service attack.

probability that action a generates n alerts of type t. These probabilities can be learned by replaying known attack actions through actual detectors (as we do in the experiments below), ideally as a part of a full dataset which includes a mix of benign and malicious behavior. Commonly, alerts are generated deterministically for given attack actions; it is evident that our model admits this as a special case (i.e., $P_{a,t} \in \{0, 1\}$). However, our generality allows us to handle important cases where alerts are, indeed, stochastic. For example, consider a *Port Scan* attack (as a part of a reconnaissance step). Port scan alert rules commonly consider the number of certain kinds of packets (such as ICMP packets) observed over a small time period (say, several seconds), and raise an alert if this number exceeds a predefined threshold. The number of such packets, of course, also depends on background traffic, which is stochastic, so that the triggering of the alert is also stochastic if the attack is sufficiently stealthy to avoid exceeding such a threshold in isolation.

Let E_a be the cost for executing an attack $a \in A$. One method to estimate these costs is to examine the difficulty of executing the exploit based on the CVSS complexity metrics. The main limitation to the attacker capabilities is a budget constraint D that limits how many, and which combination of, attacks can be executed.[10] While it is difficult to reliably estimate this budget, our case studies in Section 7.5 demonstrate that our approach is robust to uncertainty about this parameter. Specifically, any attack decision $\boldsymbol{\alpha}_{-1}$ with $\alpha_{-1,a}$ the probability that the attack a is executed by the attacker in a given time period, must abide by the following constraint:

$$\sum_{a \in A} \alpha_{-1,a} E_a \leq D. \tag{7.1}$$

[10]Note that this easily admits the possibility of multiple attackers, where D becomes the total budget of all attackers. This case is equivalent to assuming that attackers coordinate. This is a safe assumption, since if they do not, the defender's utility can only increase.

For our purposes, it is useful to represent the attacker as consisting of two modules: *Attack Oracle* and *Attack Generator*, as seen in Figure 7.1. The attack oracle runs a *policy*, which maps observed the state of the ADE to attacks that are executed. In each time period, after observing ADE state, the attack oracle chooses attack actions, which are then executed by the attack generator, triggering alerts and thereby modifying the state of the ADE. Below we present our approach for approximating the optimal attack policies.

Adversary's Goals. The adversary aims to successfully execute attacks. Success entails avoiding being detection by the defender, *which only happens if alerts associated with an attack are inspected*. Thus, if an attack triggers a collection of alerts, but none of these are chosen by the defender to be inspected in the current round, the attack succeeds. Different attacks, however, entail different consequences and, therefore, different rewards to the attacker (and loss to the defender). As a result, the adversary will ultimately need to balance rewards to be gained from successful attacks and the likelihood of being detected.

7.2.3 Defender Model

Defender's Knowledge. Unlike the adversary, the defender can only partially observe the state of the ADE. In particular, the defender only observes $\mathbf{N}^{(k)}$, the numbers of remaining uninvestigated alerts, grouped by alert type (since clearly the defender cannot directly observe actually attacks). In addition, we assume that the defender knows the attack budget and costs of (representative) attacks. In our experiments, we study the impact of relaxing this assumption (see Section 7.5), and provide practical guidance on this issue.

Defender's Capabilities. The defender chooses subsets of alerts in $\mathbf{N}^{(k)}$ to investigate in each time period k. This choice is constrained by the defender's budget, which in practice can translate to time the defender has to investigate alerts. Since different types of alerts may

need different amounts of time to investigate, or more generally, incur varying investigation costs, the budget constraint is on the total cost of investigating chosen alerts. Formally, let C_t be the investigation cost of an alert of type t, and let $\alpha^{(k)}_{+1,t}$ be the number of alerts of type t chosen to be investigated by the defender in period k. Then the budget constraint takes the following mathematical form:

$$\sum_{t \in T} C_t \alpha^{(k)}_{+1,t} \leq B. \tag{7.2}$$

An additional constraint imposed by the problem definition is that the defender can only investigate existing alerts:

$$\forall t \in T : \alpha^{(k)}_{+1,t} \leq N^{(k)}_t. \tag{7.3}$$

Just as with the adversary, it is useful to represent the defender as consisting of two modules: *Defense Oracle* and *Alert Analyzer*, as shown in Figure 7.1. The defense oracle runs a *policy*, which maps *partially observed* state of the ADE to the choice of a subset of alerts to be investigated. In each time period, after observing the set of as yet uninvestigated alerts, the defense oracle chooses which alerts to investigate, and this policy is then implemented by the alert analyzer, which thereby modifies ADE state (marking the selected alerts as having been investigated). Below we present our approach for approximately computing optimal defense policies that are robust to attacks as defined in our threat model above.

Defender's Goals. The goal of the defender is to guard a computer system or network by detecting attacks through alert inspection. To achieve its goal, the defender develops an investigation policy to allocate its limited budget to investigation activities in order to minimize consequences of successful attacks, where we assume that an attack will fail to accomplish its primary objectives if the alerts it causes the ADE to emit are investigated in a timely manner.

7.2.4 An Illustrative Example

Since our system is built on top of an abstracted model of alert investigation, the results are generally applicable to a wide range of real-world problems. We will use intrusion detection as an illustrative example in this section. *Port Scan* reconnaissance attack is one of the most common initial steps in remote exploitation and is a common occurrence faced by many enterprise IT professionals. In a Suricata IDS system, each alert item has different levels of categorization. For example, at the lowest layer, the port scan may trigger two types of alert, 1) *Httprecon Web Server Fingerprint Scan*, and 2) *ET SCAN NMAP -sO*. At a higher level, these alerts can be categorized into *attempted-recon* (since both reflect potential reconnaissance efforts by the attacker), as is the case in the *Emerging Threats Ruleset* of Suricata. A defender can choose different granularities of attack categorization to map the IDS alert types into the abstracted types in our proposed model based on individual needs. Besides categorization, the defender can also make use of other attributes in the IDS alerts to aid in abstracted type assignment. For example, a port scan on the enterprise file server can be assigned to the abstracted type of *high-risk-recon*, while a port scan on employee desktop can be assigned to *attempted-recon*.

In addition to the alerts corresponding to an actual attack action, normal user behavior can generate false positive alerts. For example, a user who is scraping the web for weather data monitoring may trigger the *ET POLICY POSSIBLE Web Crawl using Curl*, which is grouped into the *attempted-recon* type by the same *Emerging Threats* Suricata ruleset. Leveraging the proposed game-theoretic model on these abstracted alerts, it is possible for the defender to devise an optimal defense policy for a wide range of alert applications even in the face of possible false positives.

7.3 Game Theoretic Model of Robust Alert Prioritization

We now turn to the proposed approach for robust alert prioritization. We model the interaction between the defender and attacker as a *zero-sum* game, which allows us to define and subsequently compute robust stochastic inspection policies for the defender. In this section, we formally describe the game model. We then present the computational approach for solving it in Section 7.4.

7.3.1 Strategies

The game has two players: the defender (denoted by $v = +1$) and the adversary (denoted by $v = -1$). Each player's strategies are policies, that is, mappings from an observed ADE state to the probability distribution over actions to take in that state. In a given state, the defender chooses a subset of alerts to investigate; thus, the defender's set of possible actions is the set of all alert subsets that satisfy the constraints (7.2) and (7.3). The attacker's choices in a given state correspond to subsets of actions A to take. Consequently, the set of adversary's actions is the set of all subsets of attacks satisfying constraint (7.1). Note that the combinatorial nature of both players' action spaces and of the state space makes even *representing deterministic policies* non-trivial; we will deal with this issue in Section 7.4. Moreover, we will consider stochastic policies. An equivalent way to represent stochastic policies is as probability distributions over deterministic policies, which map observed state to a *particular* action (subset of alerts for the defender, subset of attacks for the adversary).

Henceforth, we call deterministic policies of the players their *pure strategies* and stochastic policies are termed *mixed strategies*, following standard terminology in game theory.[11]

Let $\boldsymbol{\pi}_{-1}$ denote the attacker's policy, which maps the fully observed state of ADE, $\boldsymbol{O}_{-1}^{(k)} = \langle \mathbf{N}^{(k)}, \mathbf{M}^{(k)}, \mathbf{S}^{(k)} \rangle$, to a subset of attacks. Let $\boldsymbol{\alpha}_{-1}^{(k)} = \boldsymbol{\pi}_{-1}(\boldsymbol{O}_{-1}^{(k)})$, where $\boldsymbol{\alpha}_{-1}^{(k)} = \{\alpha_{-1,a}^{(k)}\}_{a \in A}$ are (for the moment) binary indicators with $\alpha_{-1,a}^{(k)} = 1$ iff an action $a \in A$ is chosen by the attacker. In other words, the vector $\boldsymbol{\alpha}_{-1}^{(k)}$ represents the choice of actions made by the adversary. Similarly, $\boldsymbol{\pi}_{+1}$ denotes the defender's policy, which maps the portion of ADE state $\boldsymbol{O}_{+1}^{(k)} = \mathbf{N}^{(k)}$ observed by the defender to the number of alerts of each type to investigate. Analogously to the attacker, $\boldsymbol{\alpha}_{+1}^{(k)} = \boldsymbol{\pi}_{+1}(\boldsymbol{O}_{+1}^{(k)})$, where $\boldsymbol{\alpha}_{+1}^{(k)} = \{\alpha_{+1,t}^{(k)}\}_{t \in T}$ are the counts of alerts chosen to be investigated for each type t. Now, notice that all alerts of type t are equivalent by definition; consequently, it makes no difference to the defender which of these are chosen, and we therefore choose the fraction $\frac{\alpha_{+1,t}^{(k)}}{N_t^{(k)}}$ of alerts of type t uniformly at random.

Let $\boldsymbol{\Pi}_v$ be player v's set of pure strategies, where each pure strategy $\boldsymbol{\pi}_v \in \boldsymbol{\Pi}_v$ is a policy as defined above. A mixed strategy of player v is then a probability distribution $\boldsymbol{\sigma}_v = \{\sigma_v(\boldsymbol{\pi}_v)\}_{\boldsymbol{\pi}_v \in \boldsymbol{\Pi}_v}$ over the player's pure strategies $\boldsymbol{\Pi}_v$ where $\sigma_v(\boldsymbol{\pi}_v)$ is the probability that player v uses policy $\boldsymbol{\pi}_v$. Since a mixed strategy $\boldsymbol{\sigma}_v$ is a distribution over a finite set of pure strategies, it satisfies $0 \leq \sigma_v(\boldsymbol{\pi}_v) \leq 1$ and $\sum_{\boldsymbol{\pi}_v \in \boldsymbol{\Pi}_v} \sigma_v(\boldsymbol{\pi}_v) = 1$. Let $\boldsymbol{\Sigma}_v$ denote the set of all mixed strategies of player v.

[11] At decision time, players can sample from their respective mixed strategies in each round, thereby determining their decisions in that round. We assume that while the defender's mixed strategy is known to the attacker, the realizations, or samples, of deterministic policies drawn in each round are not observed by the attacker; for example, the sampling process can take place after the entire set of alerts in that round are observed. Note that if we re-sample independently in each round, the attacker learns no additional information about the defender's policy from past rounds.

7.3.2 Utilities

For any strategy profile of the two players, $(\boldsymbol{\pi}_v, \boldsymbol{\pi}_{-v})$, we denote the utility of each player v by $U_v(\boldsymbol{\pi}_v, \boldsymbol{\pi}_{-v})$, $v \in \{+1, -1\}$. Since our game is *zero-sum*, $\sum_{v\in\{+1,-1\}} U_v(\boldsymbol{\pi}_v, \boldsymbol{\pi}_{-v}) = 0$. When player v chooses pure strategy $\boldsymbol{\pi}_v \in \boldsymbol{\Pi}_v$ and its opponent $-v$ plays mixed strategy $\boldsymbol{\sigma}_{-v} \in \boldsymbol{\Sigma}_{-v}$, then the expected utility of v is

$$U_v(\boldsymbol{\pi}_v, \boldsymbol{\sigma}_{-v}) = \sum_{\boldsymbol{\pi}_{-v}\in\boldsymbol{\Pi}_{-v}} \sigma_{-v}(\boldsymbol{\pi}_{-v}) U_v(\boldsymbol{\pi}_v, \boldsymbol{\pi}_{-v}). \tag{7.4}$$

Similarly, the expected utility of player v when it chooses the mixed strategy $\boldsymbol{\sigma}_v \in \boldsymbol{\Sigma}_v$ and its opponent play the mixed strategy $\boldsymbol{\sigma}_{-v} \in \boldsymbol{\Sigma}_{-v}$ is

$$U_v(\boldsymbol{\sigma}_v, \boldsymbol{\sigma}_{-v}) = \sum_{\boldsymbol{\pi}_v\in\boldsymbol{\Pi}_v} \sigma_{-v}(\boldsymbol{\pi}_v) U_v(\boldsymbol{\pi}_v, \boldsymbol{\sigma}_{-v}). \tag{7.5}$$

Next, we describe how to compute the utility of player v, $U_v(\boldsymbol{\pi}_v, \boldsymbol{\pi}_{-v})$, when its policy is $\boldsymbol{\pi}_v$ and the opponent's policy $\boldsymbol{\pi}_{-v}$ are given.

Consider arbitrary pure strategies of both players, $\boldsymbol{\pi}_{+1}$ and $\boldsymbol{\pi}_{-1}$. The game begins with an initial system state $\langle \boldsymbol{N}^{(0)}, \boldsymbol{M}^{(0)}, \boldsymbol{S}^{(0)} \rangle = \langle \mathbf{0}, \mathbf{0}, \mathbf{0} \rangle$. The system state is then updated in each time period k as follows:

1. *Alert investigation.* The defender first investigates a subset of alerts produced thus far. Specifically, the defender chooses the number of alerts of each type to investigate $\{\alpha_{+1,t}^{(k)}\}_{t\in T}$ according to its policy $\boldsymbol{\pi}_{+1}(\boldsymbol{O}_{+1}^{(k)})$ given current observed state $\boldsymbol{O}_{+1}^{(k)}$. For each attack $a \in A$, let $\widetilde{M}_a^{(k)}$ be an indicator of whether attack a has been executed by the beginning of time period k, but has not been investigated. If $M_a^{(k)} = 0$, we have $\widetilde{M}_a^{(k)} = 0$ as no attack $a \in A$ has been executed. If $M_a^{(k)} = 1$, then $\widetilde{M}_a^{(k)} = 1$ with

probability

$$p_a^{(k)} = \prod_{t \in T} \left\{ \frac{C(N_t^{(k)} - S_{a,t}^{(k)}, \alpha_{+1,t}^{(k)})}{C(N_t^{(k)}, \alpha_{+1,t}^{(k)})} \right\}, \tag{7.6}$$

where $C(n, r)$ is the number of possible combinations of r objects from a set of n objects. $p_a^{(k)}$ is then the probability that attack a is not detected by the defender.

2. *Attack generation.* The adversary produces attacks by executing actions according to its policy $\{\alpha_{-1,a}^{(k)}\}_{a \in A} = \boldsymbol{\pi}_{-1}(\boldsymbol{O}_{-1}^{(k)})$ given the fully observed ADE state $\boldsymbol{O}_{-1}^{(k)}$. Then $M_a^{(k+1)} = \alpha_{-1,a}^{(k)}$ for each $a \in A$.

3. *Triggering alerts.* Each attack $a \in A$ can trigger alerts as follows. For each attack $a \in A$ and alert type $t \in T$, if $M_a^{(k+1)} = 1$, then $S_{a,t}^{(k+1)} = n$ with probability $P_{a,t}(n)$ for $n \geq 0$. This probability can be estimated, for example, by feeding inputs which include representative attacks into an attack detector and observing relative frequencies of alerts that are triggered. In addition, false alerts are generated according to the distribution $\mathcal{F}_t$, which we can estimate from data of normal behavior and associated alert counts. Let $f_t^{(k)}$ be the number of false alerts of type $t \in T$ that have been generated. Then the total number of alerts in the next time period $k+1$ is $N_t^{(k+1)} = f_t^{(k)} + S_{a,t}^{(k+1)}$.

In order to define the reward received by the defender in time period k, we make the following assumption: *if any of the alerts raised by an attack is chosen to be inspected, then the attack is detected; otherwise, the attack is not detected.* Let L_a be the loss incurred by the defender when an attack $a \in A$ is not detected. Then the reward of the defender obtained in time period k is

$$R_{+1}^{(k)} = -\sum_{a \in A} L_a \cdot \widetilde{M}_a^{(k)}. \tag{7.7}$$

For an arbitrary pure strategy profile of the defender and adversary, $(\boldsymbol{\pi}_{+1}, \boldsymbol{\pi}_{-1})$, the defender's utility from the game is the expected total discounted sum of the reward accrued in each

time period:

$$U_{+1}(\boldsymbol{\pi}_{+1}, \boldsymbol{\pi}_{-1}) = \mathbb{E}\left[\sum_{k=0}^{\infty} \tau^k \cdot R_{+1}^{(k)}\right], \tag{7.8}$$

where $\tau \in (0, 1)$ is a temporal discounting factor which implies that future rewards are less important than current rewards. That is, imminent losses are more important to the defender than potential future losses. The adversary's utility is then $U_{-1}(\boldsymbol{\pi}_{+1}, \boldsymbol{\pi}_{-1}) = -U_{+1}(\boldsymbol{\pi}_{+1}, \boldsymbol{\pi}_{-1})$.

7.3.3 Solution Concept

Our goal of finding robust alert investigation policies amounts to computing a *mixed-strategy Nash equilibrium (MSNE)* of our game by the well-known equivalence between MSNE, maximin, and minimax solutions in zero-sum games [54]. A mixed-strategy profile $(\boldsymbol{\sigma}_v^*, \boldsymbol{\sigma}_{-v}^*)$ of the two players is an MSNE if it satisfies the following condition for all $v \in \{+1, -1\}$

$$U_v(\boldsymbol{\sigma}_v^*, \boldsymbol{\sigma}_{-v}^*) \geq U_v(\boldsymbol{\sigma}_v, \boldsymbol{\sigma}_{-v}^*) \quad \forall \boldsymbol{\sigma}_v \in \boldsymbol{\Sigma}_v. \tag{7.9}$$

That is, each player v chooses a stochastic policy $\boldsymbol{\sigma}_v^*$ that is the *best response* (is optimal for v) when its opponent chooses $\boldsymbol{\sigma}_{-v}^*$.

7.4 Computing Robust Alert Prioritization Policies

7.4.1 Solution Overview

For given sets of policies, $\mathbf{\Pi}_{+1}$ and $\mathbf{\Pi}_{-1}$, a standard approach to computing the MSNE of a zero-sum game is to solve a linear program of the following form:

$$\begin{aligned} \max \quad & U_v^* \\ \text{s.t.} \quad & \textstyle\sum_{\boldsymbol{\pi}_v \in \mathbf{\Pi}_v} U_v(\boldsymbol{\pi}_v, \boldsymbol{\pi}_{-v}) \cdot \sigma_v(\boldsymbol{\pi}_v) \geq U_v^*, \forall \boldsymbol{\pi}_{-v} \in \mathbf{\Pi}_{-v} \\ & \textstyle\sum_{\boldsymbol{\pi}_v \in \mathbf{\Pi}_v} \sigma_v(\boldsymbol{\pi}_v) = 1 \\ & \sigma_v(\boldsymbol{\pi}_v) \geq 0 \qquad\qquad\qquad\qquad \forall \boldsymbol{\pi}_v \in \mathbf{\Pi}_v \end{aligned} \tag{7.10}$$

where in our case the optimal solution $\boldsymbol{\sigma}_{+1}^*$ yields the robust alert prioritization policy for the defender. However, using this approach for our problem entails two principal technical challenges: 1) the space of policies for both players is intractably large, and 2) it is even intractable to explicitly represent individual policies, since they map a combinatorial set of states to a combinatorial set of actions for both players.

We propose an adversarial reinforcement learning approach to address these challenges, which combines a *double oracle* framework [75] with neural reinforcement learning. The general double oracle approach is illustrated in Figure 7.2. We start with an arbitrary small collection of policies for both players, $(\mathbf{\Pi}_{+1}, \mathbf{\Pi}_{-1})$, and solve the linear program (7.10), obtaining provisional equilibrium mixed strategies $(\boldsymbol{\sigma}_{+1}, \boldsymbol{\sigma}_{-1})$ of the restricted game. Next, we query the attack oracle to compute the adversary's best response $\boldsymbol{\pi}_{-1}(\boldsymbol{\sigma}_{+1})$ to the defender's equilibrium mixed strategy $\boldsymbol{\sigma}_{+1}$, and, similarly, query the defense oracle to compute the defender's best response $\boldsymbol{\pi}_{+1}(\boldsymbol{\sigma}_{-1})$ to the adversary's equilibrium mixed strategy $\boldsymbol{\sigma}_{-1}$. The best response policies are then added to the policy sets $(\mathbf{\Pi}_{+1}, \mathbf{\Pi}_{-1})$ of the players, and we then

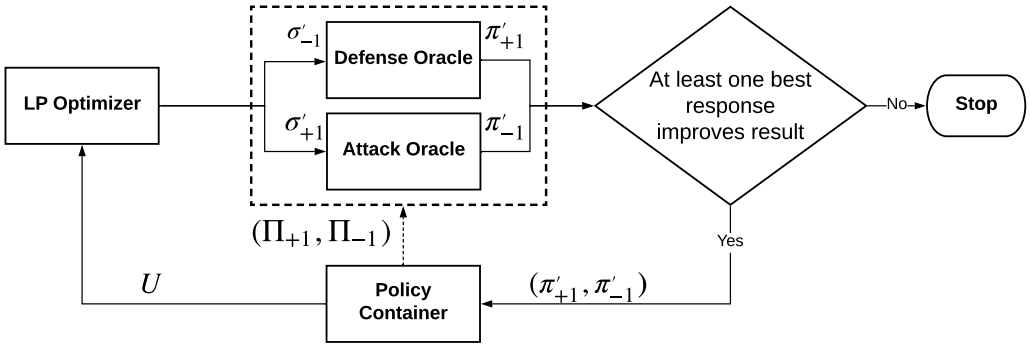

Figure 7.2: The game solver based on the double oracle algorithm.

re-solve the linear program and repeat the process. The process stops when neither player's best response policy yields appreciable improvement in utility compared to the provisional equilibrium mixed strategy. Since the space of possible policies in our case is infinite, this process may not converge.

However, in our experiments the procedure converged in fewer than 15 iterations (see Figure 7.12 in Section 7.5.4), with the fast convergence in part due to the way we represent policies, as discussed below. The main question that remains is how to compute or approximate the best response oracles for both players. To this end, we use reinforcement learning techniques with policies represented using neural networks. Below, we explain both our double oracle approach and our neural reinforcement learning methods (including the specific way in which we represent policies) in further detail.

7.4.2 Policy-Based Double Oracle Method

As displayed in Figure 7.2, our game solver is an extension of the double oracle algorithm proposed in [112] and is partitioned into four parts: a policy container, a linear programming (LP) optimizer, a defense oracle, and an attack oracle. The policy container stores the policies of the two players, $\mathbf{\Pi}_{+1}$ and $\mathbf{\Pi}_{-1}$, as well as a utility matrix $\boldsymbol{U}$, whose elements are

$U_{+1}(\boldsymbol{\pi}_{+1}, \boldsymbol{\pi}_{-1})$ for all $\boldsymbol{\pi}_{+1} \in \boldsymbol{\Pi}_{+1}$ and $\boldsymbol{\pi}_{-1} \in \boldsymbol{\Pi}_{-1}$. The LP optimizer solves the game by computing the current mixed-strategy Nash equilibrium given the utility matrix $\boldsymbol{U}$. The defense and attack oracles are agents that apply reinforcement learning to compute the optimal responses to their opponents' mixed strategies, which are provided by the LP optimizer.

Our solver works in an iterative manner such that the players' policies and the utility matrix grow incrementally. Initially, $\boldsymbol{\Pi}_{+1}$, $\boldsymbol{\Pi}_{-1}$ can be set up with some basic policies, for example, uniformly allocating each player's budget among their choices. Then, the policy sets, jointly encapsulated in a *policy container*, are updated in each iteration as follows:

1. First, the LP optimizer computes the mixed-strategy Nash Equilibrium $(\boldsymbol{\sigma}'_{+1}, \boldsymbol{\sigma}'_{-1})$ of the current iteration by solving the optimization problems presented in Eq. (7.10).

2. The oracle of player v computes the best response policy $\boldsymbol{\pi}'_v$ given that its opponent uses its equilibrium mixed-strategy $\boldsymbol{\sigma}'_{-v}$, for $v \in \{+1, -1\}$.

3. If $U_v(\boldsymbol{\pi}'_v, \boldsymbol{\sigma}'_{-v}) \leq U_v(\boldsymbol{\sigma}'_v, \boldsymbol{\sigma}'_{-v})$ for all $v \in \{+1, -1\}$, the double oracle algorithm terminates and returns $(\boldsymbol{\sigma}'_{+1}, \boldsymbol{\sigma}'_{-1})$ as the approximate MSNE. Otherwise, add $\boldsymbol{\pi}'_v$ to the corresponding $\boldsymbol{\Pi}_v$, update the utility matrix $\boldsymbol{U}$ and continue from Step 2.

The resulting $(\boldsymbol{\sigma}'_{+1}, \boldsymbol{\sigma}'_{-1})$ is an approximate mixed-strategy Nash equilibrium $(\boldsymbol{\sigma}^*_{+1}, \boldsymbol{\sigma}^*_{-1})$.

Next, we describe how the defense and attack oracles apply neural reinforcement learning to compute their best responses to an arbitrary mixed-strategy of the opponent.

7.4.3 Approximate Best Response Oracles with Neural Reinforcement Learning

We now turn to our approach to compute $\boldsymbol{\pi}'_v$, the optimal response of player v when its opponent uses a mixed strategy $\boldsymbol{\sigma}'_{-v}$ such that

$$\boldsymbol{\pi}'_v = \arg\max_{\boldsymbol{\pi}_v} U_v(\boldsymbol{\pi}_v, \boldsymbol{\sigma}'_{-v}). \tag{7.11}$$

This problem poses a major technical challenge, since the spaces of possible policies for both the defender and the attacker are quite large. To address this, we propose using the reinforcement learning (RL) paradigm. However, the use of RL poses two further challenges in our setting. First, for a given state, each player's set of possible actions is combinatorial. For example, the attacker is choosing subsets of attacks, whereas the defender is choosing subsets of alerts. Consequently, we cannot use common methods such as *Q-learning*, which requires explicitly representing the action-value function $Q(x, a)$ for every possible action a, even if we approximate this function over states x using, e.g., a neural network, as is common in deep RL. We can address this issue by appealing to *actor-critic* methods for RL, where the policy is represented as a parametric function $\boldsymbol{\pi}_{v;\theta}$ with parameters θ. However, this brings up the second challenge: actor-critic approaches learn policies using gradient-based methods, which require that the actions are continuous. In our case, however, the actions are discrete.

One solution is to learn the action-value function $Q(x, a)$ over a vector-based representation of actions, such as using a binary vector to indicate which attacks are used. The problem with this approach, however, is that the resulting policy $\boldsymbol{\pi}_v \in \arg\max_{a \in A} Q(x, a)$ is hard to compute in real time, since it involves a combinatorial optimization problem in its own right. We therefore opt for a much more scalable solution that uses the actor-critic paradigm with an

alternative representation of the adversary and defender policies, which admits gradient-based learning.

We start with the adversary. Recall that the adversary's policy maps a state to a subset of attack actions A, with the constraint on the total budget used by the chosen actions. Instead of returning a discrete subset of actions, we map the adversary's policy to a *probability distribution* over actions, overloading our prior notation so that $\alpha^{(k)}_{-1,a}$ now denotes the *probability* that action $a \in A$ is executed. Now the policy can be used with actor-critic methods, but it may violate the budget constraint. To address this final issue, we simply project the probability distribution into the feasible space at execution time by normalizing it by the total cost of the distribution, and then multiplying by the budget constraint. Notice that in this process we have relaxed the attacker's budget constraint to hold only in *expectation*; however, this only makes the attacker stronger. An interesting side-effect of our transformation of the adversary's policy space is that the RL method will now effectively search in the space of *stochastic* adversary policies. An associated benefit is that it leads to faster convergence of the double oracle approach.

Next, consider the defender. In this case, we can simply represent the policy as a mapping to fractions of the *total defense budget* allocated to each alert type t. In other words, for each alert type t, the policy will output the maximum fraction of the defense budget that will be used to inspect alerts of type t. This simultaneously makes the mapping continuous, and obviates the need to explicitly deal with the budget constraint.

The final nuance is that RL methods are typically designed for a fixed environment, whereas our setting is a game. However, note that since we are concerned only with each player's best response to the other's mixed strategy, we can embed the mixed strategy of the opponent as

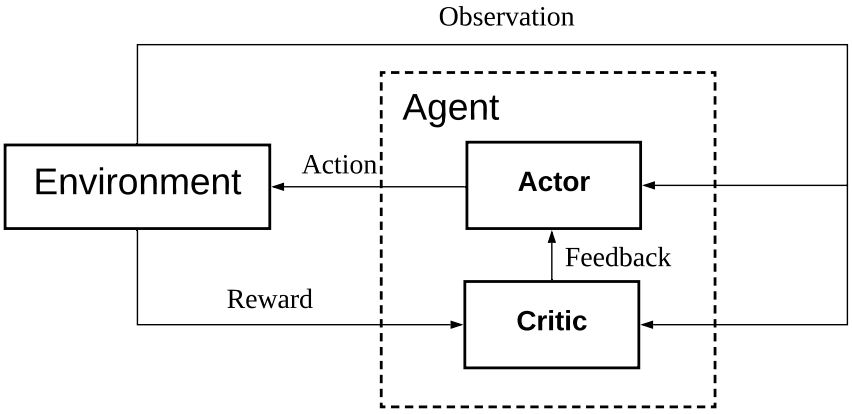

Figure 7.3: The interactions among actor, critic and environment.

a part of the environment. Next, we describe our application of actor-critic methods to our problem, given the alternative representations of adversary and defender policies above.

The basic idea of the actor-critic method is that we can iteratively learn and improve a policy without enumerating actions by using two parallel processes that interact with each other: an actor which develops a policy, and a critic network which evaluates the policy. The interaction between the actor and critic in illustrated in Figure 7.3. In each iteration, the actor and critic proceed as follows:

1. The actor executes an action according to its policy given the observation of the environment.

2. Upon receiving the action, the environment updates its system state and returns a reward to the critic.

3. The critic updates its evaluation method and provides feedback to the actor.

4. The actor updates its policy according to the feedback given by the critic.

We propose *DDPG-MIX*, actor-critic algorithm that operates in continuous action spaces and computes an approximate best response to an opponent who uses a stochastic policy. DDPG-MIX is an extension of the *Deep Deterministic Policy Gradient (DDPG)* approach proposed in [61] to our setting, and the full algorithm is outlined in Algorithm 4. For each player v, DDPG-MIX employs two neural networks to represent the actor and critic: a policy network $\boldsymbol{\pi}_v(\boldsymbol{O}_v|\boldsymbol{\theta}_v^\pi)$ for the actor, which has parameters $\boldsymbol{\theta}_v^\pi$ and maps an observation $\boldsymbol{O}_v$ into an action, and a value network $Q_v(\boldsymbol{O}_v, \boldsymbol{\alpha}_v|\boldsymbol{\theta}_v^Q)$ for the critic, which has parameters $\boldsymbol{\theta}_v^Q$ and maps an observation $\boldsymbol{O}_v$ and an action $\boldsymbol{\alpha}_v$ into a value. Initially, these two neural networks are randomly initialized. Then, we train these two iteratively with multiple episodes, each of which contains multiple steps. At the beginning of each episode, the opponent samples a deterministic policy $\boldsymbol{\pi}_{-v}$ with its mixed-strategy $\boldsymbol{\sigma}_{-v}$. The policy network and value network are then updated as follows. First, we generate an action by using the ϵ-greedy method: we randomly choose an action with probability ϵ (called *exploration* in RL), and apply the policy network $\boldsymbol{\pi}_v(\boldsymbol{O}_v|\boldsymbol{\theta}_v^\pi)$ to produce an action corresponding to the current state with probability $1-\epsilon$ (called *exploitation*). Player v then executes the action produced and so does its opponent, which executes an action $\boldsymbol{\alpha}_{-v}$ returned by $\boldsymbol{\pi}_{-v}$. Once the system state of the environment is updated, player v receives the reward and stores the transition into a memory buffer. Player v then samples a minibatch, a subset of transitions randomly sampled from the buffer, to update the value network $Q_v(\boldsymbol{O}_v, \boldsymbol{\alpha}_v|\boldsymbol{\theta}_v^Q)$ by minimizing a loss function as in most regression tasks. The sampled gradient of the value network with respect to $\boldsymbol{\alpha}_v$ is then forwarded to the policy network, which is further applied to update $\boldsymbol{\pi}_v(\boldsymbol{O}_v|\boldsymbol{\theta}_v^\pi)$ as presented in Eq. (7.12) in Algorithm 4. After a fixed number of episodes, the resulting policy network $\boldsymbol{\pi}_v(\boldsymbol{O}_v|\boldsymbol{\theta}_v^\pi)$ is returned as the parameterized optimal response to an opponent with mixed-strategy $\boldsymbol{\sigma}_{-v}$.

Algorithm 4 DDPG-MIX Algorithm: Compute the pure-strategy best response of player v when its opponent takes mixed-strategy $\boldsymbol{\sigma}_{-v}$.

Input:
The set of opponent's pure strategies, $\boldsymbol{\Pi}_{-v}$;
Mixed strategy of the opponent, $\boldsymbol{\sigma}_{-v}$;
Output:
The value network of player v, $Q_v(\boldsymbol{O}_v, \boldsymbol{\alpha}_v|\boldsymbol{\theta}_v^Q)$;
The policy network of player v, $\boldsymbol{\pi}_v(\boldsymbol{O}_v|\boldsymbol{\theta}_v^\pi)$;
Randomly initialize $Q_v(\boldsymbol{O}_v, \boldsymbol{\alpha}_v|\boldsymbol{\theta}_v^Q)$ and $\boldsymbol{\pi}_v(\boldsymbol{O}_v|\boldsymbol{\theta}_v^\pi)$;
Initialize replay memory $\mathcal{D}$;
for *episode* $= 0, M-1$ **do**
  Initialize the system state $\langle \boldsymbol{N}^{(0)}, \boldsymbol{M}^{(0)}, \boldsymbol{S}^{(0)} \rangle = \langle \mathbf{0}, \mathbf{0}, \mathbf{0} \rangle$;
  Sample the opponent's policy $\boldsymbol{\pi}_{-v}$ by using $\boldsymbol{\sigma}_{-v}$ over $\boldsymbol{\Pi}_{-v}$;
  for $k = 0, K-1$ **do**
    With probability ϵ select a random action $\boldsymbol{\alpha}_v^{(k)}$; Otherwise, select $\boldsymbol{\alpha}_v^{(k)} = \boldsymbol{\pi}_v(\boldsymbol{O}_v^{(k)}|\boldsymbol{\theta}_v^\mu)$;
    Execute $\boldsymbol{\alpha}_v^{(k)}$ and $\boldsymbol{\alpha}_{-v}^{(k)} = \boldsymbol{\pi}_{-v}(\boldsymbol{O}_{-v}^{(k)})$, observe reward $r_v^{(k)}$ and transit the system state to $\boldsymbol{S}^{k+1}$;
    Store transition $(\boldsymbol{O}_v^{(k)}, \boldsymbol{\alpha}_v^{(k)}, r_v^{(k)}, \boldsymbol{O}_v^{(k+1)})$ in $\mathcal{D}$;
    Sample a random minibatch of N transitions $(\boldsymbol{O}_v^{(i)}, \boldsymbol{\alpha}_v^{(i)}, r_v^{(i)}, \boldsymbol{O}_v^{(i+1)})$ from $\mathcal{D}$;
    Set $y_v^{(i)} = r_v^{(i)} + \tau Q_v(\boldsymbol{O}_v^{(i+1)}, \boldsymbol{\pi}(\boldsymbol{O}_v^{(i+1)}|\boldsymbol{\theta}_v^\pi)|\boldsymbol{\theta}_v^Q)$;
    Update the value network by minimizing the loss

$$\mathcal{L}(\boldsymbol{\theta}_v^Q) = \frac{1}{N}\sum_i (y_v^{(i)} - Q_v(\boldsymbol{O}_v^i, \boldsymbol{\alpha}_v^{(i)}|\boldsymbol{\theta}_v^Q))^2;$$

    Update the policy network by using the sampled policy gradient:

$$\nabla_{\boldsymbol{\theta}_v^\pi} \mathcal{J} \approx \frac{1}{N}\sum_i J_a \cdot J_\theta \tag{7.12}$$

    where

$$\begin{cases} J_a = \nabla_{\boldsymbol{\alpha}_v} Q_v(\boldsymbol{O}_v, \boldsymbol{\alpha}_v|\boldsymbol{\theta}_v^Q)|_{\boldsymbol{O}_v=\boldsymbol{O}_v^{(i)}, \boldsymbol{\alpha}_v=\boldsymbol{\pi}_v(\boldsymbol{O}_v^{(i)})} \\ J_\theta = \nabla_{\boldsymbol{\theta}_v^\pi} \boldsymbol{\pi}(\boldsymbol{O}_v|\boldsymbol{\theta}_v^\pi)|_{\boldsymbol{O}_v^{(i)}} \end{cases} \tag{7.13}$$

  end for
end for
return Player v's policy network, $\boldsymbol{\pi}_v(\boldsymbol{O}_v|\boldsymbol{\theta}_v^\pi)$.

7.4.4 Preprocessing

An important consideration in applying the above approaches is scalability of training. One way to significantly improve scalability is through preprocessing, and pruning alerts for which the (near-)optimal decision is obvious. We use the following pruning step to this end. Suppose that there is an alert type t which is generated by benign traffic with probability at most ϵ, where ϵ is very small (for example, $\epsilon = 0$, in which case alerts of type t *never* correspond to a false positive). In most realistic cases, it is nearly optimal to always inspect such alerts. Consequently, we prune all alerts with false positive rate below a small pre-defined ϵ (in our implementation below, we set $\epsilon = 0$), and mark them for inspection (correspondingly reducing the available budget for inspecting other alerts).

7.5 Experimental Results

In this section, we present case studies to investigate the robustness of our proposed approach for alert prioritization. We conduct our experiments in two applications: intrusion detection which employs a signature-based detection system and fraud detection which applies a learning-based detection system. We start with a broad introduction of the experimental methodology, including the details of the implementation of our approach and evaluation methods. We then proceed to describe each case study in detail.

7.5.1 Experimental Methodology

Implementation. The DDPG-MIX algorithm was implemented in TensorFlow [1], an open-source library for neural network learning. The architecture of the policy and value networks for both players are displayed in Table 7.2. We used Adam for learning the parameters of

Table 7.2: Architecture of the implemented policy and value networks.

Neural network	Layer	Number of units	Activation function	Initializer
Policy network	Input	T (defender); $\|T\| + \|A\| \cdot (1 + \|T\|)$ (adversary)	-	-
	Hidden	16 (Fraud detection); 32 (Intrusion detection)	Tanh	Xavier [31]
	Output	$\|T\|$ (defender); $\|A\|$ (adversary)	Sigmoid	Xavier
Value netwrok	Input	$2 \cdot \|T\|$ (defender); $\|T\| + \|A\| \cdot (2 + \|T\|)$ (adversary)	-	-
	Hidden	32 (Fraud detection); 64 (Intrusion detection)	Relu	He Normal [37]
	Output	1	Relu	He Normal

the neural networks with learning rates of 0.001 and 0.002 for the policy and value networks, respectively. The discount factor τ was set to be 0.95, and we set the size of the memory buffer to 40,000. The learning process contained 500 episodes, each with 400 learning steps. The collection of policies used in the double-oracle framework was initialized with a pair of policies that uniformly allocate each player's budget among their choices.

Our experiments were conducted on a server running Ubuntu 16.04 with Intel(R) Xeon(R) CPU E5-2695 v4 @ 2.10GHz, 18 cores and 64 GB memory. Each experiment was repeatedly executed 20 times with 20 different random seeds.

Evaluation Method. We use the expected loss of the defender (equivalently, gain of the adversary) as the metric throughout our evaluation. Specifically, for a given defense policy, we evaluate the loss of the defender using several models of the adversary. First, we used Algorithm 4 to compute the best response of the adversary, as anticipated by our approach. In addition, to evaluate the general robustness of our approach, we employed two alternative policies for the adversary: *Uniform*, a policy which uniformly distributes the adversary's budget over attack actions; and *Greedy*, a policy which allocates the budget to attacks in the order of expected adversary utility. Specifically, the *Greedy* adversary prioritizes the attack actions according to $L_a \cdot \min\{\frac{\tilde{D}}{c_a}, 1\}$, where $\tilde{D}$ is the available attack budget, adding actions in this priority order until the adversary's budget is exhausted.

We first conduct our experiments by assuming that the defender knows the adversary's capabilities. Subsequently, we evaluate the robustness of our approach when the defender is uncertain about the adversary's capabilities, and use it to provide practical guidance. Finally, we provide results on the computational cost of our approach.

7.5.2 Case Study I: Network Intrusion Detection

Our first case study involves a signature-based network intrusion detection scenario, using the Suricata, a state-of-the-art open source intrusion detection system (IDS), combined with the CICIDS2017 dataset. Our case study evaluates our alert prioritization method in two cases: i) the defender has full knowledge of the adversary; and ii) the defender is uncertain about the adversary's capabilities.

CICIDS2017 dataset. The CICIDS2017 dataset [96] records benign and malicious network flows in pcap format, captured in a real-world network between 07/03/2017 and 07/27/2017. The network consists of 10 desktops belonging to regular users and 5 laptops owned by attackers. The desktops are used to generate natural benign background traffic by using a profile system that abstracts the behaviors of regular users. The laptops are employed to produce malicious traffic of the following classes of attacks: Brute Force, Botnet, DDoS, DoS, Heartbleed, Infiltration, Portscan, and Web Attack.

Suricata IDS. We employ Suricata[12] to conduct our case study on the CICIDS2017 dataset. Suricata is an open-source network intrusion detection system which performs analysis of passing traffic on a network by using a set of signatures (also called rules). If a traffic pattern matches any of the signatures, then a corresponding alert is triggered and sent to the network administrator.

[12]Available at https://suricata-ids.org/about/open-source/.

Table 7.3: Alert types of Suricata in our experiments.

Alert type	**Description**	**Priority**
attempted-recon	Attempted Information Leak	2
attempted-user	Attempted User Privilege Gain	1
bad-unknow	Potentially Bad Traffic	2
misc-acticity	Misc activity	3
not-suspicious	Not Suspicious Traffic	3
policy-violation	Potential Corporate Privacy Violation	1
protocol-command-decode	Generic Protocol Command Decode	3
trojan-activity	A Network Trojan was Detected	1
unsuccessful-user	Unsuccessful User Privilege Gain	1
web-application-attack	Web Application Attack	1

A Suricata signature contains the following parts: *action*, *header*, *rule options*, and *priority*. *Action* describes the operation of Suricata when a signature is matched, which can be either dropping a packet or raising an alert. *Header* defines the protocol, port, and IP addresses of the source and destination in a signature. *Rule options* include a list of keywords, for example, the corresponding alert type associated with a priority. Finally, the *priority* keyword comes with a numerical value ranging from 1 to 255 where 1 indicates the highest priority and 255 the lowest.

In our experiments, we use Suricata to scan the pcap files in the CICIDS2017 dataset. Specifically, we use the *Emerging Threats Ruleset (ETR)*[13] to analyze the network traffic in the dataset. ETR defines a total of 33 alert types, and we select the 10 most common alert types exhibited during our experiments, which are shown in Table 7.3.

Experimental Setup. We use the following steps to set up our experiments for the case study. First, we used 30 minutes as the fixed length of each time period. Then, we utilized the Suricata IDS to scan and detect intrusions for both malicious and benign traffic in the CICIDS2017 data. By doing so, we obtained the number of alerts of each type raised by each attack action, as well as the number of false alerts in each time period. In the preprocessing

[13]Available at `https://rules.emergingthreats.net/open/suricata/`.

Table 7.4: Attack actions and alert types used in the case study of intrusion detection.

Attack action	**Number of each alert type raised**							E_a	L_a
	attempted-recon	attempted-user	bad-unknown	misc-activity	not-suspicious	policy-violation	protocol-command-decode		
Brute Force	1230	0	0	0	0	0	0	120	3.6
Botnet	0	4	2	106	0	54	0	60	6.0
DoS	0	0	0	0	0	24	0	74	4.0
Heartbleed	0	0	4	0	10	0	0	20	3.6
Infiltration	710	2	862	12	0	80	600	52	1.4
PortScan	138	0	320	30	0	0	0	80	1.4
Web Attack	0	0	6	0	0	0	0	62	2.7

Table 7.5: Average number of false alerts triggered in each time period

Alert type	**Avg. number of false alerts in each period**
attempted-recon	7,200
attempted-user	44,100
bad-unknown	1,600
misc-activity	7,300
not-suspicious	17,400
policy-violation	4,000
protocol-command-decode	10,200

step we pruned alert types that were triggered *only* by malicious traffic, as discussed in Section 7.4.4. As a result, we were left with 7 out of the 10 alert types to consider using our full adversarial RL framework. In addition, we filtered out the attack actions that do not raise any alerts, since those attacks will never be detected using Suricata, leaving 7 out of 8 representative attacks for our experiments. The final attack actions and alert types that we use in the experiments are given in Table 7.4.

We used Poisson distribution to fit the distribution of alerts raised by benign traffic in each time period. Since the benign traffic in the CICIDS2017 dataset was captured from only 10 desktop which is far less than the number of computers in a real-world local area network, we amplified the corresponding mean of each type of alert by a factor of 100. The resulting average numbers are shown in Table 7.5. We set the cost of investigating each alert to 1.0 (i.e., equal for all alerts). Next, we used the base score of the *Common Vulnerability Scoring System (CVSS)* to measure the loss of defender if an attack action was not detected. Specifically, we employed CVSS v3.0[14] to compute L_a for $a \in A$. Note that since the defender

[14]Available at https://www.first.org/cvss/calculator/3.0.

observes only *alerts* but not the actual attacks, alert-investigation decisions in deployment cannot directly take advantage of the CVSS scores to quantify the risk of underlying attack. However, since the ground truth is available during training and evaluation, CVSS scores are used to provide additional information on the impact of the attack. For example, the cost of mounting a Brute Force attack is 120 minutes. We document L_a (loss to the defender from a successful attack) and E_a (execution cost of the attack) for $a \in A$ in Table 7.4.

Baselines. The performance of the proposed approach is compared with two alternative policies for alert prioritization: *Uniform*, a policy which uniformly allocates the defender's budget over alert types, and *Suricata* priorities, where the defender exhausts the defense budget according to the built-in prioritization of the Suricata IDS, shown in Table 7.3. We tried two additional baselines from prior literature that use game theoretic alert prioritization: *GAIN* [56] and *RIO* [127], but these do not scale computationally to the size of our IDS case study (we compare to these in our second, smaller, case study). We did not compare to alert correlation methods for reducing the number of false alerts, since these techniques are entirely orthogonal and complementary to our setting (we address the issue of limited alert inspection budget in the face of false alerts, whatever means are used to generate alerts). Throughout, we refer to our proposed approach as *ARL*.

Results. Figure 7.4 presents our evaluation of the robustness of alert prioritization approaches when the defender knows the adversary's capabilities, and the results suggest that our approach significantly outperforms the other baselines. Specifically, the proposed approach is 50% better than the Uniform policy, which in turn is significantly better than using Suricata priorities. There are a few reasons why deterministic priority-based approaches perform so poorly. First, determinism allows attackers to easily circumvent the policy by focusing on attacks that trigger alerts which are rarely inspected. Moreover, such naive deterministic policies also fail to exploit the empirical relationships between attacks and alerts they tend to

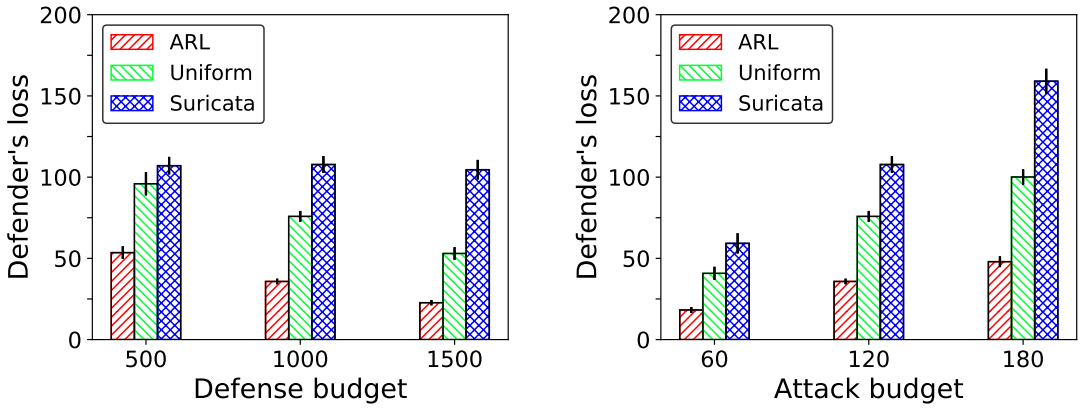

Figure 7.4: Intrusion detection: loss of the defender when it knows the attack budget. Left: Defender's loss for different defense budgets, with attack budget fixed at 120. Right: Defender's loss for different attack budgets, with defense budget fixed at 1000.

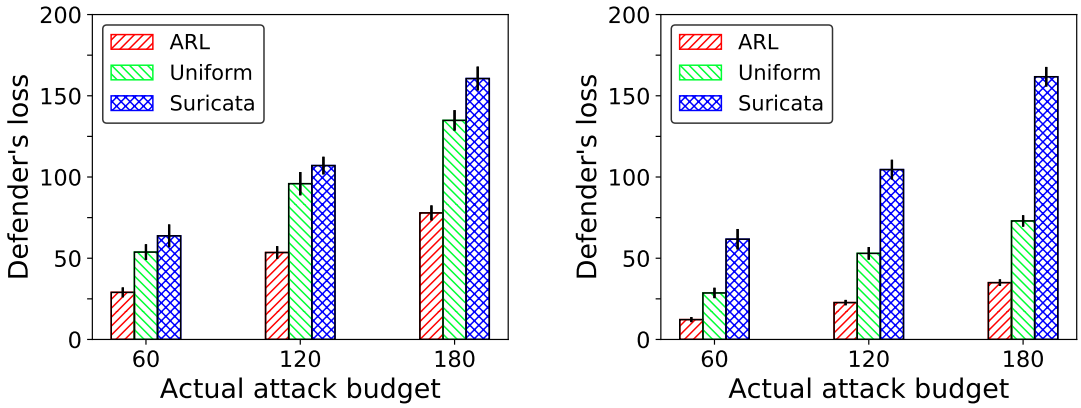

Figure 7.5: Intrusion detection: loss of the defender when it is uncertain of the attack budget. Left: def. budget=500. Right: def. budget=1500. The defender's estimate of the attack budget is 120 in all cases. Thus, if the actual attack budget is 60, then the defender overestimates the adversary's budget; if the actual attack budget is 180, then it is underestimated by the defender.

trigger: for example, if an attack triggers multiple alerts, but one of these alert types happens to have very few alerts in current logs, static priority-based policies will not leverage this structure. In contrast, by learning a *policy* of alert inspection which maps arbitrary alert observations to a decision about which to inspect, we can make decisions at a significantly finer granularity.

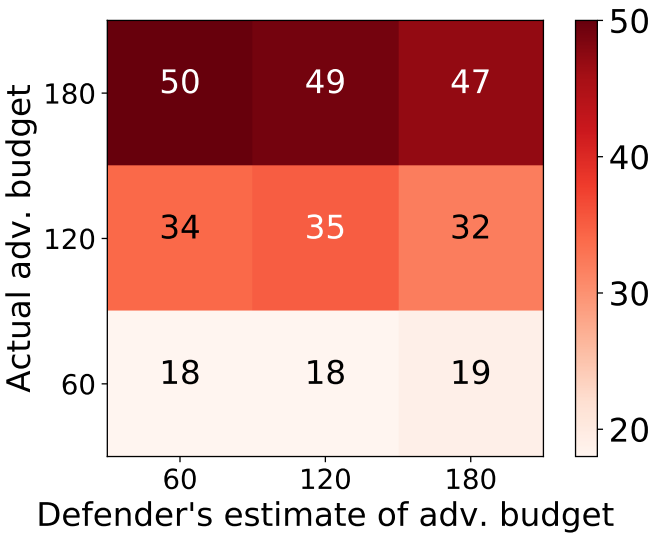

Figure 7.6: Intrusion detection: loss of the defender when it has different estimates of the attack budget.

Evaluating the alert prioritization methods when the defender is uncertain about the attack budget (Figures 7.5 and 7.6), we can observe that the proposed ARL approach still achieves the lowest defender loss both when the attack budget is underestimated and when overestimated, and it is still far better than the baselines. In addition, Figure 7.6 shows that when the attack budget is underestimated or overestimated, there is only a 5% performance degradation compared to when the defender has full knowledge of the adversary. This demonstrates that our approach remains robust to a strategic adversary even when the defender does not precisely know the adversary's capabilities. Moreover, in this domain we can see that neither over- nor underestimating adversary's budget is particularly harmful, although overestimation appears to be slightly better.

Our final consideration is the impact of uncertainty about the adversary's rationality (Figure 7.7). Specifically, we now study how our approach performs, compared to the baselines, if the adversary is in some way myopic, either using a simple uniform strategy (*Uniform*) or greedily choosing attacks in order of impact (*Greedy*). We can observe that although

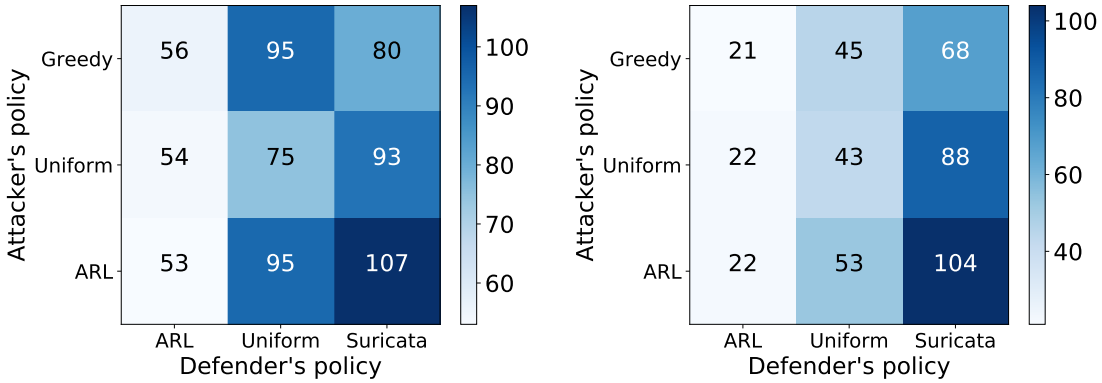

Figure 7.7: Intrusion detection: loss of the defender when it is certain of the attack budget but is uncertain of the attack policy. The attack budget is fixed as 120. Left: def. budget=500. Right: def. budget=1500.

we assume a very strong adversary, our ARL approach significantly outperforms the other baselines even when the adversary is using a different attack policy.

7.5.3 Case Study II: Fraud Detection

While NIDS settings are a natural fit for our approach, we now demonstrate its generalizability by considering a very different problem in which our goal is to identify fraudulent credit card transactions. Just as with the first case study, we will present the results first when the defender has full knowledge of the adversary's capabilities, and subsequently study the impact of defender's uncertainty about these.

Fraud dataset. The fraud dataset[15] contains 284,807 credit card transactions, of which 482 are fraudulent. Each transaction is represented by a vector of 30 numerical features, 28 of which are transformed using *Principle Component Analysis (PCA)*. In addition, each feature vector is associated with a binary label indicating the type of transaction (regular or fraudulent). In order to make it meaningful in our context, we cluster the set of fraudulent

[15]Available at: https://www.kaggle.com/mlg-ulb/creditcardfraud.

Table 7.6: Number of transactions in the modified fraud dataset

Original transaction type	**Label**	**Count**
Genuine	0	284,308
Fraudulent	1	11
	2	21
	3	72
	4	250
	5	14
	6	124

transactions into n subsets, indicating a type of attack, using a Gaussian Mixture model [8]. In our experiments, we set $n = 6$, and modify the dataset with fraudulent labels replaced by cluster assignments. The counts of each type of transaction is shown in Table 7.6.

Learning-based fraud detector. We developed a fraud detector using supervised learning on the fraud dataset. The main challenge is that the dataset is highly imbalanced, as shown in Table 7.6: the fraudulent transactions only account for $< 0.2\%$ of all transactions. To address this challenge, we apply *Synthetic Minority Over-sampling Technique (SMOTE)* to produce synthetic data for the minority classes to balance the data. Our implementation contains the following steps:

(i) *Dataset splitting*: We use stratified split to partition the modified fraud dataset into training and test data with equal size, which contain roughly the same proportions of the fraudulent and non-fraudulent data.

(ii) *Binary classification:* We use SMOTE and linear SVM to learn a binary classifier to predict whether a transaction is fraudulent. The resulting classifier has an AUC >99% and a recall >90% on the test data, which indicates that more than 90% of the fraudulent transactions can be detected.

(iii) *Multi-class classification:* We now restrict attention to only the fraudulent transactions to learn a conditional classifier to predict the type of fraud. Specifically, we learn 6 independent classifiers each of which corresponds to one fraud type and returns a binary classification result indicating whether a fraudulent transaction belongs to this type. Similarly to Step (ii), we use SMOTE and linear SVM to learn these classifiers, each of which admits $> 94\%$ recall.

Once the fraud detector is implemented, we evaluate the detector using the test dataset. We first predict the test data by using the binary classifier obtained in Step (ii) above. If any transaction in the test data is classified as fraudulent, then it is further inspected by the 6 classifiers we construct for multi-class classification. If a fraudulent transaction is predicted as any type of fraud, then a corresponding alert is triggered. Otherwise, an alert corresponding to the fraud type which is predicted with the highest classification score is triggered.

Experimental Setup. To evaluate the robustness of the proposed approach for alert prioritization in fraud detection, we first computed the distributions of the true and false alerts identified by the fraud detector that we implemented. By doing so, we obtained the probability that any attack $a \in A$ triggers an alert $t \in T$, as well as the number of false alarms associated with each type of alert, each of which has a value of 1 as the investigation cost. We filtered out alert types that were triggered *only* by fraudulent transactions (as we had done before), leaving 3 out of 6 alert types. We also filtered out the attack actions which are associated with the alert types omitted above, as these attacks can always be detected by investigating the corresponding alerts. The resulting distribution of the alerts triggered by frauds is given in Table 7.7.

We used $[1, 3, 2]$ as the adversary's cost of the mounting each type of attack action. We employed the mean amount of each type of fraudulent transaction as the loss of the defender if any such type of attack action is not detected, measured by the unit of 10 Euros. The

Table 7.7: Probability that an attack action triggers each type of alert

Attack action	**Alert type**		
	1	2	3
1	0.9	0.61	0
2	0.09	0.87	0.12
3	0	0.41	0.85

corresponding defender's loss for each undetected attack was $[9.4, 12.1, 16.0]$. In addition, we used 30 minutes as the fixed length of each time period in our experiments. Based on our classification results, the average number of false alerts that occur of each type in a time period was $[10, 47, 39]$. Similar to our IDS case study, we simulated the distribution of false alerts by using Poisson processes with the above mean values.

Baselines. The performance of the proposed approach is investigated by comparing with three alternative policies for alert prioritization: *Uniform*, a policy which uniformly allocates the defender's budget over each alert type; *GAIN* [56], a game theoretic approach which prioritizes alert types, and always inspects all alerts of a selected type; and *RIO* [127], another game theoretic approach which prioritizes alerts, and computes an approximately optimal number of alerts of each type to inspect.

Results. Figure 7.8 shows the results when the defender has full knowledge of the adversary's capabilities. We can observe that the proposed approach (*ARL*) outperforms other baselines in all settings, typically by at least 25%. The main reason for the advantage is similar to that in the IDS setting: the ability to have a policy that is carefully optimized and conditional on state significantly increases its efficiency. Interestingly, the alternative game theoretic alert prioritization approaches, GAIN and RIO, are in some cases worse than the uniformly random policy. The key reason is that they can be myopic in that they independently optimize for a single time period, whereas attacks can be adaptive. The proposed approach, in contrast, explicitly considers such adaptivity.

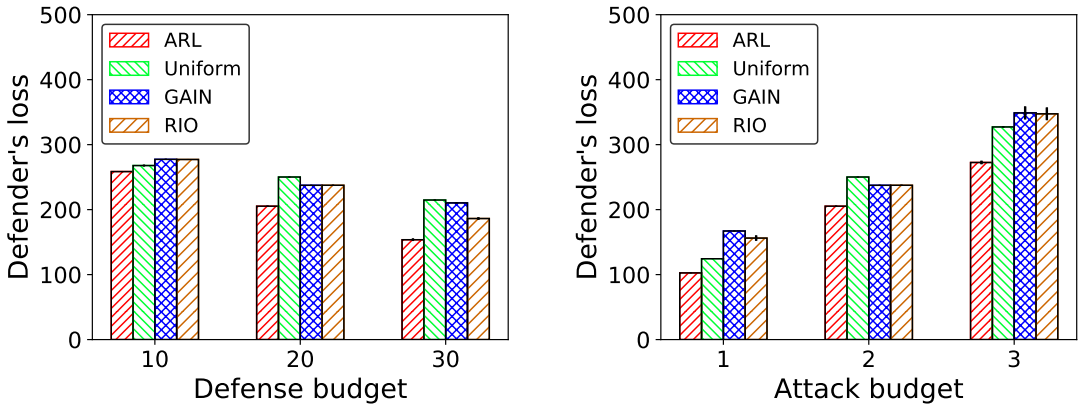

Figure 7.8: Fraud detection: loss of the defender when it knows the attack budget. Left: Defender's loss by its budget, with attack budget adv_budget being fixed as 2. Right: Defender's loss by attack budget, with defense budget def_budget being fixed as 20.

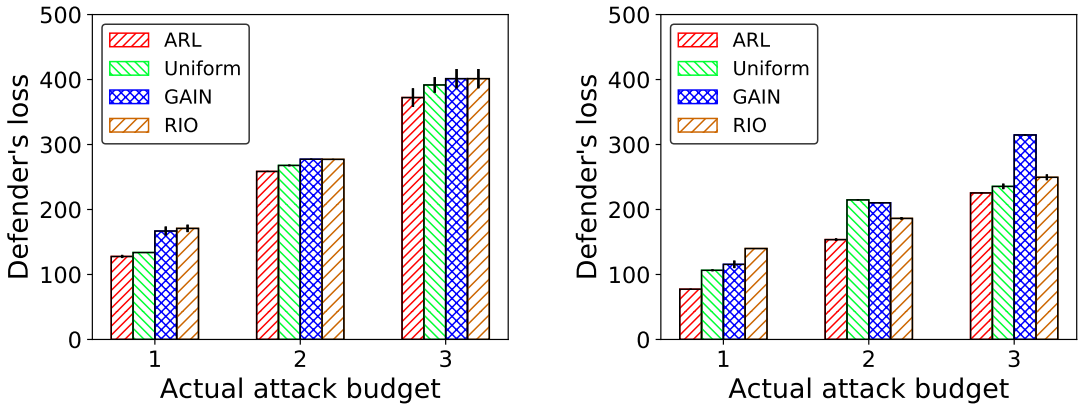

Figure 7.9: Fraud detection: loss of the defender when it is uncertain of the attack budget. Left: def. budget=10. Right: def. budget=30. The defender's estimate of the attack budget is 2. If the actual attack budget is 1, then the defender overestimates the adversary's budget; if the actual attack budget is 3, then it is underestimated.

Figures 7.9 and 7.10 investigate performance of our approach when the attack budget is uncertain. It can be seen in Figure 7.9 that ARL remains the best approach to use, despite this uncertainty. Interestingly, *GAIN* can, in contrast, be rather fragile to such uncertainty. Considering Figure 7.10, both under- and overestimation of the attack budget incurs a limited performance impact ($< 10\%$). More interesting, however, is the observation that it is actually better to slightly *underestimate* the adversary's budget: in the worst case, this

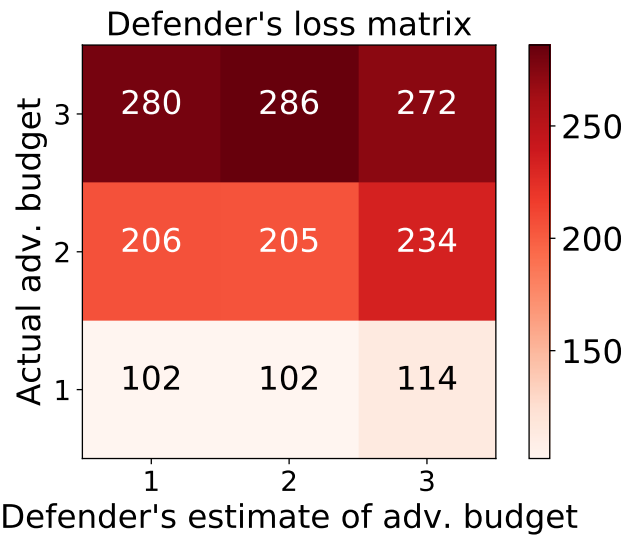

Figure 7.10: Fraud detection: loss of the defender when it has different estimates of the attack budget.

hurts performance less than 3%. Effectively, the approach remains quite robust even against stronger attacks, whereas overestimating the budget does not take sufficient advantage of weaker adversaries.

Finally, we study the robustness of ARL compared to other baselines when the attacker is using different policies (*Uniform* or *Greedy*) instead of the RL-based policy that is assumed by our approach (Figure 7.11). Here, the results are slightly more ambiguous than we observed in the IDS domain: when the adversary is using the *Greedy* policy, *RIO* does outperform ARL by 8% when the defender's budget is small, and by 13% when the defender's budget is large. However, in these cases, the adversary can gain a great deal by more carefully designing its policy. Thus, when the defender's budget is large, a rational adversary can cause *RIO* to degrade by nearly 18%, where ARL is quite robust to such adversaries.

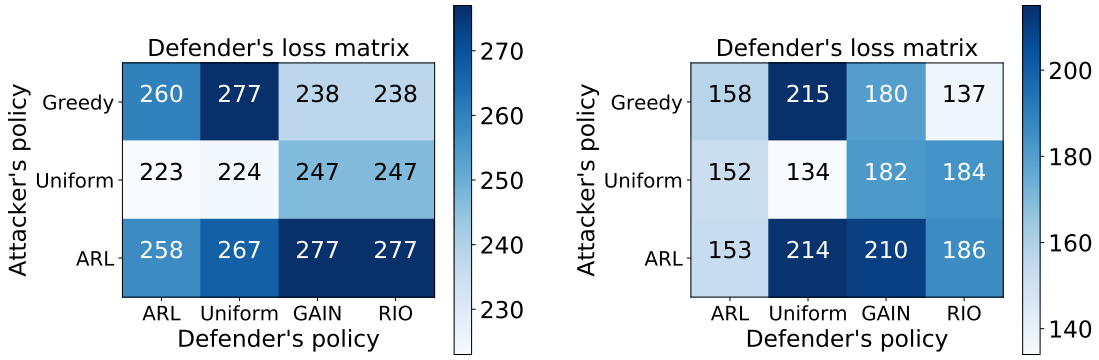

Figure 7.11: Fraud detection: loss of the defender when it is certain of the attack budget but is uncertain of the attack policy. The attack budget is fixed as 2. Left: def. budget=10. Right: def. budget=30.

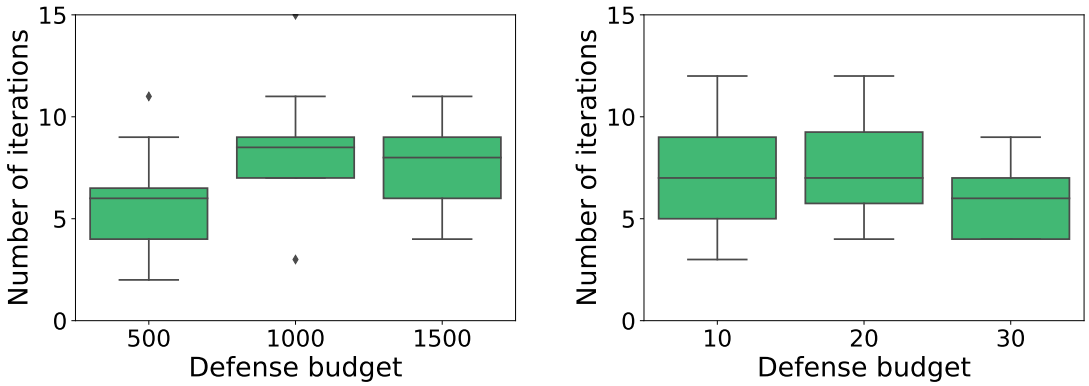

Figure 7.12: Computational cost. Left: Number of double oracle iterations in network intrusion detection with adv. budget=120. Right: Number of double oracle iterations in fraud detection with adv. budget=2.

7.5.4 Computational Cost

Figure 7.12 presents our evaluation of the computational cost of the proposed alert prioritization approach. The results show that the double oracle algorithm can converge very fast in practice, with fewer than 15 iterations in most cases; indeed, in the vast majority of instances we need fewer than 10 iterations.

Another interesting observation is non-monotonicity of convergence time (in terms of iterations) as we increase the defense budget. In the IDS setting, for example, increasing the defense budget increases the number of iterations when we go from a budget of 500 to 1000, but the computational cost remains stable as we further increase the budget to 1500. In contrast, in the fraud detection case study, increasing the budget from 10 to 20 has little impact on the number of iterations, but further increasing it to 30 actually *reduces* the number of iterations necessary for convergence. To understand this phenomenon, note that increasing the defender's budget has two opposing effects: on the one hand, the search space for the defender increases significantly, but on the other hand, it may become much easier to compute a near-optimal defense with a larger budget (for example, with a large enough budget, we can almost always inspect all alerts).

7.6 Conclusion

Since even after applying techniques for reducing the alert burden (*e.g.*, alert correlation) there often remain vastly more alerts than time to investigate them, the success of detection often hinges on how defenders prioritize certain alerts over others. In practice, prioritization is typically based on non-strategic heuristics (*e.g.*, Suricata's built-in priority values), which may easily be exploited by a strategic attacker who can adapt to the prioritization. Strategic prioritization approaches attempt to prevent this by using game-theory to capture adaptive attackers; however, existing strategic approaches severely restrict the defender's policy (e.g., strict prioritization) for the sake of computational tractability.

In contrast,in this chapter we introduced a general model of alert prioritization that does not impose any restrictions on the defender's policy, and we proposed a novel double oracle and reinforcement learning based approach for finding approximately optimal prioritization policies

efficiently. Our experimental results—based on case studies of IDS and fraud detection—demonstrate that these policies significantly outperform non-strategic prioritization and prior game-theoretic approaches. Further, to demonstrate the strength of our attacker model, we also showed that the attacker policies found by our approach outperform multiple baseline policies.

For practitioners, the key task in applying our approach is estimating the parameter values of our model. In our case studies, we showed how to estimate parameters in two domains (*e.g.*, for NIDS, using CVSS score to estimate attack impact and CVSS complexity for attack cost). The most difficult parameter to estimate is the attacker's budget; however, our experimental results show that our approach is robust to uncertainty in the attacker's budget and outperforms other approaches even when the budget is misestimated. We leave studying the sensitivity to other parameters to future work.

Part IV

Robust Decentralized Learning Ecosystem

Chapter 8

One VS. Many: Adversarial Regression with Multiple Learners

In previous chapters, we have investigated robust machine learning with a single learner against an adversary. However, in many situations an adversary's decision is aimed at a collection of learners, rather than specifically targeted at each independently. In this chapter, we study the problem of adversarial linear regression with multiple learners. We first approximate the resulting game by exhibiting an upper bound on learner loss functions, and show that the resulting game has a unique symmetric equilibrium. We then present an algorithm for computing this equilibrium, and show through extensive experiments that equilibrium models are significantly more robust than conventional regularized linear regression.

8.1 Overview

In this chapter, we investigate the problem of adversarial regression with a collection of learners and a single adversary. We model the resulting game as an interaction between multiple

learners, who simultaneously learn linear regression models, and an attacker, who observes the learned models (as in white-box attacks [103]), and modifies the original feature vectors at test time in order to induce incorrect predictions. Crucially, rather than customizing the attack to each learner (as in typical models), the attacker chooses a single attack for *all* learners. We term the resulting game a *Multi-Learner Stackelberg Game*, to allude to its two stages, with learners jointly acting as Stackelberg leaders, and the attacker being the follower. Our first contribution is the formal model of this game. Our second contribution is to approximate this game by deriving upper bounds on the learner loss functions. The resulting approximation yields a game in which there always exists a symmetric equilibrium, and this equilibrium is unique. In addition, we prove that this unique equilibrium can be computed by solving a convex optimization problem. Our third contribution is to show that the equilibrium of the approximate game is robust, both theoretically (by showing it to be equivalent to a particular robust optimization problem), and through extensive experiments, which demonstrate it to be much more robust to attacks than standard regularization approaches.

8.2 Model

We investigate the interactions between a collection of learners $\mathcal{N} = \{1, 2, ..., n\}$ and an attacker in regression problems, modeled as a *Multi-Learner Stackelberg Game (MLSG)*. At the high level, this game involves two stages: first, all learners choose (train) their models from data, and second, the attacker transforms test data (such as features of the environment, at prediction time) to achieve malicious goals. Below, we first formalize the model of the learners and the attacker, and then formally describe the full game.

8.2.1 Modeling the Players

At training time, a set of training data $(\mathbf{X}, \mathbf{y})$ is drawn from an unknown distribution $\mathcal{D}$. $\mathbf{X} \in \mathbb{R}^{m \times d}$ is the training sample and $\mathbf{y} \in \mathbb{R}^{m \times 1}$ is a vector of values of each data in $\mathbf{X}$. We let $\mathbf{x}_j \in \mathbb{R}^{d \times 1}$ denote the jth instance in the training sample, associated with a corresponding value $y_j \in \mathbb{R}$ from $\mathbf{y}$. Hence, $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_m]^\top$ and $\mathbf{y} = [y_1, y_2, ..., y_m]^\top$. On the other hand, test data can be generated either from $\mathcal{D}$, the same distribution as the training data, or from $\mathcal{D}^{'}$, a modification of $\mathcal{D}$ generated by an attacker. The nature of such malicious modifications is described below. We let β $(0 \leq \beta \leq 1)$ represent the probability that a test instance is drawn from $\mathcal{D}^{'}$ (i.e., the malicious distribution), and $1 - \beta$ be the probability that it is generated from $\mathcal{D}$.

The action of the ith learner is to select a $d \times 1$ vector $\boldsymbol{\theta}_i$ as the parameter of the linear regression function $\hat{\mathbf{y}}_i = \mathbf{X}\boldsymbol{\theta}_i$, where $\hat{\mathbf{y}}_i$ is the predicted values for data $\mathbf{X}$. The expected cost function of the ith learner at test time is then

$$\begin{aligned} c_i(\boldsymbol{\theta}_i, \mathcal{D}^{'}) = \beta \mathbb{E}_{(\mathbf{X}^{'}, \mathbf{y}) \sim \mathcal{D}^{'}}[\ell(\mathbf{X}^{'}\boldsymbol{\theta}_i, \mathbf{y})] \\ + (1 - \beta)\mathbb{E}_{(\mathbf{X}, \mathbf{y}) \sim \mathcal{D}}[\ell(\mathbf{X}\boldsymbol{\theta}_i, \mathbf{y})]. \end{aligned} \tag{8.1}$$

where $\ell(\hat{\mathbf{y}}, \mathbf{y}) = ||\hat{\mathbf{y}} - \mathbf{y}||_2^2$. That is, the cost function of a learner i is a combination of its expected cost from both the attacker and the honest source.

Every instance $(\mathbf{x}, y)$ generated according to $\mathcal{D}$ is, with probability β, maliciously modified by the attacker into another, $(\mathbf{x}', y)$, as follows. We assume that the attacker has an instance-specific target $z(\mathbf{x})$, and wishes that the prediction made by each learner i on the modified instance, $\hat{y} = \boldsymbol{\theta}_i^\top \mathbf{x}^{'}$, is close to this target. We measure this objective for the attacker by $\ell(\hat{\mathbf{y}}, \mathbf{z}) = ||\hat{\mathbf{y}} - \mathbf{z}||_2^2$ for a vector of predicted and target values $\hat{\mathbf{y}}$ and $\mathbf{z}$, respectively. In

addition, the attacker incurs a cost of transforming a distribution $\mathcal{D}$ into $\mathcal{D}^{'}$, denoted by $R(\mathcal{D}^{'}, \mathcal{D})$.

After a dataset $(\mathbf{X}^{'}, \mathbf{y})$ is generated in this way by the attacker, it is used simultaneously against all the learners. This is natural in most real attacks: for example, spam templates are commonly generated to be used broadly, against many individuals and organizations, and, similarly, malicious executable programs are often produced to be generally effective, rather than custom made for each target. The expected cost function of the attacker is then a sum of its total expected cost for all learners plus the cost of transforming $\mathcal{D}$ into $\mathcal{D}^{'}$ with coefficient $\lambda > 0$:

$$c_a(\{\boldsymbol{\theta}_i\}_{i=1}^n, \mathcal{D}^{'}) = \sum_{i=1}^{n} \mathbb{E}_{(\mathbf{X}^{'}, \mathbf{y}) \sim \mathcal{D}^{'}}[\ell(\mathbf{X}^{'}\boldsymbol{\theta}_i, \mathbf{z})] + \lambda R(\mathcal{D}^{'}, \mathcal{D}). \tag{8.2}$$

As is typical, we estimate the cost functions of the learners and the attacker using training data $(\mathbf{X}, \mathbf{y})$, which is also used to simulate attacks. Consequently, the cost functions of each learner and the attacker are estimated by

$$c_i(\boldsymbol{\theta}_i, \mathbf{X}^{'}) = \beta\ell(\mathbf{X}^{'}\boldsymbol{\theta}_i, \mathbf{y}) + (1-\beta)\ell(\mathbf{X}\boldsymbol{\theta}_i, \mathbf{y}) \tag{8.3}$$

and

$$c_a(\{\boldsymbol{\theta}_i\}_{i=1}^n, \mathbf{X}^{'}) = \sum_{i=1}^{n} \ell(\mathbf{X}^{'}\boldsymbol{\theta}_i, \mathbf{z}) + \lambda R(\mathbf{X}^{'}, \mathbf{X}) \tag{8.4}$$

where the attacker's modification cost is measured by $R(\mathbf{X}^{'}, \mathbf{X}) = ||\mathbf{X}^{'} - \mathbf{X}||_F^2$, the squared Frobenius norm.

8.2.2 The Multi-Learner Stackerlberg Game

We are now ready to formally define the game between the n learners and the attacker. The MLSG has two stages: in the first stage, learners simultaneously select their model parameters $\boldsymbol{\theta}_i$, and in the second stage, the attacker makes its decision (manipulating $\mathbf{X}^{'}$) after observing the learners' model choices $\{\boldsymbol{\theta}_i\}_{i=1}^{n}$. We assume that the proposed game satisfies the following assumptions:

1. The learners have complete information about parameters β, λ and $\mathbf{z}$. This is a strong assumption, and we relax it in our experimental evaluation (Section 8.6), providing guidance on how to deal with uncertainty about these parameters.

2. Each learner has the same action (model parameter) space $\boldsymbol{\Theta} \subseteq \mathbb{R}^{d\times 1}$ which is nonempty, compact and convex. The action space of the attacker is $\mathbb{R}^{m\times d}$.

3. The columns of the training data $\mathbf{X}$ are linearly independent.

We use *Multi-Learner Stackelberg Equilibrium* (MLSE) as the solution for the MLSG, defined as follows.

Definition 1 (Multi-Learner Stackelberg Equilibrium (MLSE))**.** *An action profile* $(\{\boldsymbol{\theta}_i^*\}_{i=1}^{n}, \mathbf{X}^*)$ *is an MLSE if it satisfies*

$$\begin{aligned} \boldsymbol{\theta}_i^* &= \arg\min_{\boldsymbol{\theta}_i \in \boldsymbol{\Theta}} c_i(\boldsymbol{\theta}_i, \mathbf{X}^*(\boldsymbol{\theta})), \forall i \in \mathcal{N} \\ \texttt{s.t.} \quad \mathbf{X}^*(\boldsymbol{\theta}) &= \arg\min_{\mathbf{X}' \in \mathbb{R}^{m\times d}} c_a(\{\boldsymbol{\theta}_i\}_{i=1}^{n}, \mathbf{X}^{'}). \end{aligned} \tag{8.5}$$

where $\boldsymbol{\theta} = \{\boldsymbol{\theta}_i\}_{i=1}^{n}$ *constitutes the joint actions of the learners.*

At the high level, the MLSE is a blend between a Nash equilibrium (among all learners) and a Stackelberg equilibrium (between the learners and the attacker), in which the attacker plays a best response to the *observed* models $\boldsymbol{\theta}$ chosen by the learners, and given this behavior by the attacker, all learners' models $\boldsymbol{\theta}_i$ are mutually optimal.

The following lemma characterizes the best response of the attacker to arbitrary model choices $\{\boldsymbol{\theta}_i\}_{i=1}^n$ by the learners.

Lemma 1 (Best Response of the Attacker)**.** *Given $\{\boldsymbol{\theta}_i\}_{i=1}^n$, the best response of the attacker is*

$$\mathbf{X}^* = (\lambda \mathbf{X} + \mathbf{z} \sum_{i=1}^{n} \boldsymbol{\theta}_i^\top)(\lambda \mathbf{I} + \sum_{i=1}^{n} \boldsymbol{\theta}_i \boldsymbol{\theta}_i^\top)^{-1}. \tag{8.6}$$

Proof. We derive the best response of the attacker by using the first order condition. Let $\nabla_{X'} c_a(\{\boldsymbol{\theta}_i\}_{i=1}^n, \mathbf{X}')$ denote the gradient of c_a with respect to $\mathbf{X}'$. Then

$$\nabla_{X'} c_a = 2 \sum_{i=1}^{n} (\mathbf{X}' \boldsymbol{\theta}_i - \mathbf{z}) \boldsymbol{\theta}_i^\top + 2\lambda (\mathbf{X}' - X).$$

Due to convexity of c_a, let $\nabla_{X'} c_a = \mathbf{0}$, we have

$$\mathbf{X}^* = (\lambda \mathbf{X} + \mathbf{z} \sum_{i=1}^{n} \boldsymbol{\theta}_i^\top)(\lambda \mathbf{I} + \sum_{i=1}^{n} \boldsymbol{\theta}_i \boldsymbol{\theta}_i^\top)^{-1}.$$

□

Lemma 1 shows that the best response of the attacker, $\mathbf{X}^*$, has a closed form solution, as a function of learner model parameters $\{\boldsymbol{\theta}_i\}_{i=1}^n$. Let $\boldsymbol{\theta}_{-i} = \{\boldsymbol{\theta}_j\}_{j \neq i}$, then $c_i(\boldsymbol{\theta}_i, \mathbf{X}^*)$ in Eq. (8.5)

can be rewritten as

$$c_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i}) = \beta\ell(\mathbf{X}^*(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i})\boldsymbol{\theta}_i, \mathbf{y}) + (1-\beta)\ell(\mathbf{X}\boldsymbol{\theta}_i, \mathbf{y}). \tag{8.7}$$

Using Eq. (8.7), we can then define a *Multi-Learner Nash Game (MLNG)*:

Definition 2 (Multi-Learner Nash Game (MLNG))**.** *A static game, denoted as* $\langle \mathcal{N}, \boldsymbol{\Theta}, (c_i)\rangle$ *is a* Multi-Learner Nash Game *if*

1. *The set of players is the set of learners* $\mathcal{N}$*,*
2. *the cost function of each learner* i *is* $c_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i})$ *defined in Eq.* (8.7)*,*
3. *all learners simultaneously select* $\boldsymbol{\theta}_i \in \boldsymbol{\Theta}$*.*

We can then define *Multi-Learner Nash Equilibrium (MLNE)* of the game $\langle \mathcal{N}, \boldsymbol{\Theta}, (c_i)\rangle$:

Definition 3 (Multi-Learner Nash Equilibrium (MLNE))**.** *An action profile* $\boldsymbol{\theta}^* = \{\boldsymbol{\theta}_i^*\}_{i=1}^n$ *is a* Multi-Learner Nash Equilibrium *of the MLNG* $\langle \mathcal{N}, \boldsymbol{\Theta}, (c_i)\rangle$ *if it is the solution of the following set of coupled optimization problem:*

$$\min_{\boldsymbol{\theta}_i \in \boldsymbol{\Theta}} c_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i}), \forall i \in \mathcal{N}. \tag{8.8}$$

Combining the results above, the following result is immediate.

Theorem 1. *An action profile* $(\{\boldsymbol{\theta}_i^*\}_{i=1}^n, \mathbf{X}^*)$ *is an MLSE of the multi-learner Stackelberg game if and only if* $\{\boldsymbol{\theta}_i^*\}_{i=1}^n$ *is a MLNE of the multi-learner Nash game* $\langle \mathcal{N}, \boldsymbol{\Theta}, (c_i)\rangle$*, with* $\mathbf{X}^*$ *defined in Eq.* (8.6) *for* $\boldsymbol{\theta}_i = \boldsymbol{\theta}_i^*, \forall i \in \mathcal{N}$*.*

Theorem 1 shows that we can reduce the original $(n+1)$-player Stackelberg game to an n-player simultaneous-move game $\langle \mathcal{N}, \boldsymbol{\Theta}, (c_i) \rangle$. In the remaining sections, we focus on analyzing the Nash equilibrium of this multi-learner Nash game.

8.3 Theoretical Analysis

In this section, we analyze the game $\langle \mathcal{N}, \boldsymbol{\Theta}, (c_i) \rangle$. As presented in Eq. (8.6), there is an inverse of a complicated matrix to compute the best response of the attacker. Hence, the cost function $c_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i})$ shown in Eq. (8.7) is intractable. To address this challenge, we first derive a new game, $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i) \rangle$ with tractable cost function for its players, to approximate $\langle \mathcal{N}, \boldsymbol{\Theta}, (c_i) \rangle$. Afterward, we analyze existence and uniqueness of the *Nash Equilibrium* of $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i) \rangle$.

8.3.1 Approximation of The Game

We start our analysis by computing $(\lambda \mathbf{I} + \sum_{i=1}^{n} \boldsymbol{\theta}_i \boldsymbol{\theta}_i^\top)^{-1}$ presented in Eq. (8.6). Let matrix $\mathbf{A}_n = \lambda \mathbf{I} + \sum_{i=1}^{n} \boldsymbol{\theta}_i \boldsymbol{\theta}_i^\top$, and $\mathbf{A}_{-i} = \lambda \mathbf{I} + \sum_{j \neq i} \boldsymbol{\theta}_j \boldsymbol{\theta}_j^\top$. Then, $\mathbf{A}_n = \mathbf{A}_{-i} + \boldsymbol{\theta}_i \boldsymbol{\theta}_i^\top$. Similarly, let matrix $\mathbf{B}_n = \lambda \mathbf{X} + \mathbf{z} \sum_{i=1}^{n} \boldsymbol{\theta}_i^\top$, and $\mathbf{B}_{-i} = \lambda \mathbf{X} + \mathbf{z} \sum_{j \neq i} \boldsymbol{\theta}_j^\top$, which implies that $\mathbf{B}_n = \mathbf{B}_{-i} + \mathbf{z} \boldsymbol{\theta}_i^\top$ The best response of the attacker can then be rewritten as $\mathbf{X}^* = \mathbf{B}_n \mathbf{A}_n^{-1}$. We then obtain the following results.

Lemma 2. *$\mathbf{A}_n$ and $\mathbf{A}_{-i}$ satisfy*

1. *$\mathbf{A}_n$ and $\mathbf{A}_{-i}$ are invertible, and the corresponding invertible matrices, $\mathbf{A}_n^{-1}$ and $\mathbf{A}_{-i}^{-1}$, are positive definite.*
2. $\mathbf{A}_n^{-1} = \mathbf{A}_{-i}^{-1} - \frac{\mathbf{A}_{-i}^{-1} \boldsymbol{\theta}_i \boldsymbol{\theta}_i^\top \mathbf{A}_{-i}^{-1}}{1 + \boldsymbol{\theta}_i^\top \mathbf{A}_{-i}^{-1} \boldsymbol{\theta}_i}$.
3. $\boldsymbol{\theta}_i^\top \mathbf{A}_{-i}^{-1} \boldsymbol{\theta}_i \leq \frac{1}{\lambda} \boldsymbol{\theta}_i^\top \boldsymbol{\theta}_i$.

Proof. The proof is included in the Appendix C.1. □

Lemma 2 allows us to relax $\ell(\mathbf{X}^*(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i})\boldsymbol{\theta}_i, \mathbf{y})$ as follows:

Lemma 3.

$$\begin{aligned}\ell(\mathbf{X}^*(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i})\boldsymbol{\theta}_i, \mathbf{y}) \leq \ell(\mathbf{B}_{-i}\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i, \mathbf{y}) \\ + \frac{1}{\lambda^2}||\mathbf{z}-\mathbf{y}||_2^2(\boldsymbol{\theta}_i^\top \boldsymbol{\theta}_i)^2.\end{aligned} \tag{8.9}$$

Proof. The proof is included in the Appendix C.2. □

Note that in Eq. (8.9), $\mathbf{B}_{-i}$ and $\mathbf{A}_{-i}$ only depend on $\{\boldsymbol{\theta}_j\}_{j\neq i}$. Hence, the RHS of Eq. (8.9) is a strictly convex function with respect to $\boldsymbol{\theta}_i$. Lemma 3 shows that $\ell(\mathbf{X}^*(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i})\boldsymbol{\theta}_i, \mathbf{y})$ can be relaxed by moving $\boldsymbol{\theta}_i$ out of $\mathbf{X}^*(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i})$ and adding a regularizer $(\boldsymbol{\theta}_i^\top \boldsymbol{\theta}_i)^2$ with its coefficient $\frac{||\mathbf{z}-\mathbf{y}||_2^2}{\lambda^2}$. Motivated by this method, we iteratively relax $\ell(\mathbf{X}^*(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i})\boldsymbol{\theta}_i, \mathbf{y})$ by adding corresponding regularizers. We now identify a tractable upper bound function for $c_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i})$.

Theorem 2.

$$\begin{aligned}c_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i}) &\leq \bar{c}_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i}) \\ &= \ell(\mathbf{X}\boldsymbol{\theta}_i, \mathbf{y}) + \frac{\beta}{\lambda^2}||\mathbf{z}-\mathbf{y}||_2^2 \sum_{j=1}^{n}(\boldsymbol{\theta}_j^\top \boldsymbol{\theta}_i)^2 + \epsilon,\end{aligned} \tag{8.10}$$

where ϵ is a positive constant and $\epsilon < +\infty$.

Proof. We prove by extending the results in Lemma 3 and iteratively relaxing the cost function. The details are included in Appendix C.3. □

As represented in Eq. (8.10), $\bar{c}_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i})$ is strictly convex with respect to $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j (\forall j \neq i)$. We then use the game $\langle \mathcal{N}, \boldsymbol{\Theta}, (\bar{c}_i) \rangle$ as an approximation of $\langle \mathcal{N}, \boldsymbol{\Theta}, (c_i) \rangle$. Let

$$\begin{aligned} \widetilde{c}_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i}) &= \bar{c}_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i}) - \epsilon \\ &= \ell(\mathbf{X}\boldsymbol{\theta}_i, \mathbf{y}) + \frac{\beta}{\lambda^2} ||\mathbf{z} - \mathbf{y}||_2^2 \sum_{j=1}^{n} (\boldsymbol{\theta}_j^\top \boldsymbol{\theta}_i)^2, \end{aligned} \tag{8.11}$$

then $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i) \rangle$ has the same Nash equilibrium with $\langle \mathcal{N}, \boldsymbol{\Theta}, (\bar{c}_i) \rangle$ if one exists, as adding or deleting a constant term does not affect the optimal solution. Hence, we use $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i) \rangle$ to approximate $\langle \mathcal{N}, \boldsymbol{\Theta}, (c_i) \rangle$, and analyze the Nash equilibrium of $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i) \rangle$ in the remaining sections.

8.3.2 Existence of Nash Equilibrium

As introduced in Section 8.2, each learner has identical action spaces, and they are trained with the same dataset. We exploit this symmetry to analyze the existence of a Nash equilibrium of the approximation game $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i) \rangle$.

We first define a *Symmetric Game* [15]:

Definition 4 (Symmetric Game)**.** *An n-player game is symmetric if the players have the same action space, and their cost functions $c_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i})$ satisfy*

$$c_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i}) = c_j(\boldsymbol{\theta}_j, \boldsymbol{\theta}_{-j}), \forall i, j \in \mathcal{N} \tag{8.12}$$

if $\boldsymbol{\theta}_i = \boldsymbol{\theta}_j$ and $\boldsymbol{\theta}_{-i} = \boldsymbol{\theta}_{-j}$.

In a symmetric game $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i) \rangle$ it is natural to consider a *Symmetric Equilibrium*:

Definition 5 (Symmetric Equilibrium)**.** *An action profile* $\{\boldsymbol{\theta}_i^*\}_{i=1}^n$ *of* $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i)\rangle$ *is a symmetric equilibrium if it is a Nash equilibrium and* $\boldsymbol{\theta}_i^* = \boldsymbol{\theta}_j^*, \forall i, j \in \mathcal{N}$*.*

We now show that our approximate game is symmetric, and always has a symmetric Nash equilibrium.

Theorem 3 (Existence of Nash Equilibrium)**.** $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i)\rangle$ *is a symmetric game and it has at least one symmetric equilibrium.*

Proof. As described above, the players of $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i)\rangle$ use the same action space and complete information of others. Hence, the cost function c_i is symmetric, making $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i)\rangle$ a symmetric game. As $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i)\rangle$ has nonempty, compact and convex action space, and the cost function $\widetilde{c}_i$ is continuous in $\{\boldsymbol{\theta}_i\}_{i=1}^n$ and convex in $\boldsymbol{\theta}_i$, according to Theorem 3 in [15], $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i)\rangle$ has at least one symmetric Nash equilibrium. □

8.3.3 Uniqueness of Nash Equilibrium

While we showed that the approximate game always admits a symmetric Nash equilibrium, it leaves open the possibility that there may be multiple symmetric equilibria, as well as equilibria which are not symmetric. We now demonstrate that this game in fact has a *unique* equilibrium (which must therefore be symmetric).

Theorem 4 (Uniqueness of Nash Equilibrium)**.** $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i)\rangle$ *has a unique Nash equilibrium, and this unique NE is symmetric.*

Proof. We have known that $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i)\rangle$ has at least one NE, and each learner has an nonempty, compact and convex action space $\boldsymbol{\Theta}$. Hence, we can apply Theorem 2 and

Theorem 6 of [92]. That is, for some fixed $\{r_i\}_i^n (0 < r_i < 1, \sum_{i=1}^n r_i = 1)$, if the matrix in Eq. (8.13) is positive definite, then $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i) \rangle$ has a unique NE.

$$Jr(\boldsymbol{\theta}) = \begin{bmatrix} r_1 \nabla_{\boldsymbol{\theta}_1, \boldsymbol{\theta}_1} \widetilde{c}_1(\boldsymbol{\theta}) & \dots & r_1 \nabla_{\boldsymbol{\theta}_1, \boldsymbol{\theta}_n} \widetilde{c}_1(\boldsymbol{\theta}) \\ \vdots & & \vdots \\ r_n \nabla_{\boldsymbol{\theta}_n, \boldsymbol{\theta}_1} \widetilde{c}_n(\boldsymbol{\theta}) & \dots & r_n \nabla_{\boldsymbol{\theta}_n, \boldsymbol{\theta}_n} \widetilde{c}_n(\boldsymbol{\theta}) \end{bmatrix} \tag{8.13}$$

We first let $r_1 = r_2 = ... = r_n = \frac{1}{n}$ and decompose $Jr(\boldsymbol{\theta})$ as follows,

$$Jr(\boldsymbol{\theta}) = \frac{2}{n}\mathbf{P} + \frac{2\beta||\mathbf{z} - \mathbf{y}||_2^2}{\lambda^2 n}(\mathbf{Q} + \mathbf{S} + \mathbf{T}), \tag{8.14}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are *block diagonal matrices* such that $\mathbf{P}_{ii} = \mathbf{X}^\top \mathbf{X}$, $\mathbf{P}_{ij} = \mathbf{0}$, $\mathbf{Q}_{ii} = 4\boldsymbol{\theta}_i \boldsymbol{\theta}_i^\top + \boldsymbol{\theta}_i^\top \boldsymbol{\theta}_i \mathbf{I}$ and $\mathbf{Q}_{ij} = \mathbf{0}$, $\forall i, j \in \mathcal{N}, j \neq i$. $\mathbf{S}$ and $\mathbf{T}$ are *block symmetric matrices* such that $\mathbf{S}_{ii} = \boldsymbol{\theta}_i^\top \boldsymbol{\theta}_i \mathbf{I}$, $\mathbf{S}_{ij} = \boldsymbol{\theta}_i^\top \boldsymbol{\theta}_j \mathbf{I}$, $\mathbf{T}_{ii} = \sum_{j \neq i} \boldsymbol{\theta}_j \boldsymbol{\theta}_j^\top$ and $\mathbf{T}_{ij} = \boldsymbol{\theta}_j \boldsymbol{\theta}_i^\top$, $\forall i, j \in \mathcal{N}, j \neq i$.

Next, we prove that $\mathbf{P}$ is *positive definite*, and $\mathbf{Q}$, $\mathbf{S}$ and $\mathbf{T}$ are *positive semi-definite*. Hence, $Jr(\boldsymbol{\theta})$ is positive definite, which indicates that $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i) \rangle$ has a unique NE. As Theorem 3 points out, the game has at least one symmetric NE. Therefore, the NE is unique and must be symmetric. Due to space limitation the details of this proof are included in Appendix C.4. □

8.4 Computing the Equilibrium

Having shown that $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i) \rangle$ has a unique symmetric Nash equilibrium, we now consider computing its solution. We exploit the symmetry of the game which enables to reduce the search space of the game to only symmetric solutions. Particularly, we derive the symmetric Nash equilibrium of $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i) \rangle$ by solving a single convex optimization problem. We obtain the following result.

Theorem 5. *Let*

$$f(\boldsymbol{\theta}) = \ell(\mathbf{X}\boldsymbol{\theta}, \mathbf{y}) + \frac{\beta(n+1)}{2\lambda^2}||\mathbf{z}-\mathbf{y}||_2^2(\boldsymbol{\theta}^\top\boldsymbol{\theta})^2, \tag{8.15}$$

Then, the unique symmetric NE of $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i)\rangle$, $\{\boldsymbol{\theta}_i^*\}_{i=1}^n$, *can be derived by solving the following convex optimization problem*

$$\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} f(\boldsymbol{\theta}) \tag{8.16}$$

and then letting $\boldsymbol{\theta}_i^* = \boldsymbol{\theta}^*, \forall i \in \mathcal{N}$, *where* $\boldsymbol{\theta}^*$ *is the solution of Eq.* (8.16).

Proof. We prove this theorem by characterizing the first-order optimality conditions of each learner's minimization problem in Eq. (8.8) with c_i being replaced with its approximation $\widetilde{c}_i$. Let $\{\boldsymbol{\theta}_i^*\}_{i=1}^n$ be the NE, then it satisfies

$$(\boldsymbol{\eta} - \boldsymbol{\theta}_i^*)^\top \nabla_{\boldsymbol{\theta}_i}\widetilde{c}_i(\boldsymbol{\theta}_i^*, \boldsymbol{\theta}_{-i}^*) \geq 0, \forall \boldsymbol{\eta} \in \boldsymbol{\Theta}, \forall i \in \mathcal{N} \tag{8.17}$$

where $\nabla_{\boldsymbol{\theta}_i}\widetilde{c}_i(\boldsymbol{\theta}_i^*, \boldsymbol{\theta}_{-i}^*)$ is the gradient of $\widetilde{c}_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i})$ with respect to $\boldsymbol{\theta}_i$ and is evaluated at $\{\boldsymbol{\theta}_i^*\}_{i=1}^n$. Then, Eq. (8.17) is equivalent to the equations as follows:

$$\begin{cases} (\boldsymbol{\eta} - \boldsymbol{\theta}_1^*)^\top \nabla_{\boldsymbol{\theta}_1}\widetilde{c}_1(\boldsymbol{\theta}_1^*, \boldsymbol{\theta}_{-1}^*) \geq 0, \forall \boldsymbol{\eta} \in \boldsymbol{\Theta}, \\ \boldsymbol{\theta}_1^* = \boldsymbol{\theta}_j^*, \forall j \in \mathcal{N} \setminus \{1\} \end{cases} \tag{8.18}$$

The reasons are: first, any solution of Eq. (8.17) satisfies Eq. (8.18), as $\{\boldsymbol{\theta}_i^*\}_{i=1}^n$ is symmetric; Second, any solution of Eq. (8.18) also satisfies Eq. (8.17). By definition of symmetric game, if $\boldsymbol{\theta}_1^* = \boldsymbol{\theta}_j^*$, then $\nabla_{\boldsymbol{\theta}_1}\widetilde{c}_1(\boldsymbol{\theta}_1^*, \boldsymbol{\theta}_{-1}^*) = \nabla_{\boldsymbol{\theta}_j}\widetilde{c}_j(\boldsymbol{\theta}_j^*, \boldsymbol{\theta}_{-j}^*)$, and we have

$$(\boldsymbol{\eta} - \boldsymbol{\theta}_j^*)^\top \nabla_{\boldsymbol{\theta}_j}\widetilde{c}_j(\boldsymbol{\theta}_j^*, \boldsymbol{\theta}_{-j}^*), \forall \boldsymbol{\eta} \in \boldsymbol{\Theta}, \forall j \in \mathcal{N} \setminus \{1\}$$

Hence, Eq. (8.17) and Eq. (8.18) are equivalent. Eq. (8.18) can be further rewritten as

$$(\boldsymbol{\eta} - \boldsymbol{\theta}_1^*)^\top \nabla_{\boldsymbol{\theta}_1} \widetilde{c}_1(\boldsymbol{\theta}_1^*, \boldsymbol{\theta}_{-1}^*)|_{\boldsymbol{\theta}_1^*=\ldots=\boldsymbol{\theta}_n^*} \geq 0, \forall \boldsymbol{\eta} \in \boldsymbol{\Theta}. \tag{8.19}$$

We then let

$$\begin{aligned} F(\boldsymbol{\theta}_1^*) &= \nabla_{\boldsymbol{\theta}_1} \widetilde{c}_1(\boldsymbol{\theta}_1^*, \boldsymbol{\theta}_{-1}^*)|_{\boldsymbol{\theta}_1^*=\ldots=\boldsymbol{\theta}_n^*} \\ &= 2\mathbf{X}^\top(\mathbf{X}\boldsymbol{\theta}_1^* - \mathbf{y}) + \frac{2\beta(n+1)}{\lambda^2}||\mathbf{z} - \mathbf{y}||_2^2 \boldsymbol{\theta}_1^{*\top} \boldsymbol{\theta}_1^* \boldsymbol{\theta}_1^*. \end{aligned} \tag{8.20}$$

Then, $F(\boldsymbol{\theta}_1^*) = \nabla_{\boldsymbol{\theta}_1} f(\boldsymbol{\theta}_1^*)$ where $f(\cdot)$ is defined in Eq. (8.15). Hence, we have

$$(\boldsymbol{\eta} - \boldsymbol{\theta}_1^*)^\top \nabla_{\boldsymbol{\theta}_1} f(\boldsymbol{\theta}_1^*) \geq 0, \forall \boldsymbol{\eta} \in \boldsymbol{\Theta}, \tag{8.21}$$

This means that $\boldsymbol{\theta}_1^*$ is the solution of the optimization problem in Eq. (8.16) which finally completes the proof. □

A deeper look at Eq. (8.15) reveals that the Nash equilibrium can be obtained by each learner independently, without knowing others' actions. This means that the Nash equilibrium can be computed in a distributed manner while the convergence is still guaranteed. Hence, our proposed approach is highly scalable, as increasing the number of learners does not impact the complexity of finding the Nash equilibrium. We investigate the robustness of this equilibrium both using theoretical analysis and experiments in the remaining sections.

8.5 Robustness Analysis

We now draw a connection between the multi-learner equilibrium in the adversarial setting, derived above, and robustness, in the spirit of the analysis by [125]. Specifically, we prove

the equivalence between Eq. (8.16) and a robust linear regression problem where data is maliciously corrupted by some disturbance $\triangle$. Formally, a robust linear regression solves the following problem:

$$\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \max_{\triangle\in\mathcal{U}} ||\mathbf{y} - (\mathbf{X} + \triangle)\boldsymbol{\theta}||_2^2, \tag{8.22}$$

where the uncertainty set $\mathcal{U} = \{\triangle \in \mathbb{R}^{m\times d} \,|\, \triangle^T\triangle = \mathbf{G} : |\mathbf{G}_{ij}| \leq c|\theta_i\theta_j| \;\forall i,j\}$, with $c = \frac{\beta(n+1)}{2\lambda^2}||\mathbf{z} - \mathbf{y}||_2^2$. Note that $\boldsymbol{\theta}$ is a vector and θ_i is the i-th element of $\boldsymbol{\theta}$.

From a game-theoretic point of view, in training phase the defender is simulating an attacker. The attacker maximizes the training error by adding disturbance to $\mathbf{X}$. The magnitude of the disturbance is controlled by a parameter $c = \frac{\beta(n+1)}{2\lambda^2}||\mathbf{z} - \mathbf{y}||_2^2$. Consequently, the robustness of Eq. (8.22) is guaranteed if and only if the magnitude reflects the uncertainty interval. This sheds some light on how to choose λ, β and $\mathbf{z}$ in practice. One strategy is to over-estimate the attacker's strength, which amounts to choosing small values of λ, large values of β and exaggerated target $\mathbf{z}$. The intuition of this strategy is to enlarge the uncertainty set so as to cover potential adversarial behavior. In Experiments section we will show this strategy works well in practice. Another insight from Eq. (8.22) is that the fundamental reason *MLSG* is robust is because it proactively takes adversarial behavior into account.

Theorem 6. *The optimal solution $\boldsymbol{\theta}^*$ of the problem in Eq.* (8.16) *is an optimal solution to the robust optimization problem in Eq.* (8.22).

Proof. Fix $\boldsymbol{\theta}^*$, we show that

$$\max_{\triangle\in\mathcal{U}} ||\mathbf{y} - (\mathbf{X} + \triangle)\boldsymbol{\theta}^*||_2^2 = ||\mathbf{y} - \mathbf{X}\boldsymbol{\theta}^*||_2^2 + c(\boldsymbol{\theta}^{*T}\boldsymbol{\theta}^*)^2.$$

The left-hand side can be expanded as:

$$
\begin{aligned}
&\max_{\triangle\in\mathcal{U}} ||\mathbf{y}-(\mathbf{X}+\triangle)\boldsymbol{\theta}^*||_2^2 \\
=&\max_{\triangle\in\mathcal{U}} ||\mathbf{y}-\mathbf{X}\boldsymbol{\theta}^*-\triangle\boldsymbol{\theta}^*||_2^2 \\
\leq&\max_{\triangle\in\mathcal{U}} ||\mathbf{y}-\mathbf{X}\boldsymbol{\theta}^*||_2^2+\max_{\triangle\in\mathcal{U}} ||\triangle\boldsymbol{\theta}^*||_2^2 \\
=&\max_{\triangle\in\mathcal{U}} ||\mathbf{y}-\mathbf{X}\boldsymbol{\theta}^*||_2^2+\max_{\triangle\in\mathcal{U}} \boldsymbol{\theta}^{*T}\triangle^T\triangle\boldsymbol{\theta}^* \\
&(\text{substitute } \triangle^T\triangle=\mathbf{G}) \\
=&||\mathbf{y}-\mathbf{X}\boldsymbol{\theta}^*||_2^2+\max_{\mathbf{G}} \boldsymbol{\theta}^{*T}\mathbf{G}\boldsymbol{\theta}^* \\
=&||\mathbf{y}-\mathbf{X}\boldsymbol{\theta}^*||_2^2+\max_{\mathbf{G}} \sum_{i=1}^{d}|\theta_i^*|^2\mathbf{G}_{ii}+2\sum_{j=1}^{d}\sum_{i=1}^{j-1}\theta_i^*\theta_j^*\mathbf{G}_{ij} \\
\leq&||\mathbf{y}-\mathbf{X}\boldsymbol{\theta}^*||_2^2+c\sum_{i=1}^{d}|\theta_i^*|^4+2c\sum_{j=1}^{d}\sum_{i=1}^{j-1}(\theta_i^*\theta_j^*)^2 \\
=&||\mathbf{y}-\mathbf{X}\boldsymbol{\theta}^*||_2^2+c\Big(\sum_{i=1}^{d}|\theta_i^*|^2\Big)^2 \\
=&||\mathbf{y}-\mathbf{X}\boldsymbol{\theta}^*||_2^2+c(\boldsymbol{\theta}^{*T}\boldsymbol{\theta}^*)^2.
\end{aligned}
$$

Now we define $\triangle^*=[\sqrt{c}\theta_1^*\mathbf{u},\cdots,\sqrt{c}\theta_n^*\mathbf{u}]$, where θ_i^* is the i-th element of $\boldsymbol{\theta}^*$ and $\mathbf{u}$ is defined as:

$$
\mathbf{u}\triangleq\begin{cases}\frac{\mathbf{y}-\mathbf{X}\boldsymbol{\theta}^*}{||\mathbf{y}-\mathbf{X}\boldsymbol{\theta}^*||_2}, & \textit{if } \mathbf{y}\neq\mathbf{X}\boldsymbol{\theta}^* \\ \textit{any vector with unit } L_2 \textit{ norm}, & \text{otherwise}\end{cases} \tag{8.23}
$$

Then we have:

$$
\begin{aligned}
&\max_{\triangle \in \mathcal{U}} \|\mathbf{y} - (\mathbf{X} + \triangle)\boldsymbol{\theta}^*\|_2^2 \\
\geq &||\mathbf{y} - (\mathbf{X} + \triangle^*)\boldsymbol{\theta}^*||_2^2 \\
= &||\mathbf{y} - \mathbf{X}\boldsymbol{\theta}^* - \triangle^*\boldsymbol{\theta}^*||_2^2 \\
= &||\mathbf{y} - \mathbf{X}\boldsymbol{\theta}^* - \sum_{i=1}^{d} \sqrt{c}|\theta_i^*|^2\mathbf{u}||_2^2 \\
&(\mathbf{u} \text{ is in the same direction as } \mathbf{y} - \mathbf{X}\boldsymbol{\theta}^* \ ) \\
= &||\mathbf{y} - \mathbf{X}\boldsymbol{\theta}^*||_2^2 + ||\sum_{i=1}^{d} \sqrt{c}|\theta_i^*|^2\mathbf{u}||_2^2 \\
= &||\mathbf{y} - \mathbf{X}\boldsymbol{\theta}^*||_2^2 + c(\boldsymbol{\theta}^{*T}\boldsymbol{\theta}^*)^2
\end{aligned}
\tag{8.24}
$$

□

8.6 Experimental Results

8.6.1 Experimental Setup

As previously discussed, a dataset is represented by $(\mathbf{X}, \mathbf{y})$, where $\mathbf{X}$ is the feature matrix and $\mathbf{y}$ is the vector of labels. We use $(\mathbf{x}_j, \mathbf{y}_j)$ to denote the j-th instance and its corresponding label. The dataset is equally divided into a training set $(\mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}})$ and a testing set $(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}})$. We conducted experiments on three datasets: Wine Quality (red wine),PDF malware (PDF), and Boston Housing Market (Boston). The number of learners is set to 5. Due to space limitation the experimental results for the Boston dataset are included in Appendix C.6.

The Wine Quality dataset [18] contains 1599 instances and each instance has 11 features. Those features are physicochemical and sensory measurements for wine. The response variables are quality scores ranging from 0 to 10, where 10 represents for best quality and 0 for least quality. The PDF malware dataset consists of 18658 PDF files collected from the internet. We employed an open-sourced tool *mimicus*[16] to extract 135 real-valued features from PDF files [103]. We then applied *peepdf*[17] to score each PDF between 0 and 10, with a higher score indicating greater likelihood of being malicious.

Throughout, we abbreviate our proposed approach as *MLSG*, and compare it to three other algorithms: ordinary least squares (*OLS*) regression, as well as *Lasso*, and Ridge regression (*Ridge*). *Lasso* and *Ridge* are ordinary least square with L_1 and L_2 regularizers. In our evaluation, we simulate the attacker for different values of β (the probability that a given instance is maliciously manipulated). The specific attack targets $\mathbf{z}$ vary depending on the dataset; we discuss these below. For our evaluation, we compute model parameters (for the equilibrium, in the case of *MLSG*) on training data. We then use test data to compute optimal attacks, characterized by Eq. (8.6). Let $\mathbf{X}'_{\text{test}}$ be the test feature matrix after adversarial manipulation, $\hat{\mathbf{y}}^A_{\text{test}}$ the associated predicted labels on manipulated test data, $\hat{\mathbf{y}}_{\text{test}}$ predicted labels on untainted test data, and $\mathbf{y}_{\text{test}}$ the ground truth labels for test data. We use root expected mean square error (RMSE) as an evaluation metric, where the expectation is with respect to the probability β of a particular instance being maliciously manipulated: $\sqrt{\frac{\beta(\hat{\mathbf{y}}^A_{\text{test}}-\mathbf{y}_{\text{test}})^T(\hat{\mathbf{y}}^A_{\text{test}}-\mathbf{y}_{\text{test}})+(1-\beta)(\hat{\mathbf{y}}_{\text{test}}-\mathbf{y}_{\text{test}})^T(\hat{\mathbf{y}}_{\text{test}}-\mathbf{y}_{\text{test}})}{N}}$, where N is the size of the test data.

[16] https://github.com/srndic/mimicus
[17] https://github.com/rohit-dua/peePDF

8.6.2 The Red Wine Dataset

Recall that the response variables in red wine dataset are quality scores ranging from 0 to 10. We simulated an attacker whose target is to increase the overall scores of testing data. In practice this could correspond to the scenario that wine sellers try to manipulate the evaluation of third-party organizations. We formally define the attacker's target as $\mathbf{z} = \mathbf{y} + \Delta$, where $\mathbf{y}$ is the ground-truth response variables and Δ is a real-valued vector representing the difference between the attacker's target and the ground-truth. Since the maximum score is 10, any element of $\mathbf{z}$ that is greater than 10 is clipped to 10. We define Δ to be homogeneous (all elements are the same); generalization to heterogeneous values is direct. The mean and standard deviation of $\mathbf{y}$ are $\mu_r = 5.64$ and $\sigma_r = 0.81$. We let $\Delta = 5\sigma_r \times \mathbf{1}$, where $\mathbf{1}$ is a vector with all elements equal to one. The intuition for this definition is to simulate the generating process of adversarial data. Specifically, by setting the attacker's target to an unrealistic value (i.e. in current case outside the $3\sigma_r$ of μ_r), the generated adversarial data $\mathbf{X}'$ is supposed to be intrinsically different from $\mathbf{X}$. For ease of exposition we use the term *defender* to refer to *MLSG*.

Remember that in Eq. ((8.11)) there are three hyper-parameters in the defender's loss function: λ, β, and $\mathbf{z}$. λ is the regularization coefficient in the attacker's loss function shown in Eq. ((8.4)). It is negatively proportional to the attacker's strength. β is the probability of a test data being malicious. $\mathbf{z}$ is the predication targets of the attacker. In practice these three hyper-parameters are externally set by the attacker. In the first experiment below we assume the defender knows the values of these three hyper-parameters, which corresponds to the *best case*. The result is shown in Figure 8.1. Each bar is averaged over 50 runs, where at each run we randomly sampled training and test data. The regularization parameters of *Lasso* and *Ridge* were selected by cross-validation. Figure 8.1 demonstrates that *MLSG* approximate

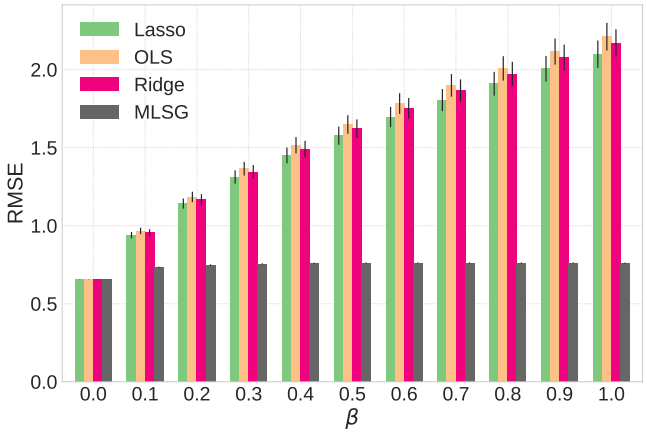

Figure 8.1: RMSE of $\mathbf{y}'$ and $\mathbf{y}$ on the red wine dataset. The defender knows λ, β, and $\mathbf{z}$.

equilibrium solution is significantly more robust than conventional linear regression learning, with and without regularization.

In the second experiment we relaxed the assumption that the defender knows λ, β and $\mathbf{z}$, and instead simulated the practical scenario that the defender obtains estimates for these (for example, from historical attack data), but the estimates have error. We denote by $\hat{\lambda} = 0.5$ and $\hat{\beta} = 0.8$ the defender's estimates of the true λ and β.[18] Remember that β is the probability of an instance being malicious and λ is negatively proportional to the attacker's strength. So the estimation characterizes a *pessimistic* defender that is expecting very strong attacks. We experimented with two kinds of estimation about $\mathbf{z}$: 1) the defender overestimates $\mathbf{z}$: $\hat{\mathbf{z}} = \mathbf{y} + t\mathbf{1}$, where t is a random variable sampled from a uniform distribution over $[5\sigma_r, 10]$; and 2) the defender underestimates $\mathbf{z}$: $\hat{\mathbf{z}} = \mathbf{y} + t\mathbf{1}$, where t is sampled from $[0, 5\sigma_r]$. Here, we only present the results for the latter; the former can be found in Appendix C.5. In Figure 8.2 the y-axis represents the actual values of λ, and the x-axis represents the actual values of β. The color bar on the right of each figure visualizes the average RMSE. Each cell is averaged over 50 runs. The result shows that even if there is a discrepancy between the defender's

[18]We tried alternative values of $\hat{\lambda}$ and $\hat{\beta}$, and the results are consistent. We include these in Appendix C.5.

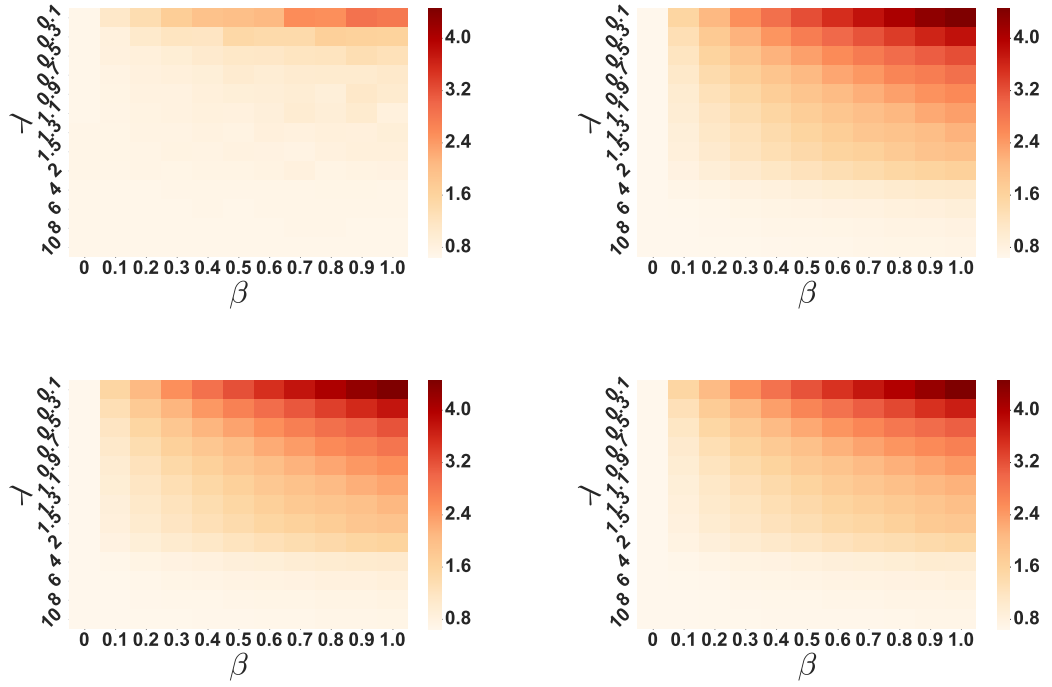

Figure 8.2: The average RMSE across different values of actual λ and β on the red wine dataset. Upper Left: *MLSG*; Upper Right: *Lasso*; Lower Left: *Ridge*; Lower Right: *OLS*.

estimation and the actual adversarial behavior, *MLSG* is consistently more robust than the other approaches.

8.6.3 The PDF Dataset

The response variables of this dataset are malicious scores ranging between 0 and 10. The mean and standard deviation of $\mathbf{y}$ are $\mu_p = 5.56$ and $\sigma_p = 2.66$. Instead of letting the Δ be non-negative as in previous two datasets, the attacker's target is to descrease the scores of malicious PDFs. Consequently, we define $\Delta = -2\sigma_p \times \mathbf{1}_{\mathcal{M}}$, where $\mathcal{M}$ is the set of indices of malicious PDF and $\mathbf{1}_{\mathcal{M}}$ is a vector with only those elements indexed by $\mathcal{M}$ being one and others being zero. Our experiments were conducted on a subset (3000 malicious PDF and 3000 benign PDF) randomly sampled from the original dataset. We evenly divided the subset

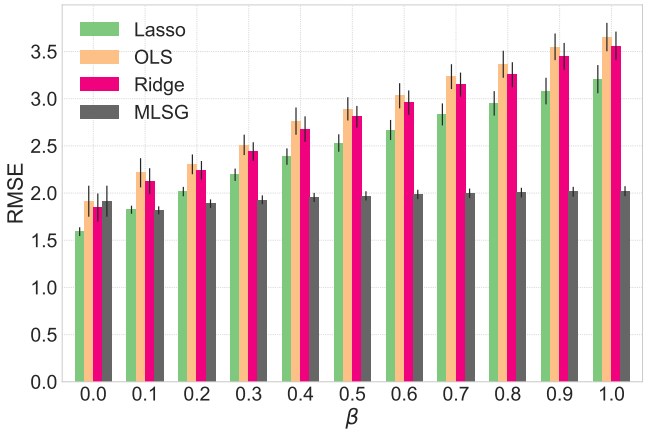

Figure 8.3: RMSE of $\mathbf{y}'$ and $\mathbf{y}$ on PDF dataset. The defender knows λ, β, and $\mathbf{z}$.

into training and testing sets. We applied PCA for dimensionality reduction of the data and selected the top-10 principal components as features. The result for *best case* is displayed in Figure 8.3. Notice that when $\beta = 0$, *MLSG* is less robust than *Lasso*. This is to be expected, as $\beta = 0$ corresponds to non-adversarial data.

Similarly as before we relaxed the assumption that the defender knows λ, β and $\mathbf{z}$ and let the defender's estimation of the true λ and β be $\hat{\lambda} = 1.5$ and $\hat{\beta} = 0.5$. We also experimented with both overestimation and underestimation of $\mathbf{z}$. The defender's estimation is $\hat{\mathbf{z}} = \mathbf{y} - t\mathbf{1}_{\mathcal{M}}$. For overestimation setting t is sampled from $[2\sigma_p, 3\sigma_p]$, and for underestimation setting it is sampled from $[\sigma_p, 2\sigma_p]$. The result for underestimated $\hat{\mathbf{z}}$ is showed in Figure 8.4. Notice that in the upper left plot of Figure 8.4 the area inside the blue rectangle corresponds to those cases where $\hat{\lambda}$ and $\hat{\beta}$ are overestimated and they are more robust than the remaining underestimated cases. Similar patterns can be observed in Figure 8.2. This further supports our claim that it is advantageous to overestimate adversarial behavior.

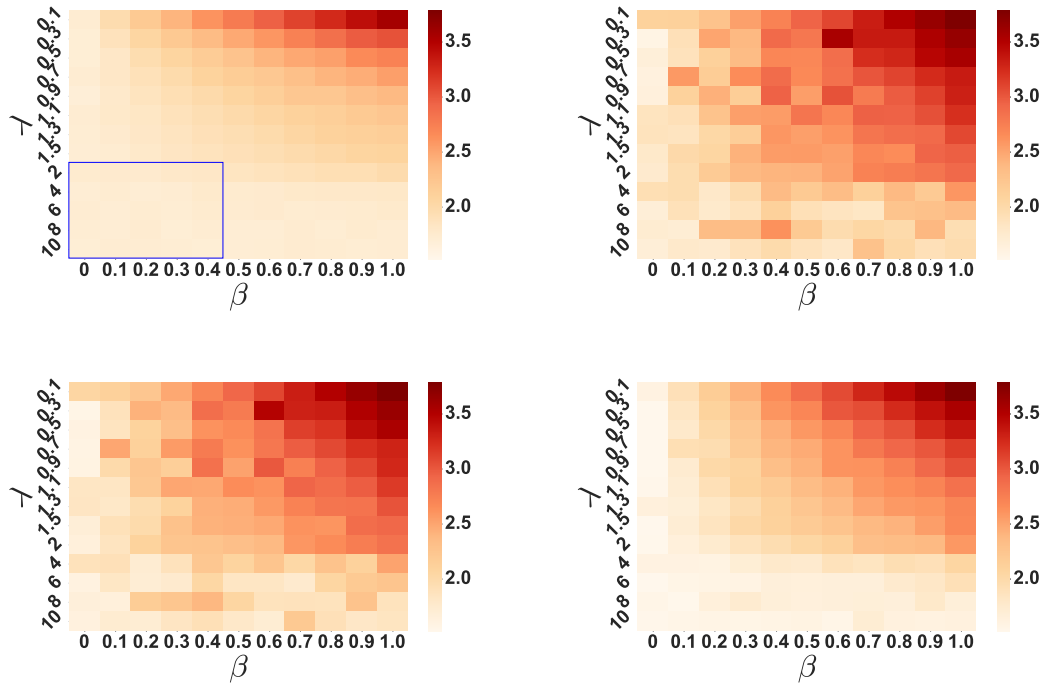

Figure 8.4: The average RMSE across different values of actual λ and β on PDF dataset. Upper Left: *MLSG*; Upper Right: *Lasso*; Lower Left: *Ridge*; Lower Right: *OLS*.

8.7 Conclusion

In this chapter, we study the problem of linear regression in adversarial settings involving multiple learners learning from the same or similar data. In our model, learners first simultaneously decide on their models (*i.e.*, learn), and an attacker then modifies test instances to cause predictions to err towards the attacker's target. We first derive an upper bound on the cost functions of all learners, and the resulting approximate game. We then show that this game has a unique symmetric equilibrium, and present an approach for computing this equilibrium by solving a convex optimization problem. Finally, we show that the equilibrium is robust, both theoretically, and through an extensive experimental evaluation.

Chapter 9

Conclusion and Future Directions

This thesis focuses on the security problem of deploying machine learning systems in adversarial settings. Starting with face recognition systems, we proposed FaceSec, a fine-grained adversarial robustness evaluation framework in Chapter 3. FaceSec incorporates four evaluation dimensions and can support evaluation of the vulnerability of different components of face recognition systems. Using FaceSec, we performed a comprehensive evaluation on five publicly available face recognition systems in various adversarial settings, and obtain some meaningful findings. We believe that FaceSec can serve as a useful framework to advance future research of adversarial learning in a broad array of machine learning systems in addition to face recognition.

Before moving toward our own defense approaches, we revisited robust ML that uses the conventional feature-space attack models. In Chapter 4, we proposed a general methodological framework for evaluating the validity of robust ML against real attacks. We showed that in the context of PDF Malware detection, defense approaches based on these feature-space

models may fail to yield ML models that are robust to realizable attacks. Our results suggest that the practical usefulness of such approaches cannot be taken for granted.

To refine the feature-space attack model, we proposed to use conserved features in Chapter 5 to boost the robustness of feature-space models without compromising their mathematical convenience. We showed that conserved features do exist, and once augmented with these features, Robust ML exhibits generalizable robustness against multiple distinct realizable attacks.

In addition to PDF malware detection, we also investigated adversarial robustness in image classification. Specifically, in Chapter 6, we focused on cognitive modeling of non-salient adversarial examples. We proposed the dual-perturbation attack, and showed that adversarial training with our attack yields significantly greater generalizable robustness to different ℓ_p attacks and aligns better with human perception than conventional robust ML models. We believe these properties make our method a promising candidate for improving adversarial training, the robustness of which is typically limited when facing different attacks.

After investigating robust ML models, we focused on deciding which of a large number of alerts to choose for further investigation—often a necessary step in the detection pipeline. In Chapter 7, we proposed a novel model and principled computational approach for robust alert prioritization. The results showed that our method significantly outperforms non-strategic approaches in nearly all cases, and prior strategic methods where these are feasible, even when the assumptions of our threat model are violated. Our study reveals the benefits of considering robustness of end-to-end systems when designing the full detection pipeline.

The last part of this thesis studied robust learning involving multiple learners and a single attacker. In Chapter 8 we proposed a game-theoretic model for adversarial regression with multiple learners. We showed that the proposed model has a unique pure-strategy

Nash Equilibrium, which is theoretically and empirically robust in adversarial settings. We envision that our work can serve as a useful framework to advance future research of robust decentralized learning.

Looking ahead, there are several future research directions.

Identifying Robust Features. In Chapter 5, we have shown that the key to robust ML against realizable attacks is the collection of conserved features. However, the proposed method for identifying conserved features is quite heuristic and incurs a large amount of execution time on classifiers that employ a vast number of features (*e.g.*, ML-based android malware detectors can have one million features). On the other hand, recent work has shown that adversarial examples can be directly attributed to the presence of *non-robust features* [49]. Several open problems can be explored: How can we automatically identify robust features? Can we construct robust features from non-robust features, *e.g.*, by using a non-linear combination of those?

General Defense against Realizable Attacks in Image Classification. Previous studies have shown that CNNs are often vulnerable to physically realizable attacks [25, 97]. Particularly, in Chapter 3, our systematic and comprehensive adversarial evaluation has shown the vulnerability of each component of face recognition systems. While there is numerous work on adversarial defense against realizable attacks in image classification, none of them provides a general paradigm in various settings. For example, [122], the state-of-the-art defense for face recognition only works on particular neural architectures and attacks (as shown in Chapter 3). Therefore, there are pressing needs for designing a general approach against such realizable attacks.

Robust Machine Learning Inspired by Cognitive Science. In Chapter 6, we have shown that the performance of adversarial training can be significantly improved if augmented

by cognitive science, and the resulting model aligns better with human perception than conventional methods. This work motivates several new research questions. The first is how we can effectively and reliably identify the foreground of an image. Most fixation models work on all objects, but they can fail to provide an accurate foreground. In contrast, semantic segmentation methods can provide a more accurate partition of foreground and background, but are limited to specific objects. This calls for a principled paradigm to achieve both accuracy and generalizability. Second, how can we quantify suspiciousness? While we provide the first model to do so, there may be more effective ways.

References

[1] Martin Abadi et al. "TensorFlow: A system for large-scale machine learning." In: *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation*. 2016, pp. 265–283.

[2] Khalid Alsubhi, Issam Aib, and Raouf Boutaba. "FuzMet: A fuzzy-logic based alert prioritization engine for intrusion detection systems." In: *International Journal of Network Management* 22.4 (2012), pp. 263–284.

[3] Bo An et al. "A deployed quantal response-based patrol planning system for the US Coast Guard." In: *Interfaces* 43.5 (2013), pp. 400–420.

[4] Anish Athalye, Nicholas Carlini, and David Wagner. "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples." In: *International Conference on Machine Learning*. 2018, pp. 274–283.

[5] Marc G Bellemare, Will Dabney, and Rémi Munos. "A distributional perspective on reinforcement learning." In: *Proceedings of the 34th International Conference on Machine Learning (ICML) – Volume 70*. JMLR. 2017, pp. 449–458.

[6] Anand Bhattad, Min Jin Chong, Kaizhao Liang, Bo Li, and David Forsyth. "Unrestricted Adversarial Examples via Semantic Manipulation." In: *International Conference on Learning Representations*. 2020.

[7] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. "Evasion Attacks against Machine Learning at Test Time." In: *European Conference on Machine Learning and Knowledge Discovery in Databases*. 2013, pp. 387–402.

[8] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2011.

[9] M. Brückner and T. Scheffer. "Static Prediction Games for Adversarial Learning Problems." In: *Journal of Machine Learning Research* 13 (2012), pp. 2617–2654.

[10] Michael Brückner and Tobias Scheffer. "Stackelberg Games for Adversarial Prediction Problems." In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2011, pp. 547–555.

[11] Anna L Buczak and Erhan Guven. "A survey of data mining and machine learning methods for cyber security intrusion detection." In: *IEEE Communications Surveys & Tutorials* 18.2 (2016), pp. 1153–1176.

[12] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. "Vggface2: A dataset for recognising faces across pose and age." In: *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. IEEE. 2018, pp. 67–74.

[13] Nicholas Carlini and David A. Wagner. "Towards Evaluating the Robustness of Neural Networks." In: *IEEE Symposium on Security and Privacy* (2017), pp. 39–57.

[14] 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: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 2017, pp. 15–26.

[15] Shih-Fen Cheng, Daniel M. Reeves, Yevgeniy Vorobeychik, and Michael P. Wellman. "Notes on Equilibria in Symmetric Games." In: *International Workshop on Game Theoretic and Decision Theoretic Agents*. 2004, pp. 71–78.

[16] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. "Certified Adversarial Robustness via Randomized Smoothing." In: *International Conference on Machine Learning*. 2019.

[17] Tianji Cong and Atul Prakash. *Masked MS-COCO for robust image classification*. `https://github.com/superctj/Masked_MSCOCO`. 2019.

[18] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. "Modeling Wine Preferences by Data Mining from Physicochemical Properties." In: *Decision Support Systems* 47.4 (2009), pp. 547–553.

[19] M. Cova, C. Kruegel, and G. Vigna. "Detection and analysis of drive- by-download attacks and malicious JavaScript code." In: *International Conference on World Wide Web*. 2010, pp. 281–290.

[20] Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. "Adversarial Classification." In: *SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2004, pp. 99–108.

[21] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. "ImageNet: A large-scale hierarchical image database." In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255.

[22] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. "Arcface: Additive angular margin loss for deep face recognition." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4690–4699.

[23] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. "Boosting adversarial attacks with momentum." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9185–9193.

[24] Yinpeng Dong, Hang Su, Baoyuan Wu, Zhifeng Li, Wei Liu, Tong Zhang, and Jun Zhu. "Efficient decision-based black-box adversarial attacks on face recognition." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 7714–7722.

[25] Kevin Eykholt et al. "Robust Physical-World Attacks on Deep Learning Visual Classification." In: *Computer Vision and Pattern Recognition*. 2018.

[26] *FaceNet Using Pytorch*. `https://github.com/timesler/facenet-pytorch`.

[27] Prahlad Fogla and Wenke Lee. "Evading Network Anomaly Detection Systems: Formal Reasoning and Practical Techniques." In: *ACM Conference on Computer and Communications Security*. 2006, pp. 59–68.

[28] Prahlad Fogla, Monirul Sharif, Roberto Perdisci, Oleg Kolesnikov, and Wenke Lee. "Polymorphic Blending Attacks." In: *USENIX Security Symposium*. 2006.

[29] Meire Fortunato et al. "Noisy networks for exploration." In: *arXiv preprint arXiv:1706.10295* (2017).

[30] Justin Gilmer, Ryan P. Adams, Ian J. Goodfellow, David Andersen, and George E. Dahl. "Motivating the Rules of the Game for Adversarial Example Research." In: arXiv preprint. 2018.

[31] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In: *Proceedings of the 13th international conference on artificial intelligence and statistics (AISTAT)*. 2010, pp. 249–256.

[32] I. Goodfellow, J. Pouget, M. Mirza, B. Xu, D. Warde, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Nets." In: *Neural Information Processing Systems*. 2014, pp. 2672–2680.

[33] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples." In: *International Conference on Learning Representations*. 2015.

[34] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. "Adversarial Perturbations Against Deep Neural Networks for Malware Classification." In: *European Symposium on Research in Computer Security*. 2017.

[35] Claudio Guarnieri, Alessandro Tanasi, Jurriaan Bremer, and Mark Schloesser. *Cuckoo Sandbox: A Malware Analysis System*. http://www.cuckoosandbox.org/. 2012.

[36] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. "Ms-celeb-1m: A dataset and benchmark for large-scale face recognition." In: *European conference on Computer Vision*. Springer. 2016, pp. 87–102.

[37] K. He, X. Zhang, S. Ren, and J. Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification." In: *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1026–1034.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[39] Sen He and Nicolas Pugeault. "Salient Region Segmentation." In: *arXiv preprint arXiv:1803.05759* (2018).

[40] Matteo Hessel et al. "Rainbow: Combining improvements in deep reinforcement learning." In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. AAAI. 2018.

[41] Nicholas J. Higham. "Analysis of the Cholesky decomposition of a semi-definite matrix." In: *Reliable Numerical Computation*. University Press, 1990, pp. 161–185.

[42] Grant Ho, Aashish Sharma, Mobin Javed, Vern Paxson, and David Wagner. "Detecting credential spearphishing in enterprise settings." In: *Proceedings of the 26th USENIX Security Symposium (USENIX Security)*. 2017, pp. 469–485.

[43] Holger H. Hoos and Thomas Stutzle. *Stochastic Local Search : Foundations & Applications*. Morgan Kaufmann, 2004.

[44] Junling Hu, Michael P Wellman, et al. "Multiagent reinforcement learning: theoretical framework and an algorithm." In: *Proceedings of the 15th International Conference on Machine Learning (ICML)*. Vol. 98. 1998, pp. 242–250.

[45] Junling Hu and Michael P Wellman. "Nash Q-learning for general-sum stochastic games." In: *Journal of Machine Learning Research* 4.Nov (2003), pp. 1039–1069.

[46] W. Hu and Y. Tan. "Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN." In: arXiv preprint. 2017.

[47] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Tech. rep. 07-49. University of Massachusetts, Amherst, 2007.

[48] Neminath Hubballi and Vinoth Suryanarayanan. "False alarm minimization techniques in signature-based intrusion detection systems: A survey." In: *Computer Communications* 49 (2014), pp. 1–17.

[49] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. "Adversarial Examples are not Bugs, they are Features." In: *NeurIPS'19*.

[50] Anil K Jain and Stan Z Li. *Handbook of face recognition*. Vol. 1. Springer, 2011.

[51] Alex Kantchelian, J. D. Tygar, and Anthony D. Joseph. "Evasion and Hardening of Tree Ensemble Classifiers." In: *International Conference on Machine Learning*. 2016, pp. 2387–2396.

[52] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." In: *arXiv preprint arXiv:1412.6980* (2014).

[53] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *CoRR* abs/1412.6980 (2015).

[54] Dmytro Korzhyk, Zhengyu Yin, Christopher Kiekintveld, Vincent Conitzer, and Milind Tambe. "Stackelberg vs. Nash in Security Games: An Extended Investigation of Interchangeability, Equivalence, and Uniqueness." In: *Journal of Artificial Intelligence Research* 41 (2011), pp. 297–327.

[55] M. Kümmerer, T. S. A. Wallis, L. A. Gatys, and M. Bethge. "Understanding Low- and High-Level Contributions to Fixation Prediction." In: *IEEE International Conference on Computer Vision*. 2017.

[56] Aron Laszka, Yevgeniy Vorobeychik, Daniel Fabbri, Chao Yan, and Bradley Malin. "A Game-Theoretic Approach for Alert Prioritization." In: *AAAI Workshop on Artificial Intelligence for Cyber Security (AICS)*. 2017.

[57] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana. "Certified Robustness to Adversarial Examples with Differential Privacy." In: *IEEE Symposium on Security and Privacy*. 2019.

[58] Bo Li and Yevgeniy Vorobeychik. "Evasion-robust classification on binary domains." In: *ACM Transactions on Knowledge Discovery from Data* (2018).

[59] Bo Li and Yevgeniy Vorobeychik. "Feature Cross-substitution in Adversarial Classification." In: *Advances in Neural Information Processing Systems*. 2014, pp. 2087–2095.

[60] Zhengfa Liang, Yiliu Feng, Yulan Guo, Hengzhu Liu, Wei Chen, Linbo Qiao, Li Zhou, and Jianfeng Zhang. "Learning for disparity estimation through feature constancy." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2811–2820.

[61] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. "Continuous control with deep reinforcement learning." In: *arXiv preprint arXiv:1509.02971* (2015).

[62] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. "Microsoft coco: Common objects in context." In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[63] Michael L Littman. "Friend-or-foe Q-learning in general-sum games." In: *Proceedings of the 18th International Conference on Machine Learning (ICML)*. Vol. 1. 2001, pp. 322–328.

[64] Michael L Littman. "Markov games as a framework for multi-agent reinforcement learning." In: *Proceedings of the 11th International Conference on International Conference on Machine Learning (ICML)*. Elsevier, 1994, pp. 157–163.

[65] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. "Sphereface: Deep hypersphere embedding for face recognition." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 212–220.

[66] Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. "Large-Margin Softmax Loss for Convolutional Neural Networks." In: *Proceedings of The 33rd International Conference on Machine Learning*. 2016, pp. 507–516.

[67] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. "Delving into transferable adversarial examples and black-box attacks." In: *arXiv preprint arXiv:1611.02770* (2016).

[68] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[69] Daniel Lowd and Christopher Meek. "Adversarial Learning." In: *ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. 2005, pp. 641–647.

[70] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. "Multi-agent actor-critic for mixed cooperative-competitive environments." In: *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*. 2017, pp. 6382–6393.

[71] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards deep learning models resistant to adversarial attacks." In: *International Conference on Learning Representations*. 2018.

[72] Davide Maiorca, Igino Corona, and Giorgio Giacinto. "Looking at the bag is not enough to find the bomb: an evasion of structural methods for malicious PDF files detection." In: *ACM Asia Conference on Computer and Communications Security*. 2013, pp. 119–130.

[73] Mohammad Hossein Manshaei, Quanyan Zhu, Tansu Alpcan, Tamer Bacşar, and Jean-Pierre Hubaux. "Game theory meets network security and privacy." In: *ACM Computing Surveys (CSUR)* 45.3 (2013), p. 25.

[74] Patrick Maupin. *PDFRW: A Pure Python library That Reads and Writes PDFs*. https://github.com/pmaupin/pdfrw. Accessed: 2017-05-18. 2017.

[75] H. B. McMahan, G. J. Gordon, and A. Blum. "Planning in the Presence of Cost Functions Controlled by an Adversary." In: *Proceedings of the 20th International Conference on Machine Learning (ICML)*. 2003, pp. 536–543.

[76] Aleksandar Milenkoski, Marco Vieira, Samuel Kounev, Alberto Avritzer, and Bryan D Payne. "Evaluating computer intrusion detection systems: A survey of common practices." In: *ACM Computing Surveys (CSUR)* 48.1 (2015), p. 12.

[77] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. "Asynchronous methods for deep reinforcement learning." In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML) – Volume 48*. 2016, pp. 1928–1937.

[78] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing Atari with deep reinforcement learning." In: *arXiv preprint arXiv:1312.5602* (2013).

[79] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning." In: *Nature* 518.7540 (2015), p. 529.

[80] Jeet Mohapatra, Tsui Wei Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel. "Towards verifying robustness of neural networks against semantic perturbations." In: *International Conference on Learning Representations*. 2020.

[81] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. "Universal adversarial perturbations." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1765–1773.

[82] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. "Deepfool: a simple and accurate method to fool deep neural networks." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2574–2582.

[83] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. "Practical black-box attacks on deep neural networks using efficient query mechanisms." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 154–169.

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

[85] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. "The limitations of deep learning in adversarial settings." In: *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2016, pp. 372–387.

[86] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. "Towards the Science of Security and Privacy in Machine Learning." In: *IEEE European Symposium on Security and Privacy*. 2018.

[87] Nicolas Papernot, Patrick McDaniel, Xi Wu, and Somesh Jha. "Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks." In: *IEEE Symposium on Security and Privacy,* 2016.

[88] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. "Deep Face Recognition." In: *Proceedings of the British Machine Vision Conference (BMVC)*. 2015, pp. 41.1–41.12.

[89] José Luis Pech-Pacheco, Gabriel Cristóbal, Jesús Chamorro-Martinez, and Joaquín Fernández-Valdivia. "Diatom autofocusing in brightfield microscopy: a comparative study." In: *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*. Vol. 3. IEEE. 2000, pp. 314–317.

[90] *PyTorch implementation of Additive Angular Margin Loss for Deep Face Recognition*. `https://github.com/foamliu/InsightFace-v2`.

[91] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. "Certified defenses against adversarial examples." In: *International Conference on Learning Representations*. 2018.

[92] J Ben Rosen. "Existence and Uniqueness of Equilibrium Points for Concave N-person Games." In: *Econometrica* (1965), pp. 520–534.

[93] Saeed Salah, Gabriel Maciá-Fernández, and Jesús E Díaz-Verdejo. "A model-based survey of alert correlation techniques." In: *Computer Networks* 57.5 (2013), pp. 1289–1317.

[94] Aaron Schlenker et al. "Don't Bury your Head in Warnings: A Game-Theoretic Approach for Intelligent Allocation of Cyber-security Alerts." In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*. 2017, pp. 381–387. DOI: 10.24963/ijcai.2017/54.

[95] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 815–823.

[96] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization." In: *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP) – Volume 1*. INSTICC. SciTePress, 2018, pp. 108–116. DOI: 10.5220/0006639801080116.

[97] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition." In: *ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2016, pp. 1528–1540.

[98] Yash Sharma and Pin-Yu Chen. "Attacking the Madry Defense Model with L_1-based Adversarial Examples." In: *ICLR-18 Workshops*. 2018.

[99] C. Smutz and A. Stavrou. *Malicious PDF Detection Using Matadata and Structural Features*. Tech. rep. 2012.

[100] C. Smutz and A. Stavrou. "Malicious PDF detection using matadata structural features." In: *Annual Computer Security Applications Conference*. 2012, pp. 239–248.

[101] Robin Sommer and Vern Paxson. "Outside the closed world: On using machine learning for network intrusion detection." In: *2010 IEEE symposium on security and privacy*. IEEE. 2010, pp. 305–316.

[102] N. Šrndic and P. Laskov. "Detection of Malicious PDF Files Based on Hierarchical Document Structure." In: *Network and Distributed System Security Symposium*. 2013.

[103] N. Šrndic and P. Laskov. "Practical Evasion of a Learning-Based Classifier: A Case Study." In: *IEEE Symposium on Security and Privacy*. 2014, pp. 197–211.

[104] Nedim Šrndić and Pavel Laskov. "Hidost: a static machine-learning-based detector of malicious files." In: *EURASIP Journal on Information Security* 2016.1 (2016), p. 22.

[105] Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. "Deep learning face representation by joint identification-verification." In: *Advances in Neural Information Processing Systems*. 2014, pp. 1988–1996.

[106] Yi Sun, Xiaogang Wang, and Xiaoou Tang. "Deep learning face representation from predicting 10,000 classes." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1891–1898.

[107] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. "Inception-v4, inception-resnet and the impact of residual connections on learning." In: *arXiv preprint arXiv:1602.07261* (2016).

[108] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. "Intriguing properties of neural networks." In: *International Conference on Learning Representations*. 2014.

[109] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. "Deepface: Closing the gap to human-level performance in face verification." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1701–1708.

[110] Gerald Tesauro. "TD-Gammon, a self-teaching backgammon program, achieves master-level play." In: *Neural Computation* 6.2 (1994), pp. 215–219.

[111] Anne M Treisman and Garry Gelade. "A feature-integration theory of attention." In: *Cognitive psychology* 12.1 (1980), pp. 97–136.

[112] Jason Tsai, Thanh H. Nguyen, and Milind Tambe. "Security Games for Controlling Contagion." In: *Proceedings of the 26th AAAI Conference on Artificial Intelligence*. AAAI'12. Toronto, Ontario, Canada: AAAI Press, 2012, pp. 1464–1470.

[113] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. "Robustness May Be at Odds with Accuracy." In: *International Conference on Learning Representations*. 2019.

[114] Emmanouil Vasilomanolakis, Shankar Karuppayah, Max Mühlhäuser, and Mathias Fischer. "Taxonomy and survey of collaborative intrusion detection." In: *ACM Computing Surveys (CSUR)* 47.4 (2015), p. 55.

[115] Yevgeniy Vorobeychik and Murat Kantarcioglu. *Adversarial Machine Learning*. Morgan and Claypool, 2018.

[116] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. "Cosface: Large margin cosine loss for deep face recognition." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5265–5274.

[117] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. "Dueling Network Architectures for Deep Reinforcement Learning." In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML)*. 2016, pp. 1995–2003.

[118] Christopher JCH Watkins and Peter Dayan. "Q-learning." In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[119] Christopher John Cornish Hellaby Watkins. "Learning from delayed rewards." PhD thesis. King's College, Cambridge, 1989.

[120] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. "A discriminative feature learning approach for deep face recognition." In: *European Conference on Computer Vision*. Springer. 2016, pp. 499–515.

[121] Eric Wong and J. Zico Kolter. "Provable defenses against adversarial examples via the convex outer adversarial polytope." In: *International Conference on Machine Learning*. 2018.

[122] Tong Wu, Liang Tong, and Yevgeniy Vorobeychik. "Defending Against Physically Realizable Attacks on Image Classification." In: *8th International Conference on Learning Representations (ICLR)*. 2020.

[123] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. "Adversarial examples for semantic segmentation and object detection." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 1369–1378.

[124] H. Xu, C. Caramanis, and S. Mannor. "Robustness and Regularization of Support Vector Machines." In: *Journal of Machine Learning Research* 10 (2009), pp. 1485–1510.

[125] Huan Xu, Constantine Caramanis, and Shie Mannor. "Robust Regression and Lasso." In: *Advances in Neural Information Processing Systems*. 2009, pp. 1801–1808.

[126] Weilin Xu, Yanjun Qi, and David Evans. "Automatically Evading Classifiers: A Case Study on PDF Malware Classifiers." In: *Network and Distributed System Security Symposium*. 2016.

[127] C. Yan, B. Li, Y. Vorobeychik, A. Laszka, D. Fabbri, and B. Malin. "Get Your Workload in Order: Game Theoretic Prioritization of Database Auditing." In: *Proceedings of the 34th IEEE International Conference on Data Engineering (ICDE)*. 2018, pp. 1304–1307. DOI: 10.1109/ICDE.2018.00136.

[128] Xiao Yang, Dingcheng Yang, Yinpeng Dong, Wenjian Yu, Hang Su, and Jun Zhu. "Delving into the Adversarial Robustness on Face Recognition." In: *arXiv preprint arXiv:2007.04118* (2020).

[129] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. "Learning face representation from scratch." In: *arXiv preprint arXiv:1411.7923* (2014).

[130] F. Zhang, P.P.K. Chan, B Biggio, D.S. Yeung, and F. Roli. "Adversarial feature selection against evasion attacks." In: *IEEE Transactions on Cybernetics* (2015).

[131] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks." In: *IEEE Signal Processing Letters* 23.10 (2016), pp. 1499–1503. DOI: 10.1109/LSP.2016.2603342.

[132] Yan Zhou, Murat Kantarcioglu, Bhavani M. Thuraisingham, and Bowei Xi. "Adversarial Support Vector Machine Learning." In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2012, pp. 1059–1067.

Appendix A

Supplement for Chapter 3

A.1 Robustness of Face Recognition Components

A.1.1 Open-Set Systems Under Dodging Attacks

To study the robustness of open-set system components under dodging attacks, we employ six different face recognition systems and then evaluate the attack success rates of dodging attacks corresponding to different target and surrogate face recognition models. Specifically, besides the five systems (VGGFace, FaceNet, ArcFace18, ArcFace50, and ArcFace101) presented in Table 3.4 of Chapter 3, we build a face recognition model by training FaceNet [95] using the VGGFace2 dataset [12] (henceforth, *FaceNet+*). Here, FaceNet and FaceNet+ are trained using the same neural architecture but different training sets, while the ArcFace variations share the same training data but with different architectures. The results are presented in Fig. A.1.

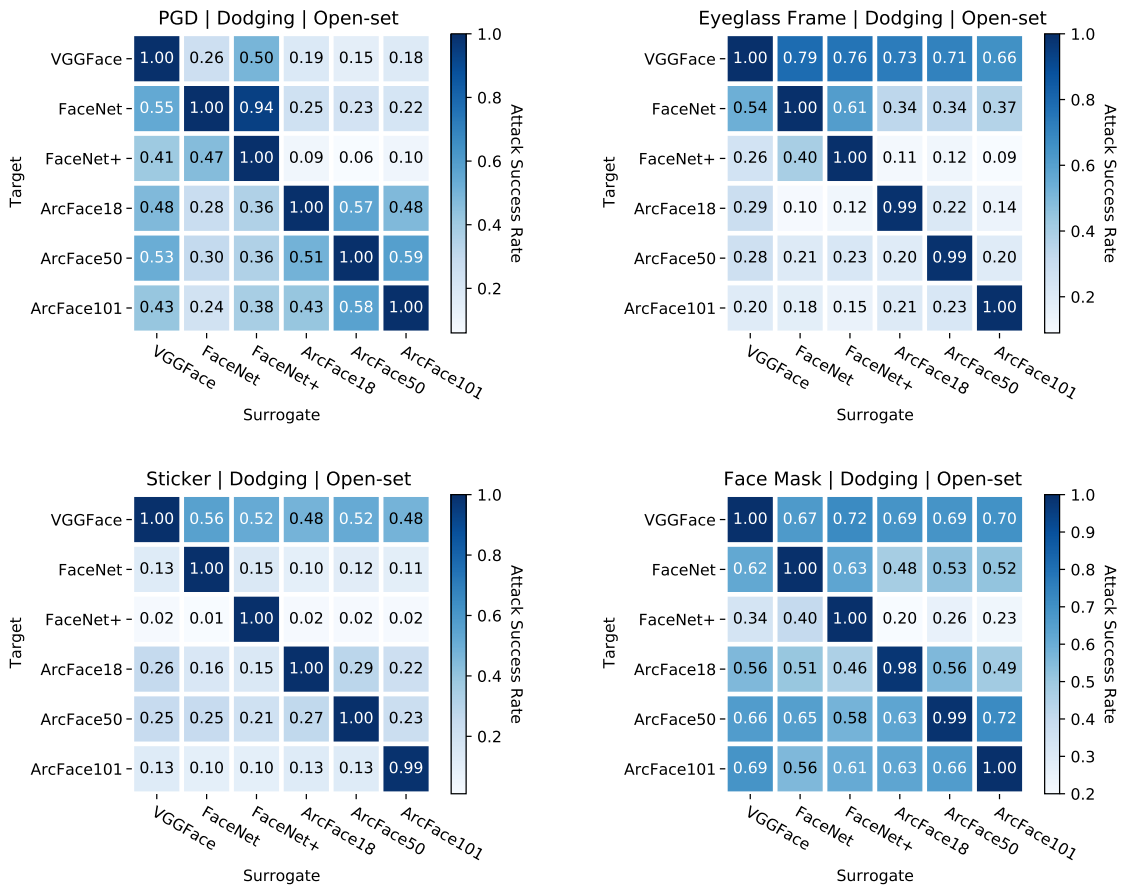

Figure A.1: Attack success rate of dodging attacks with different open-set target and surrogate models. Upper left: PGD attack. Upper right: Eyeglass frame attack. Lower left: Sticker attack. Lower right: Face mask attack.

We have the following two observations, which are similar to those observed from dodging attacks on closed-set systems in the main paper. First, in most cases, an open-set system's neural architecture is more fragile than its training set. For example, under the PGD attack, adversarial examples in response to FaceNet+ have a 94% success rate on FaceNet (which is trained using the same architecture but different training data), while the success rates among the ArcFace systems (which are built with the same training set but different neural architectures) are only around 50%. However, there are also some cases where the neural architecture exhibits similar robustness to the training set. For example, when black-box

Table A.1: Attack success rate of impersonation attacks on closed-set face recognition systems by the attacker's system knowledge. Z represents zero knowledge, T is training set, A is neural architecture, and F represents full knowledge.

Target System	**Attack Type**	**Attacker's System Knowledge**			
		Z	**T**	**A**	**F**
VGGFace	PGD	0.11	0.21	0.35	1.00
	Eyeglass Frame	0.01	0.01	0.03	0.95
	Sticker	0.00	0.00	0.00	1.00
	Face Mask	0.00	0.01	0.02	1.00
FaceNet	PGD	0.23	0.32	1.00	1.00
	Eyeglass Frame	0.00	0.00	0.28	0.99
	Sticker	0.01	0.00	0.21	1.00
	Face Mask	0.00	0.00	0.26	0.99
ArcFace18	PGD	0.18	0.25	0.69	1.00
	Eyeglass Frame	0.01	0.01	0.05	0.89
	Sticker	0.00	0.00	0.01	0.94
	Face Mask	0.01	0.01	0.03	0.77
ArcFace50	PGD	0.13	0.15	0.45	0.87
	Eyeglass Frame	0.02	0.02	0.03	0.67
	Sticker	0.00	0.00	0.00	0.58
	Face Mask	0.01	0.01	0.01	0.60
ArcFace101	PGD	0.14	0.16	0.42	0.96
	Eyeglass Frame	0.00	0.00	0.03	0.58
	Sticker	0.00	0.00	0.00	0.50
	Face Mask	0.01	0.01	0.04	0.73

attacks are too weak (under sticker attack), both neural architecture and training set are robust; when the attacks are too strong (under face mask attack), these two components exhibit similar levels of vulnerability. Second, the grid-level face mask attack is considerably more effective than the PGD attack, and significantly more potent than other physically realizable attacks. Like dodging attacks in closed-set settings, most black-box pixel-level physically realizable attacks have relatively low transferability on open-set face recognition systems, with only about 20% success rate.

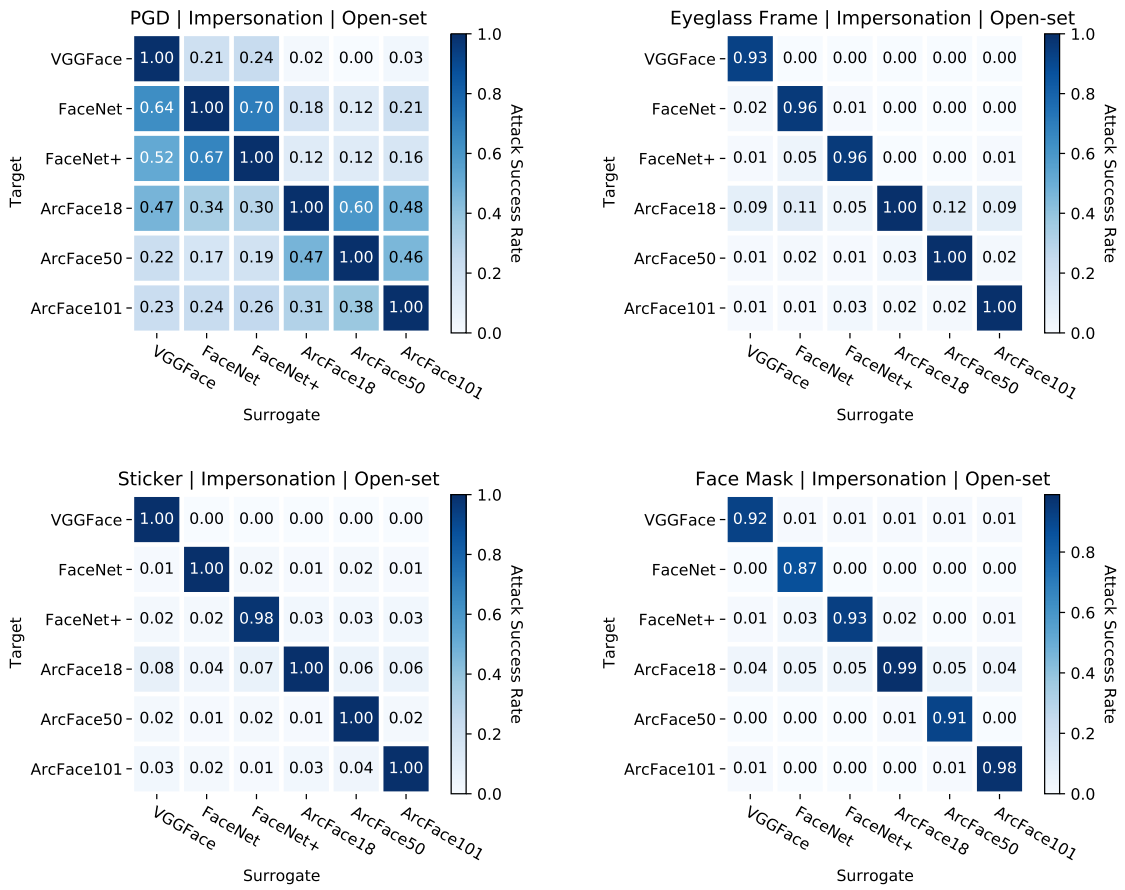

Figure A.2: Attack success rate of impersonation attacks with different open-set target and surrogate models. Upper left: PGD attack. Upper right: Eyeglass frame attack. Lower left: Sticker attack. Lower right:Face mask attack.

A.1.2 Closed-Set Systems Under Impersonation Attacks

Here, we use impersonation attacks to evaluate the robustness of closed-set systems. In our experiments, all the closed-set models are 100-class classifiers, as introduced in Section 3.3. For any input face image $\boldsymbol{x}$ and its identity $y \in [0, 99]$, we let the target identity of the impersonation attack to be $y_t = (y + 1)\%100$. An impersonation attack is successful only when the resulting adversarial example is misclassified as the target identity y_t. The results are shown in Table A.1.

We have two key findings. First, compared to Table 3.5 of Chapter 3, we observe that closed-set systems are significantly more robust to impersonation attacks than dodging attacks. Especially when an attacker has no accurate knowledge about the target system, the attack success rate of physically realizable attacks can be as low as 0%. Second, it can be seen that closed-set systems exhibit moderate robustness against digital impersonation attacks. In such attacks, the knowledge of neural architecture is significantly more important than the training set. For example, by knowing the neural architecture of ArcFace18, a PGD attack can achieve a 69% success rate. In contrast, this rate drops to 25% when only the training set is visible to the attacker.

A.1.3 Open-Set Systems Under Impersonation Attacks

To evaluate impersonation attacks on open-set systems, we randomly select 100 pairs from the LFW dataset [47]. Each pair contains two face images corresponding to different identities. We let one image as the input $\boldsymbol{x}$ and the other as the target gallery image $\boldsymbol{x}_t^*$. An impersonation attack is successful only when the resulting adversarial example and $\boldsymbol{x}_t^*$ are verified as the same identity. The experimental results are presented in Fig. A.2.

Similar to the impersonation attacks on closed-set systems, we have the following two observations that are consistent with our previous summary. First, open-set systems are very robust to black-box impersonation physically realizable attacks. In most cases, these attacks can only achieve a success rate of less than 10%. In contrast, the PGD attack is significantly more potent. And under this attack, the neural architecture is considerably more vulnerable than the training set (*e.g.*, comparing FaceNet variations to ArcFace models).

A.2 Efficacy of Using Momentum and Ensemble Models in Transfer-based Attacks

Next, we evaluate the efficacy of using momentum and ensemble-based surrogate models in transfer-based dodging attacks. For a given closed-set target face recognition system, we first train a surrogate model using the same training data. Specifically, we use both a *single* surrogate trained on a different architecture[19], and an *ensembled* surrogate by ensembling the other four systems in the way described in Section 3.2 of the main paper. We then produce white-box dodging attacks on the surrogate and evaluate the resulting examples' attack success rate on the target model. For each attack, we compare the momentum method (*i.e.*, w/ mmt) and the conventional gradient-based approach (*i.e.*, w/o mmt). The results are shown in Table A.2, A.3, A.4, and A.5.

We have two key observations. First, both ensemble and momentum contribute to stronger transferability, although in most cases, ensemble contributes more. For example, the ensemble method can boost the transferability of PGD attacks on FaceNet by 31%, while the improvement by momentum is only about 10%. Second, the efficacy of momentum and ensemble models is highly dependent on the nature of perturbation. For digital attacks, these methods combined can significantly improve transferability by up to 55%. In grid-level face mask attacks, the improvement is as considerable as up to 16%. However, both methods can only marginally boost the transferability of pixel-level realizable attacks. Especially in the sticker attacks, the improvement is nearly negligible. We leave effective transfer-based pixel-level physically realizable attacks as an open problem for future research.

[19]For a given target model, we trained four single surrogates corresponding to the other four architectures. Below, we only present the result of the surrogate that has the highest attack success rate.

Table A.2: Attack success rate of dodging PGD attacks on closed-set face recognition systems. Here, only the target system's training data is visible to the attacker, and we use different surrogate models.

Target System	**Surrogate System**			
	Single		**Ensemble**	
	w/o mmt	**w/ mmt**	**w/o mmt**	**w/ mmt**
VGGFace	0.08	0.16	0.43	0.51
FaceNet	0.42	0.52	0.73	0.83
ArcFace18	0.42	0.51	0.87	0.92
ArcFace50	0.35	0.55	0.86	0.90
ArcFace101	0.32	0.39	0.71	0.78

Table A.3: Attack success rate of dodging eyeglass frame attacks on closed-set face recognition systems. Here, only the target system's training data is visible to the attacker, and we use different surrogate models.

Target System	**Surrogate System**			
	Single		**Ensemble**	
	w/o mmt	**w/ mmt**	**w/o mmt**	**w/ mmt**
VGGFace	0.17	0.22	0.26	0.28
FaceNet	0.08	0.09	0.14	0.16
ArcFace18	0.02	0.03	0.05	0.06
ArcFace50	0.05	0.05	0.10	0.12
ArcFace101	0.02	0.03	0.02	0.03

Table A.4: Attack success rate of dodging sticker attacks on closed-set face recognition systems. Here, only the target system's training data is visible to the attacker, and we use different surrogate models.

Target System	**Surrogate System**			
	Single		**Ensemble**	
	w/o mmt	**w/ mmt**	**w/o mmt**	**w/ mmt**
VGGFace	0.02	0.02	0.06	0.06
FaceNet	0.00	0.00	0.01	0.01
ArcFace18	0.00	0.00	0.01	0.01
ArcFace50	0.00	0.00	0.00	0.01
ArcFace101	0.00	0.01	0.04	0.04

Table A.5: Attack success rate of dodging face mask attacks on closed-set face recognition systems. Here, only the target system's training data is visible to the attacker, and we use different surrogate models.

Target System	**Surrogate System**			
	Single		**Ensemble**	
	w/o mmt	**w/ mmt**	**w/o mmt**	**w/ mmt**
VGGFace	0.18	0.26	0.20	0.32
FaceNet	0.26	0.38	0.42	0.42
ArcFace18	0.21	0.33	0.21	0.33
ArcFace50	0.28	0.34	0.36	0.36
ArcFace101	0.22	0.34	0.30	0.36

Table A.6: Attack success rate of dodging attacks on open-set face recognition systems by the universality of adversarial examples. Here, N represents the batch size of face images that share a universal perturbation.

Target System	**Attack Type**	**Attacker's Capability**			
		N=1	**N=5**	**N=10**	**N=20**
VGGFace	PGD	1.00	0.89	0.81	0.53
	Eyeglass Frame	1.00	1.00	1.00	1.00
	Sticker	1.00	1.00	1.00	1.00
	Face Mask	1.00	1.00	1.00	1.00
FaceNet	PGD	1.00	0.02	0.02	0.02
	Eyeglass Frame	1.00	1.00	1.00	1.00
	Sticker	1.00	1.00	0.99	0.90
	Face Mask	1.00	1.00	0.99	0.98
ArcFace18	PGD	1.00	0.96	0.79	0.46
	Eyeglass Frame	0.99	0.86	0.70	0.67
	Sticker	1.00	1.00	1.00	0.99
	Face Mask	0.98	0.98	0.93	0.92
ArcFace50	PGD	1.00	0.91	0.75	0.47
	Eyeglass Frame	0.99	0.78	0.67	0.62
	Sticker	1.00	1.00	1.00	0.00
	Face Mask	0.99	0.99	0.99	0.94
ArcFace101	PGD	1.00	0.68	0.68	0.41
	Eyeglass Frame	1.00	0.85	0.73	0.65
	Sticker	0.99	0.98	0.97	0.97
	Face Mask	1.00	1.00	1.00	1.00

A.3 Universal Attacks

Finally, we evaluate open-set systems under universal dodging attacks. The results are shown in Table A.6. Compared to Table 3.7 of Chapter 3, we find that open-set systems are significantly more fragile to universal perturbations of all types than their closed-set counterparts. For example, when $N = 20$, universal sticker attacks on open-set ArcFace101 have a 97% success rate, but no success in closed-set settings. Moreover, we again observe that the universal grid-level face mask attack is more effective than the other perturbation types. Here, we also find that the sticker attack is as potent as the face mask attack in open-set settings.

Appendix B

Supplement for Chapter 6

B.1 Alternative Approach for Modeling Suspiciousness

Our second method of modeling the suspiciousness $\mathcal{S}(\boldsymbol{x})$ of an input image $\boldsymbol{x}$ leverages a recent computational model of image salience, DeepGaze II [55]. DeepGaze II outputs predicted pixel-level density of human fixations on an image with the total density over the entire image summing to 1. Our measure of relative salience of the foreground to background is the *foreground score*, which is defined as $\mathcal{S}(\boldsymbol{x}) = \sum_{i \in \{k | \mathcal{F}(\boldsymbol{x})_k \neq 0\}} s_i$, where s_i is the saliency score produced by DeepGaze II for pixel i of image $\boldsymbol{x}$. Since foreground, as a fraction of the image, tends to be around 50-60%, a score significantly higher than 0.5 indicates that predicted human fixation is relatively localized to the foreground.

The above approach models background suspiciousness in a way that aligns with human perception. One limitation should be noted: the DeepGaze II model is designed on non-adversarial data and can thus be attacked. This may lead to adversarial examples that have a salient background but achieve a high foreground score. However, we empirically find that

the DeepGaze II model is generally robust to perturbed images in practice. In most cases, our attack produces non-salient background when $\lambda \neq 0$ in Eq. (6.1) of Chapter 6 (e.g., see Figure B.1).

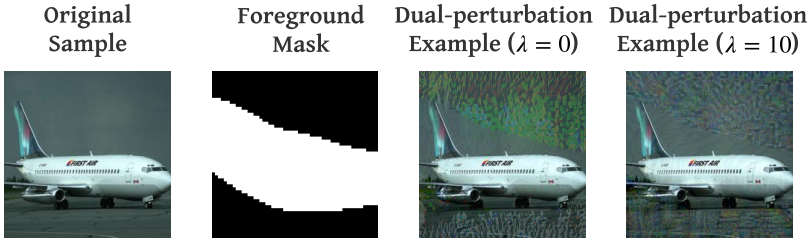

Figure B.1: An illustration of dual-perturbation attacks that leverages fixation prediction to model suspiciousness. Adversarial examples are with large ℓ_∞ perturbations on the background ($\epsilon_B = 20/255$) and small ℓ_∞ perturbations on the foreground ($\epsilon_F = 4/255$). A parameter λ is used to control background salience explicitly. Our attack produces non-salient background when $\lambda \neq 0$.

B.2 Datasets

B.2.1 Segment-6

The statistics of the Segment-6 dataset are displayed in Table B.1.

B.2.2 STL-10

The statistics of the STL-10 dataset are displayed in Table B.2.

B.2.3 ImageNet-10

The labels and number of images per class in the ImageNet-10 dataset are listed in Table B.3.

Table B.1: Number of samples in each class of the Segment-6 dataset.

Class	**Number of samples**	
	Training	**Test**
Train	3,000	200
Bird	3,000	200
Cat	3,000	200
Dog	3,000	200
Toilet	3,000	200
Clock	3,000	200
Total	18,000	1,200

Table B.2: Number of samples in each class of the STL-10 dataset.

Class	**Number of samples**	
	Training	**Test**
Airplane	500	10
Bird	500	10
Car	500	10
Cat	500	10
Deer	500	10
Dog	500	10
Horse	500	10
Monkey	500	10
Ship	500	10
Truck	500	10
Total	5,000	100

Table B.3: Number of samples in each class of the ImageNet-10 dataset.

Class	**Number of samples**	
	Training	**Test**
Airplane	500	10
Car	500	10
Cat	500	10
Dog	500	10
Truck	500	10
Elephant	500	10
Zebra	500	10
Bus	500	10
Bear	500	10
Bicycle	500	10
Total	5,000	100

B.3 Implementations

We implemented all the attack model, as well as the defense approaches in PyTorch[20], an open-source library for neural network learning. We used the ResNet34 model [38] and standard transfer learning, as the datasets employed in our experiments do not have a sufficient amount of data to achieve high accuracy. Specifically, we initialized the network with the model pre-trained on ImageNet, reset the final fully connected layer, and added a *normalization layer* in front of the ResNet34 model, which performs a channel-wise transformation of an input by subtracting $(0.485, 0.456, 0.406)$ (the mean of ImageNet) and then being divided by $(0.229, 0.224, 0.225)$ (the standard deviation of ImageNet);[21] then, we train the neural networks as usual.

Unless otherwise specified, we used 60 epochs with training batch size 128 for Segment-6. For STL-10 and ImageNet-10. we trained the classifiers for 20 epochs by using a batch size of 64. We used Adam Optimizer [52] with initial learning rate of 10^{-4} for *Clean*, and 10^{-3} for *AT-PGD* and *AT-Dual*, respectively. We dropped the learning rate by 0.1 every 20 epochs on Segment-6, and similarly at the 8th and 15th epochs on STL-10 and ImageNet-10.

As mentioned above, we implemented *PGD* and *dual-perturbation* attacks, bounded by both ℓ_∞ and ℓ_2 norms, to evaluate robustness of a classification model, as well as to build robust classifiers. For ℓ_∞ attacks, when they were used for evaluation, they are performed with 20 steps; for training robust classifiers, these attacks were performed with 10 steps at each epoch of adversarial training. Similarly, for ℓ_2 attacks, they were performed with 100 steps for evaluation, and 50 steps for adversarial training. We used the semantic segmentation masks

[20] Available at `https://pytorch.org/`.

[21] To fit the Segment-6 dataset which contains much smaller images compared to ImageNet, we also reset the first convolutional layer of the pre-trained ResNet34 model by reducing the kernel size from 7×7 to 3×3, stride from 2 to 1, and pad from 3 to 1.

on the Segment-6 dataset and used fixation prediction to identify foreground and backround on STL-10 and ImageNet-10.

B.4 Adversarial Training Using ℓ_2 Attacks on STL-10

Here, we present experimental results of the robustness of classifiers that use adversarial training with ℓ_2 norm attacks on STL-10. Specifically, we trained AT-PGD using ℓ_2 PGD attack with $\epsilon = 1.0$, and AT-Dual by using ℓ_2 dual-perturbation attack with $\{\epsilon_F, \epsilon_B, \lambda\} = \{1.0, 5.0, 0.0\}$. The results are shown in Figure B.2, B.3, B.4, and B.5.

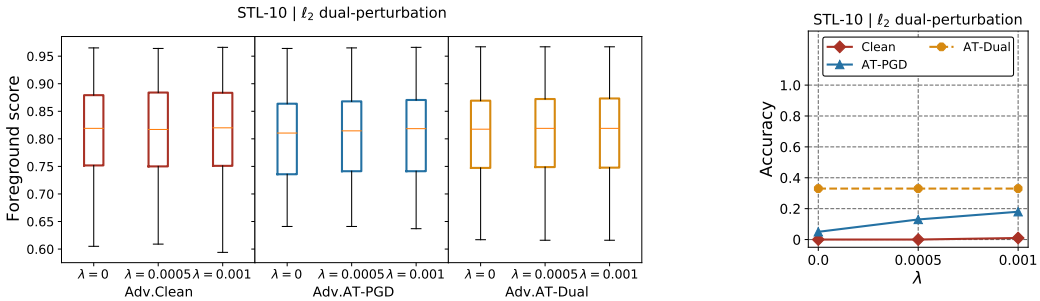

Figure B.2: Saliency analysis. The ℓ_2 dual-perturbation attacks are performed by using $\{\epsilon_F, \epsilon_B\} = \{1.0, 5.0\}$, and a variety of λ displayed in the figure. Left: foreground scores of dual-perturbation examples in response to different classifiers. Right: accuracy of classifiers on dual-perturbation examples with salience control.

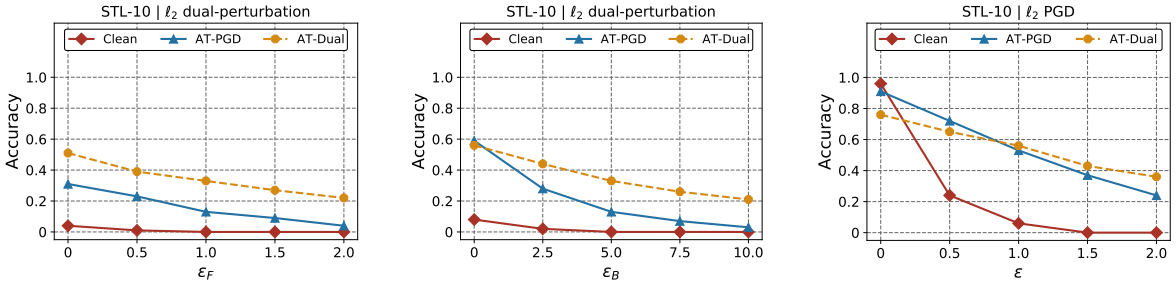

Figure B.3: Robustness to white-box ℓ_2 attacks on STL-10. Left: ℓ_2 dual-perturbation attacks with different foreground distortions. ϵ_B is fixed to be 5.0 and $\lambda = 0.0005$. Middle: ℓ_2 dual-perturbation attacks with different background distortions. ϵ_F is fixed to be 1.0 and $\lambda = 0.0005$. Right: ℓ_2 PGD attacks.

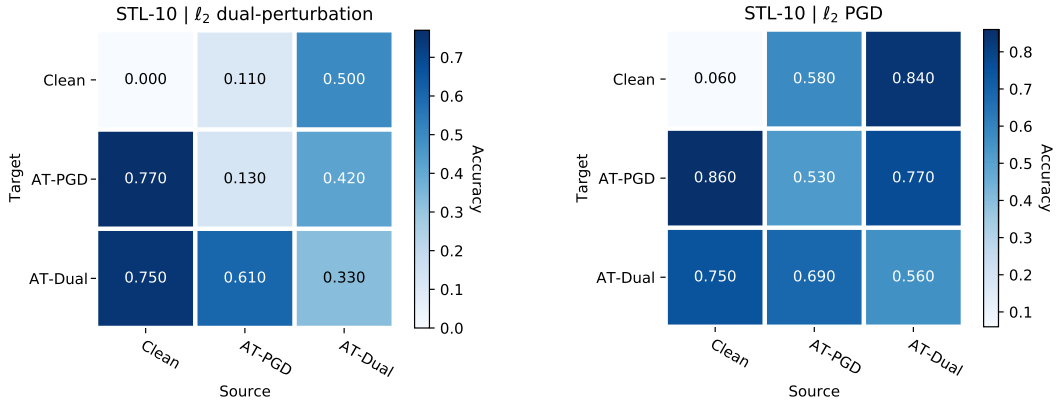

Figure B.4: Robustness against adversarial examples transferred from other models on STL-10. Left: ℓ_2 dual-perturbation attacks performed by using $\{\epsilon_F, \epsilon_B, \lambda\} = \{1.0, 5.0, 0.0005\}$ on different source models. Right: ℓ_2 PGD attacks with $\epsilon = 1.0$ on different source models.

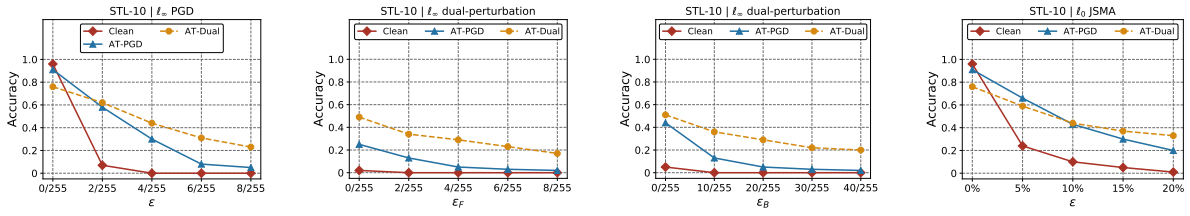

Figure B.5: Robustness to additional white-box attacks on STL-10. Left: 20 steps of ℓ_∞ PGD attacks. Middle left: 20 steps of ℓ_∞ dual-perturbation attacks with different foreground distortions. ϵ_B is fixed to be $20/255$ and $\lambda = 0.0005$. Middle right: 20 steps of ℓ_∞ dual-perturbation attacks with different background distortions. ϵ_F is fixed to be $4/255$ and $\lambda = 0.0005$. Right: ℓ_0 JSMA attacks.

B.5 Adversarial Training Using ℓ_2 Attacks on Segment-6

Now, we present experimental results of the robustness of classifiers that use adversarial training with ℓ_2 norm attacks on Segment-6. Since DeepGaze II only work on images with more than 35×35 pixels, we are unable to use DeepGaze II to compute the *foreground score (FS)* for Segment-6. Hence, in the following experiment on this dataset, we omit the salience term in the optimization problem of Eq. (6.1) and Eq. (6.7) in Chapter 6. Specifically, we trained AT-PGD using ℓ_2 PGD attack with $\epsilon = 0.5$, and AT-Dual by using ℓ_2 dual-perturbation attack with $\{\epsilon_F, \epsilon_B\} = \{0.5, 2.5\}$. The results are shown in Figure B.6, B.7, and B.8.

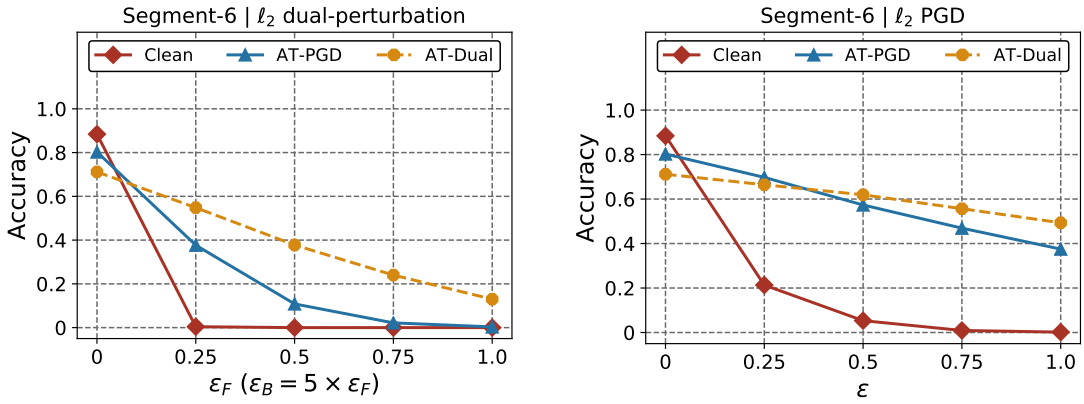

Figure B.6: Robustness to white-box ℓ_2 attacks on Segment-6. Left: ℓ_2 dual-perturbation attacks with different foreground and background distortions. Right: ℓ_2 PGD attacks.

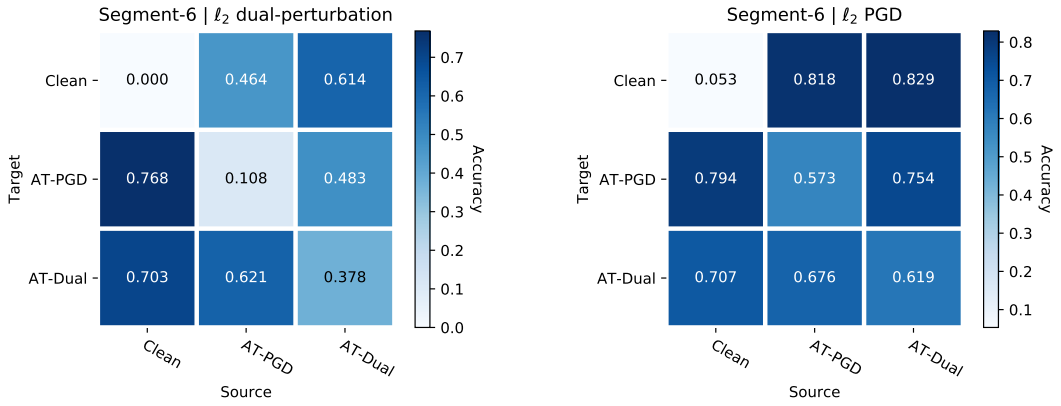

Figure B.7: Robustness against adversarial examples transferred from other models on Segment-6. Left: ℓ_2 dual-perturbation attacks performed by using $\{\epsilon_F, \epsilon_B\} = \{0.5, 2.5\}$ on different source models. Right: ℓ_2 PGD attacks with $\epsilon = 0.5$ on different source models.

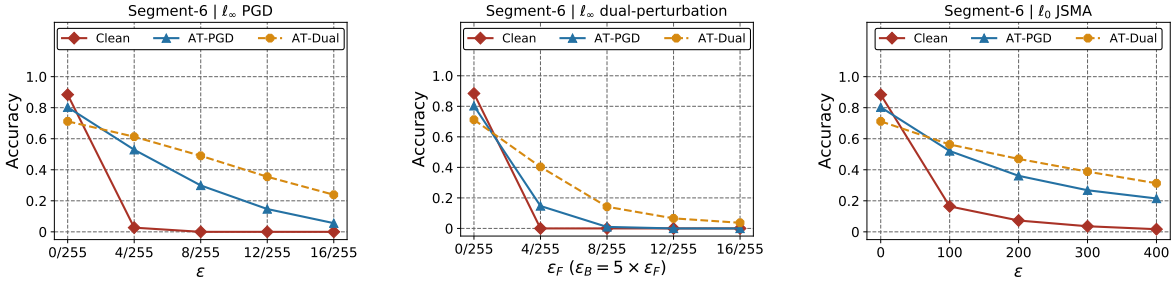

Figure B.8: Robustness to additional white-box attacks on Segment-6. Left: 20 steps of ℓ_∞ PGD attacks. Middle: 20 steps of ℓ_∞ dual-perturbation attacks with different foreground and background distortions. Right: ℓ_0 JSMA attacks.

B.6 Adversarial Training Using ℓ_∞ Attacks on ImageNet-10

Next, we present experimental results of the robustness of classifiers that use adversarial training with ℓ_∞ norm attacks on ImageNet-10. Specifically, we trained AT-PGD using ℓ_∞ PGD attack with $\epsilon = 4/255$, and AT-Dual by using ℓ_∞ dual-perturbation attack with

$\{\epsilon_F, \epsilon_B, \lambda\} = \{4/255, 20/255, 0.0\}$. The results are shown in Figure B.9, B.10, B.11, and B.12.

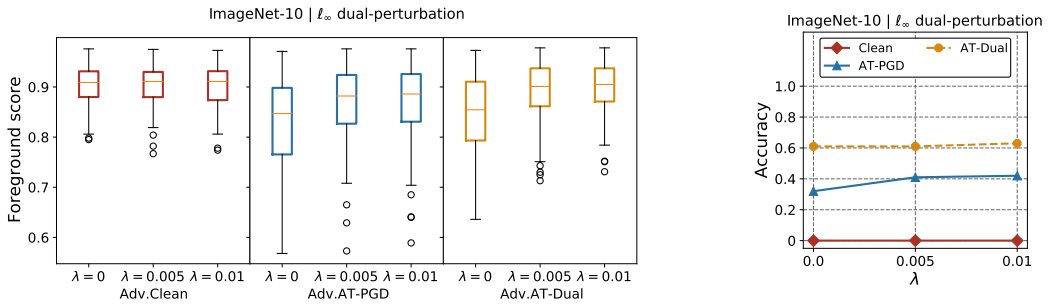

Figure B.9: Saliency analysis. The ℓ_∞ dual-perturbation attacks are performed by using $\{\epsilon_F, \epsilon_B\} = \{4/255, 20/255\}$, and a variety of λ displayed in the figure. Left: foreground scores of dual-perturbation examples in response to different classifiers. Right: accuracy of classifiers on dual-perturbation examples with salience control.

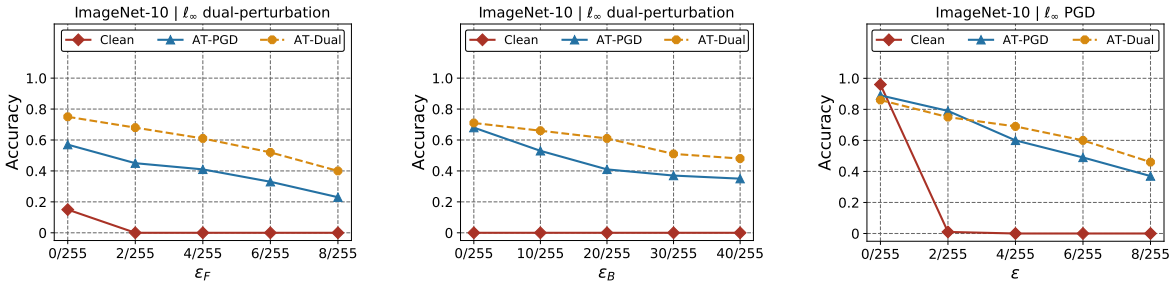

Figure B.10: Robustness to white-box ℓ_∞ attacks on ImageNet-10. Left: ℓ_∞ dual-perturbation attacks with different foreground distortions. ϵ_B is fixed to be $20/255$ and $\lambda = 0.005$. Middle: ℓ_∞ dual-perturbation attacks with different background distortions. ϵ_F is fixed to be $4/255$ and $\lambda = 0.005$. Right: ℓ_∞ PGD attacks.

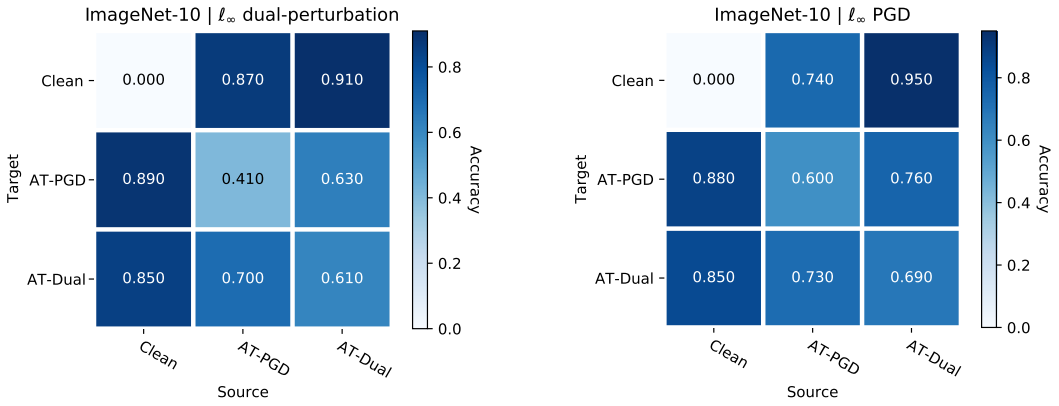

Figure B.11: Robustness against adversarial examples transferred from other models on ImageNet-10. Left: ℓ_∞ dual-perturbation attacks performed by using $\{\epsilon_F, \epsilon_B, \lambda\} = \{4/255, 20/255, 0.005\}$ on different source models. Right: ℓ_∞ PGD attacks with $\epsilon = 4/255$ on different source models.

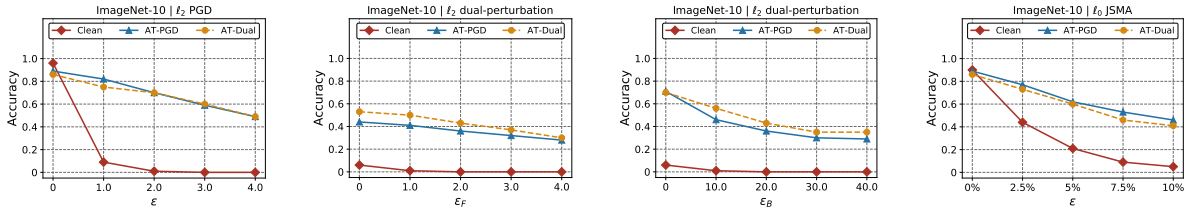

Figure B.12: Robustness to additional white-box attacks on ImageNet-10. Left: 100 steps of ℓ_2 PGD attacks. Middle left: 100 steps of ℓ_2 dual-perturbation attacks with different foreground distortions. ϵ_B is fixed to be 2.0 and $\lambda = 0.005$. Middle right: 100 steps of ℓ_2 dual-perturbation attacks with different background distortions. ϵ_F is fixed to be 20.0 and $\lambda = 0.005$. Right: ℓ_0 JSMA attacks.

B.7 Adversarial Training Using ℓ_∞ Attacks on STL-10

Now, we present experimental results of the robustness of classifiers that use adversarial training with ℓ_∞ norm attacks on STL-10. Specifically, we trained AT-PGD using ℓ_∞ PGD attack with $\epsilon = 4/255$, and AT-Dual by using ℓ_∞ dual-perturbation attack with $\{\epsilon_F, \epsilon_B, \lambda\} = \{4/255, 20/255, 0.0\}$. The results are shown in Figure B.13, B.14, B.15, and B.16.

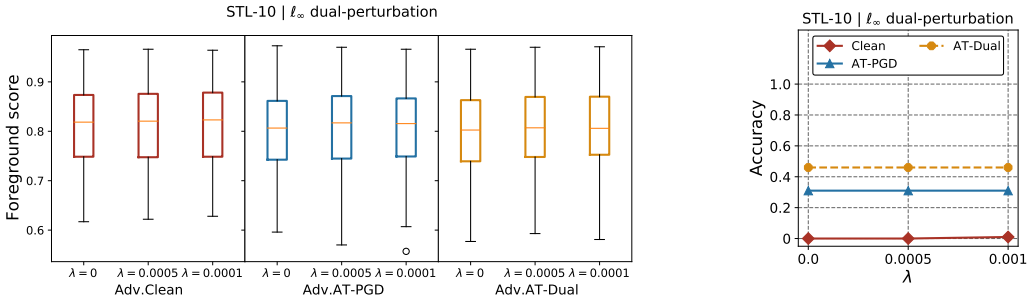

Figure B.13: Saliency analysis. The ℓ_∞ dual-perturbation attacks are performed by using $\{\epsilon_F, \epsilon_B\} = \{4/255, 20/255\}$, and a variety of λ displayed in the figure. Left: foreground scores of dual-perturbation examples in response to different classifiers. Right: accuracy of classifiers on dual-perturbation examples with salience control.

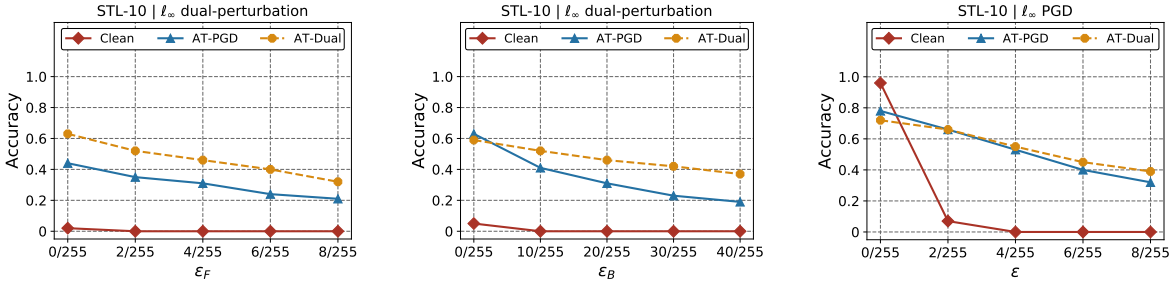

Figure B.14: Robustness to white-box ℓ_∞ attacks on STL-10. Left: ℓ_∞ dual-perturbation attacks with different foreground distortions. ϵ_B is fixed to be $20/255$ and $\lambda = 0.0005$. Middle: ℓ_∞ dual-perturbation attacks with different background distortions. ϵ_F is fixed to be $4/255$ and $\lambda = 0.0005$. Right: ℓ_∞ PGD attacks.

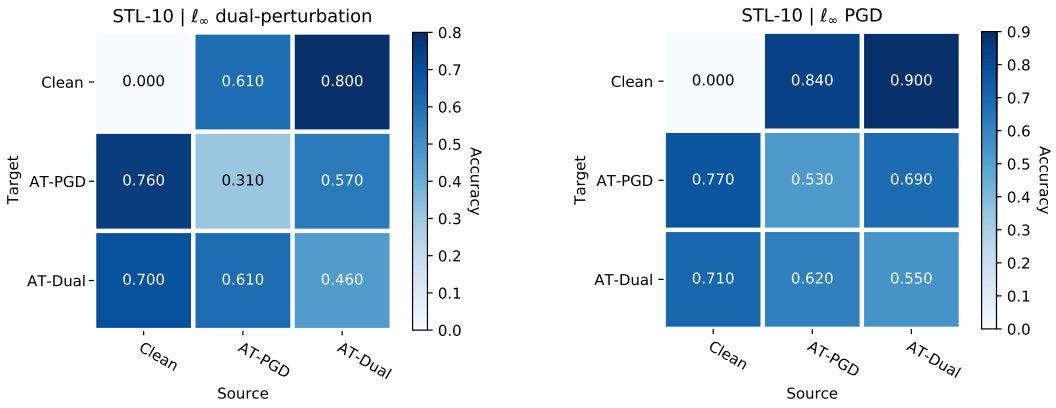

Figure B.15: Robustness against adversarial examples transferred from other models on STL-10. Left: ℓ_∞ dual-perturbation attacks performed by using $\{\epsilon_F, \epsilon_B, \lambda\} = \{4/255, 20/255, 0.0005\}$ on different source models. Right: ℓ_∞ PGD attacks with $\epsilon = 4/255$ on different source models.

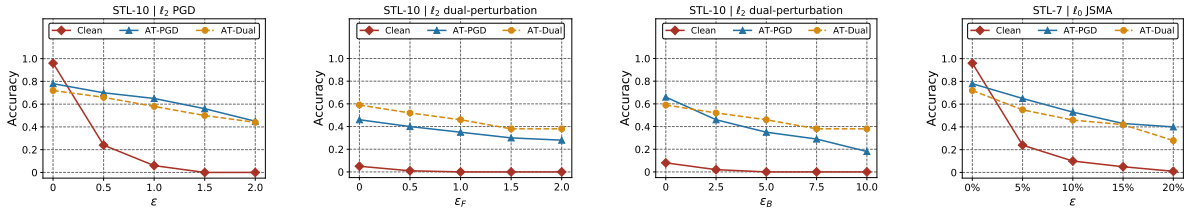

Figure B.16: Robustness to additional white-box attacks on STL-10. Left: 100 steps of ℓ_2 PGD attacks. Middle left: 100 steps of ℓ_2 dual-perturbation attacks with different foreground distortions. ϵ_B is fixed to be 5.0 and $\lambda = 0.0005$. Middle right: 100 steps of ℓ_2 dual-perturbation attacks with different background distortions. ϵ_F is fixed to be 1.0 and $\lambda = 0.0005$. Right: ℓ_0 JSMA attacks.

B.8 Adversarial Training Using ℓ_∞ Attacks on Segment-6

Finally, we present experimental results of the robustness of classifiers that use adversarial training with ℓ_∞ norm attacks on Segment-6. We trained AT-PGD using ℓ_∞ PGD attack with $\epsilon = 8/255$, and AT-Dual by using ℓ_∞ dual-perturbation attack with $\{\epsilon_F, \epsilon_B\} = \{8/255, 40/255\}$. The results are shown in Figure B.17, B.18, and B.19.

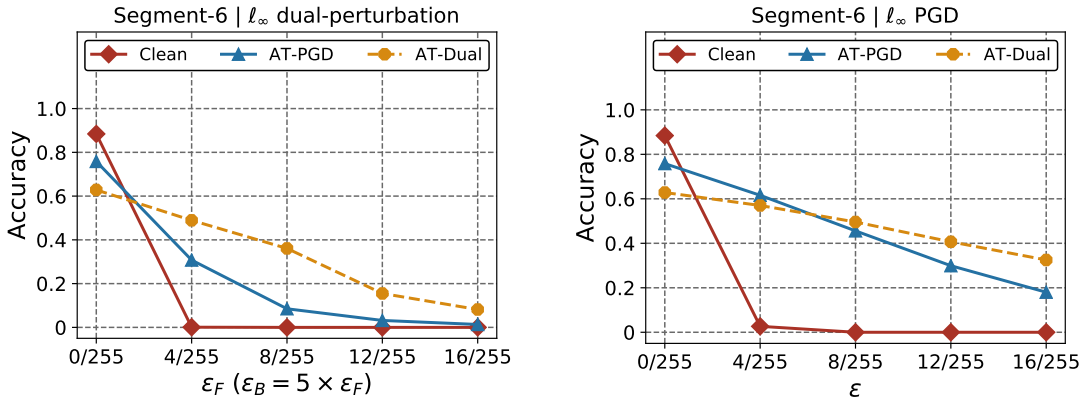

Figure B.17: Robustness to white-box ℓ_∞ attacks on Segment-6. Left: ℓ_∞ dual-perturbation attacks with different foreground and background distortions. Right: ℓ_∞ PGD attacks.

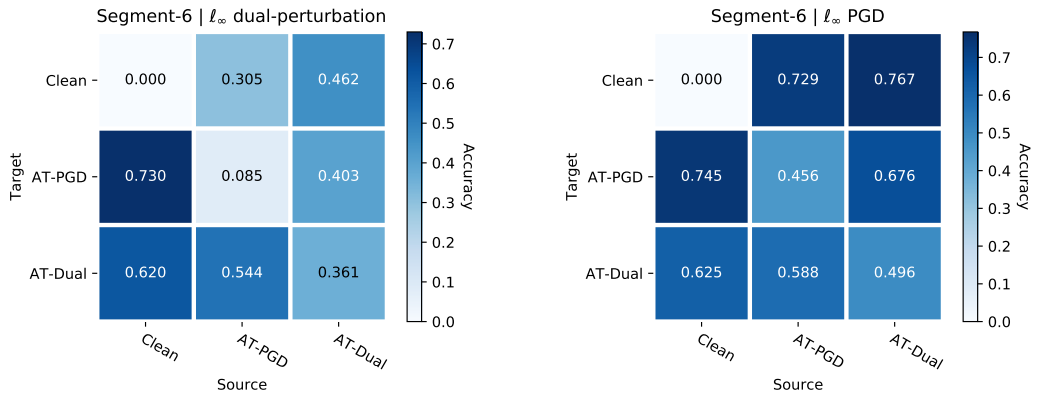

Figure B.18: Robustness against adversarial examples transferred from other models on Segment-6. Left: ℓ_∞ dual-perturbation attacks performed by using $\{\epsilon_F, \epsilon_B\} = \{8/255, 40/255\}$ on different source models. Right: ℓ_∞ PGD attacks with $\epsilon = 8/255$ on different source models.

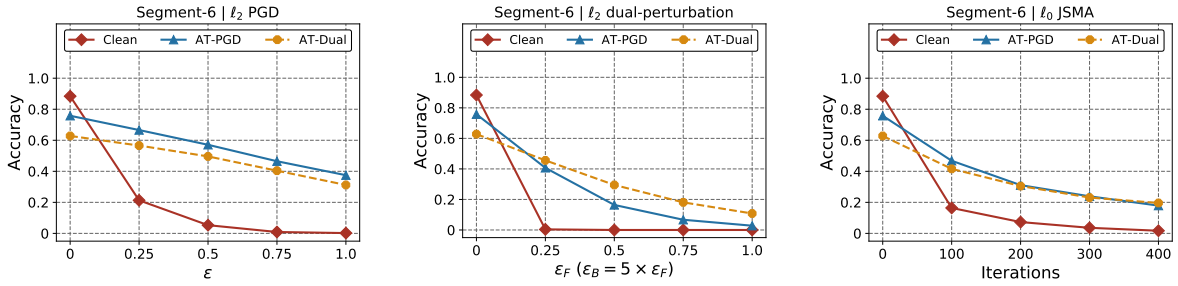

Figure B.19: Robustness to additional white-box attacks on Segment-6. Left: 100 steps of ℓ_2 PGD attacks. Middle: 100 steps of ℓ_2 dual-perturbation attacks with different foreground and background distortions. Right: ℓ_0 JSMA attacks.

B.9 Attacking Randomzied Classifiers

In addition to *deterministic classifiers* that make a deterministic prediction for a test sample, our proposed attack can be adapted to *stochastic classifiers* that apply randomization at training and prediction time. For example, for classifiers using *randomized smoothing*, we can

refine Eq. (6.1) in Chapter 6 as follows:

$$\max_{\substack{||\boldsymbol{\delta}\circ\mathcal{F}(\boldsymbol{x})||_p\leq\epsilon_F,\\ ||\boldsymbol{\delta}\circ\mathcal{B}(\boldsymbol{x})||_p\leq\epsilon_B}} \mathbb{E}_{\boldsymbol{\eta}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})}[\mathcal{L}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta}+\boldsymbol{\eta}),y\right)+\lambda\cdot\mathcal{S}\left(\boldsymbol{x}+\boldsymbol{\delta}+\boldsymbol{\eta}\right)], \tag{B.1}$$

where σ^2 is the variance of the Gaussian data augmentation in randomized smoothing.[22] The optimization problem in Eq. (B.1) can be solved by the same approach used for deterministic classifiers, with the following modification on Eq. (6.5) at the second step in Section 6.2.4:

$$\begin{cases} g_F = \mathcal{G}(\mathcal{F}(\boldsymbol{x})\circ\nabla_{\boldsymbol{\delta}^{(k)}}\mathbb{E}_{\boldsymbol{\eta}}[\mathcal{L}(h_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta}^{(k)}+\boldsymbol{\eta}),y)+\lambda\cdot\mathcal{S}\left(\boldsymbol{x}+\boldsymbol{\delta}^{(k)}+\boldsymbol{\eta}\right)]) \\ g_B = \mathcal{G}(\mathcal{B}(\boldsymbol{x})\circ\nabla_{\boldsymbol{\delta}^{(k)}}\mathbb{E}_{\boldsymbol{\eta}}[\mathcal{L}(h_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta}^{(k)}+\boldsymbol{\eta}),y)+\lambda\cdot\mathcal{S}\left(\boldsymbol{x}+\boldsymbol{\delta}^{(k)}+\boldsymbol{\eta}\right)]) \end{cases}. \tag{B.2}$$

B.9.1 Variance in Gaussian Data Augmentation

Table B.4 and B.5 show the effectiveness of *Randomized Smoothing (RS)* against the proposed dual-perturbation attack. Here, we use different variances in Gaussian data augmentation of *RS*, and fix the number of noise-corrupted copies at prediction time, n to be 100. It can be seen that *RS* is generally fragile to the dual-perturbation attacks that are adapted to randomized classifiers. Moreover, increasing σ, the variance used in Gaussian data augmentation can only marginally improve adversarial robustness to dual-perturbation attacks while significantly decrease accuracy on non-adversarial data.

B.9.2 Number of Samples with Gaussian Noise at Prediction Time

It has been observed that *Randomized Smoothing (RS)* can be computationally inefficient at prediction time as it uses a large number of noise-corrupted copies for each test sample

[22] Note that the Gaussian perturbations are only used to compute the expection of loss and are not in the resulting adversarial examples.

Table B.4: Robustness of RS against ℓ_∞ dual-perturbation attacks.

Dataset	**Defense approach**	**Attack Strength** ($\epsilon_B = 5 \times \epsilon_F$)				
		$\epsilon_F = 0/255$	$\epsilon_F = 4/255$	$\epsilon_F = 8/255$	$\epsilon_F = 12/255$	$\epsilon_F = 1$
Segment-6	RS, $\sigma = 0.25$	71.4%	9.6%	0.4%	0.1%	0.0%
	RS, $\sigma = 0.5$	61.7%	13.7%	1.9%	0.6%	0.2%
	RS, $\sigma = 1$	47.7%	15.6%	2.8%	0.4%	0.2%

Table B.5: Robustness of RS against ℓ_2 dual-perturbation attacks on Segment-6.

Defense approach	**Attack Strength** ($\epsilon_B = 5 \times \epsilon_F$)				
	$\epsilon_F = 0$	$\epsilon_F = 0.25$	$\epsilon_F = 0.5$	$\epsilon_F = 0.75$	$\epsilon_F = 1$
RS, $\sigma = 0.25$	71.4%	29.7%	6.7%	0.9%	0.1%
RS, $\sigma = 0.5$	61.7%	31.6%	11.8%	3.1%	1.3%
RS, $\sigma = 1$	47.7%	28.2%	14.4%	6.0%	1.5%

at prediction time. It is natural to ask whether the prediction time of RS can be reduced without significantly sacrificing adversarial robustness in practice. We answer this question by studying the effectiveness of RS with different n, the numbers of noise-corrupted copies at prediction time. Specifically, we fix $\sigma = 0.5$ and set n to be 1, 25, and 100. Note that when $n = 1$, there is no two-sided hypothesis test for prediction; thus, no abstentions are obtained.

Here we use ℓ_∞ dual-perturbation attacks on RS for demonstration purposes. The results are shown in Table B.6. It can be seen that when $n = 25$, the accuracy on both adversarial and non-adversarial data can drop by up to 10% compared to RS using $n = 100$. The reason is that under a small n, the prediction appears more likely to abstain. Interestingly, when $n = 1$, the accuracy can be marginally improved compared to $n = 100$, with the prediction time being reduced by 99%. This indicates that in practice, we would not lose accuracy without using the two-sided hypothesis test at prediction time.

Table B.6: Robustness of *RS* against ℓ_∞ dual-perturbation attacks under different numbers of noise-corrupted copies at prediction time.

Dataset	**Defense approach**	**Attack Strength** ($\epsilon_B = 5 \times \epsilon_F$)				
		$\epsilon_F = 0/255$	$\epsilon_F = 4/255$	$\epsilon_F = 8/255$	$\epsilon_F = 12/255$	$\epsilon_F = 1$
Segment-6	RS, $n = 1$	66.0%	19.8%	3.2%	0.8%	0.3%
	RS, $n = 25$	49.4%	9.1%	1.3%	0.5%	0.0%
	RS, $n = 100$	61.7%	13.7%	1.9%	0.6%	0.2%

B.10 Visualization of Loss Gradient

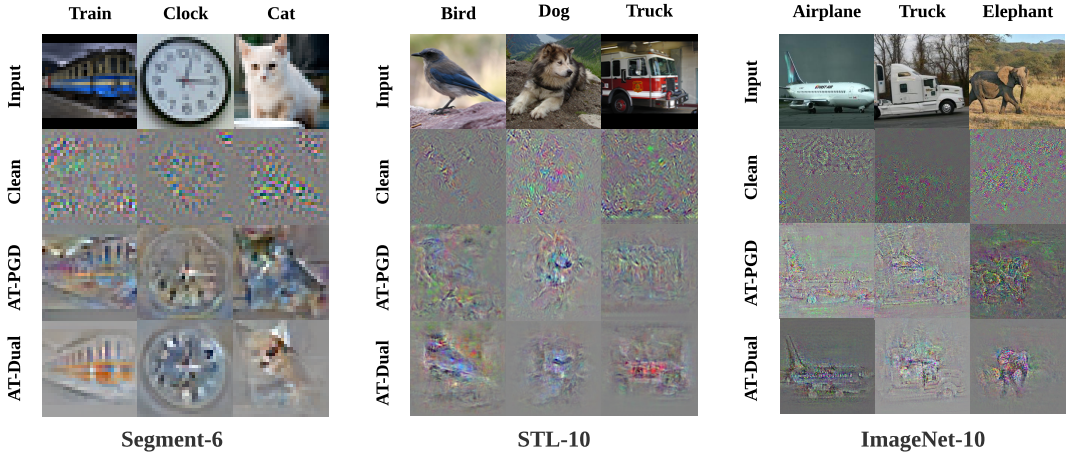

Figure B.20: Visualization of loss gradient of different classifiers with respect to pixels of *non-adversarial* inputs. AT-PGD and AT-Dual were obtained using adversarial training with corresponding ℓ_2 norm attacks.

B.11 Examples of Dual-Perturbation Attacks

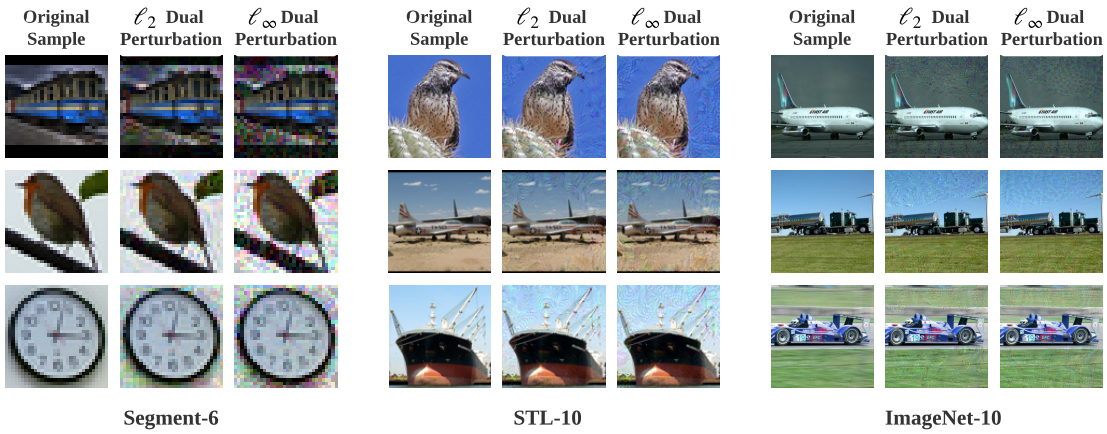

Figure B.21: Dual-perturbation attacks. Adversarial examples are produced in response to the *Clean* model for each dataset.

Appendix C

Supplement for Chapter 8

C.1 Proof of Lemma 2

Proof. 1. First, we prove that $\mathbf{A}_n = \lambda\mathbf{I} + \sum_{i=1}^{n} \boldsymbol{\theta}_i\boldsymbol{\theta}_i^\top$ is invertible, and its inverse matrix, $\mathbf{A}_n^{-1}$, is positive definite by using mathematical induction.

When $n = 1$, $\mathbf{A}_1 = \lambda\mathbf{I} + \boldsymbol{\theta}_1\boldsymbol{\theta}_1^\top$. As $\lambda\mathbf{I}$ is an invertible square matrix and $\boldsymbol{\theta}_1$ is a column vector, by using *Sherman-Morrison formula*, $\mathbf{A}_1$ is invertible.

$$\mathbf{A}_1^{-1} = \frac{1}{\lambda}(\mathbf{I} - \frac{\boldsymbol{\theta}_1\boldsymbol{\theta}_1^\top}{\lambda + \boldsymbol{\theta}_1^\top\boldsymbol{\theta}_1}).$$

For any non-zero column vector $\mathbf{u}$, we have

$$\mathbf{u}^\top\mathbf{A}_1^{-1}\mathbf{u} = \frac{\lambda\mathbf{u}^\top\mathbf{u} + \mathbf{u}^\top\mathbf{u}\boldsymbol{\theta}_1^\top\boldsymbol{\theta}_1 - \mathbf{u}^\top\boldsymbol{\theta}_1\boldsymbol{\theta}_1^\top\mathbf{u}}{\lambda(\lambda + \boldsymbol{\theta}_1^\top\boldsymbol{\theta}_1)}.$$

As $\mathbf{u}^\top \mathbf{u} > 0$ and $\lambda > 0$, according to *Cauchy-Schwarz inequality*,

$$\mathbf{u}^\top \mathbf{u} \boldsymbol{\theta}_1^\top \boldsymbol{\theta}_1 \geq \mathbf{u}^\top \boldsymbol{\theta}_1 \boldsymbol{\theta}_1^\top \mathbf{u},$$

Then, $\mathbf{u}^\top \mathbf{A}_1^{-1} \mathbf{u} > 0$. Thus, $\mathbf{A}_1^{-1}$ is a positive definite matrix.

We then assume that when $n = k (k \geq 1)$, $\mathbf{A}_k$ is invertible and $\mathbf{A}_k^{-1}$ is positive definite. Then, when $n = k + 1$,

$$\mathbf{A}_{k+1} = \mathbf{A}_k + \boldsymbol{\theta}_{k+1} \boldsymbol{\theta}_{k+1}^\top.$$

As $\mathbf{A}_k$ is invertible, $\boldsymbol{\theta}_{k+1}$ is a column vector. By using *Sherman-Morrison formula*, we have that $\mathbf{A}_{k+1}$ is invertible, and

$$\mathbf{A}_{k+1}^{-1} = \mathbf{A}_k^{-1} - \frac{\mathbf{A}_k^{-1} \boldsymbol{\theta}_{k+1} \boldsymbol{\theta}_{k+1}^\top \mathbf{A}_k^{-1}}{1 + \boldsymbol{\theta}_{k+1}^\top \mathbf{A}_k^{-1} \boldsymbol{\theta}_{k+1}}.$$

Then,

$$\mathbf{u}^\top \mathbf{A}_{k+1}^{-1} \mathbf{u} = \frac{\mathbf{u}^\top \mathbf{A}_k^{-1} \mathbf{u} + \mathbf{u}^\top \mathbf{A}_k^{-1} \mathbf{u} \cdot \boldsymbol{\theta}_{k+1}^\top \mathbf{A}_k^{-1} \boldsymbol{\theta}_{k+1} - \mathbf{u}^\top \mathbf{A}_k^{-1} \boldsymbol{\theta}_{k+1} \cdot \boldsymbol{\theta}_{k+1}^\top \mathbf{A}_k^{-1} \mathbf{u}}{1 + \boldsymbol{\theta}_{k+1}^\top \mathbf{A}_k^{-1} \boldsymbol{\theta}_{k+1}}$$

As $\mathbf{A}_k^{-1}$ is a positive definite matrix, we have $\mathbf{u}^\top \mathbf{A}_k^{-1} \mathbf{u} > 0$ and $\boldsymbol{\theta}_{k+1}^\top \mathbf{A}_k^{-1} \boldsymbol{\theta}_{k+1} > 0$. By using *Extended Cauchy-Schwarz inequality*, we have

$$\mathbf{u}^\top \mathbf{A}_k^{-1} \mathbf{u} \boldsymbol{\theta}_{k+1}^\top \mathbf{A}_k^{-1} \boldsymbol{\theta}_{k+1} > \mathbf{u}^\top \mathbf{A}_k^{-1} \boldsymbol{\theta}_{k+1} \boldsymbol{\theta}_{k+1}^\top \mathbf{A}_k^{-1} \mathbf{u}.$$

Then, $\mathbf{A}_{k+1}^{-1}$ is positive definite. Hence, $\mathbf{A}_n = \lambda \mathbf{I} + \sum_{i=1}^n \boldsymbol{\theta}_i \boldsymbol{\theta}_i^\top$ is invertible, and $\mathbf{A}_n^{-1}$ is positive definite. Similarly, we can prove that $\mathbf{A}_{-i}$ is invertible, and $\mathbf{A}_{-i}^{-1}$ is positive definite.

2. We have proved that $\mathbf{A}_n$ and $\mathbf{A}_{-i}$ are invertible. Then, the result can be obtained by using *Sherman-Morrison formula*.

3. Let $\mathbf{A}_{-i,-j} = \mathbf{A}_{-i} - \boldsymbol{\theta}_j \boldsymbol{\theta}_j^\top$. As $\mathbf{A}_{-i,-j}$ is a symmetric matrix, its inverse, $\mathbf{A}_{-i,-j}^{-1}$ is also symmetric. Using a similar approach to the one above, we can prove that $\mathbf{A}_{-i,-j}$ is invertible and $\mathbf{A}_{-i,-j}^{-1}$ is positive definite. By using *Sherman-Morrison formula*, we have

$$\mathbf{A}_{-i}^{-1} = \mathbf{A}_{-i,-j}^{-1} - \frac{\mathbf{A}_{-i,-j}^{-1} \boldsymbol{\theta}_j \boldsymbol{\theta}_j^\top \mathbf{A}_{-i,-j}^{-1}}{1 + \boldsymbol{\theta}_j^\top \mathbf{A}_{-i,-j}^{-1} \boldsymbol{\theta}_j}.$$

Then,

$$\begin{aligned}
\boldsymbol{\theta}_i^\top \mathbf{A}_{-i}^{-1} \boldsymbol{\theta}_i &= \boldsymbol{\theta}_i^\top \mathbf{A}_{-i,-j}^{-1} \boldsymbol{\theta}_i - \frac{\boldsymbol{\theta}_i^\top \mathbf{A}_{-i,-j}^{-1} \boldsymbol{\theta}_j \cdot \boldsymbol{\theta}_j^\top \mathbf{A}_{-i,-j}^{-1} \boldsymbol{\theta}_i}{1 + \boldsymbol{\theta}_j^\top \mathbf{A}_{-i,-j}^{-1} \boldsymbol{\theta}_j} \\
&= \boldsymbol{\theta}_i^\top \mathbf{A}_{-i,-j}^{-1} \boldsymbol{\theta}_i - \frac{(\boldsymbol{\theta}_i^\top \mathbf{A}_{-i,-j}^{-1} \boldsymbol{\theta}_j)^2}{1 + \boldsymbol{\theta}_j^\top \mathbf{A}_{-i,-j}^{-1} \boldsymbol{\theta}_j} \\
&\leq \boldsymbol{\theta}_i^\top \mathbf{A}_{-i,-j}^{-1} \boldsymbol{\theta}_i.
\end{aligned}$$

We then iteratively apply *Sherman-Morrison formula* and get

$$\begin{aligned}
\boldsymbol{\theta}_i^\top \mathbf{A}_{-i}^{-1} \boldsymbol{\theta}_i &\leq \boldsymbol{\theta}_i^\top \mathbf{A}_0^{-1} \boldsymbol{\theta}_i \\
&= \frac{1}{\lambda} \boldsymbol{\theta}_i^\top \boldsymbol{\theta}_i.
\end{aligned}$$

□

C.2 Proof of Lemma 3

Proof. Firstly, by using *Sherman-Morrison formula* we have

$$\begin{aligned}
&\mathbf{X}^*\boldsymbol{\theta}_i = \mathbf{B}_n\mathbf{A}_n^{-1}\boldsymbol{\theta}_i \\
&= (\mathbf{B}_{-i} + \mathbf{z}\boldsymbol{\theta}_i^\top)(\mathbf{A}_{-i}^{-1} - \frac{\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i\boldsymbol{\theta}_i^\top\mathbf{A}_{-i}^{-1}}{1+\boldsymbol{\theta}_i^\top\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i})\boldsymbol{\theta}_i \\
&= \mathbf{B}_{-i}\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i + \frac{\mathbf{z}\boldsymbol{\theta}_i^\top\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i - \mathbf{B}_{-i}\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i\boldsymbol{\theta}_i^\top\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i}{1+\boldsymbol{\theta}_i^\top\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i} \\
&= \frac{(\mathbf{B}_{-i} + \mathbf{z}\boldsymbol{\theta}_i^\top)\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i}{1+\boldsymbol{\theta}_i^\top\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i} \\
&= \frac{\mathbf{B}_n\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i}{1+\boldsymbol{\theta}_i^\top\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i}.
\end{aligned}$$

Then,

$$\begin{aligned}
\ell(\mathbf{X}^*\boldsymbol{\theta}_i, \mathbf{y}) &= ||\frac{\mathbf{B}_n\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i}{1+\boldsymbol{\theta}_i^\top\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i} - \mathbf{y}||_2^2 \\
&= ||\frac{\mathbf{B}_n\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i - \mathbf{y} - \boldsymbol{\theta}_i^\top\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i\mathbf{y}}{1+\boldsymbol{\theta}_i^\top\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i}||_2^2 \\
&\leq ||\mathbf{B}_n\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i - \mathbf{y} - \boldsymbol{\theta}_i^\top\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i\mathbf{y}||_2^2 \\
&= ||(\mathbf{B}_{-i} + \mathbf{z}\boldsymbol{\theta}_i^\top)\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i - \mathbf{y} - \boldsymbol{\theta}_i^\top\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i\mathbf{y}||_2^2 \\
&= ||\mathbf{B}_{-i}\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i - \mathbf{y} + (\mathbf{z}-\mathbf{y})\boldsymbol{\theta}_i^\top\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i||_2^2 \\
&\leq \ell(\mathbf{B}_{-i}\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i, \mathbf{y}) + ||\mathbf{z}-\mathbf{y}||_2^2(\boldsymbol{\theta}_i^\top\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i)^2
\end{aligned}$$

By using Lemma 2, we have $(\boldsymbol{\theta}_i^\top\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i)^2 \leq \frac{1}{\lambda^2}(\boldsymbol{\theta}_i^\top\boldsymbol{\theta}_i)^2$ which completes the proof. □

C.3 Proof of Theorem 2

Proof. As presented in Lemma 3, we have

$$\ell(\mathbf{X}^*\boldsymbol{\theta}_i, \mathbf{y}) \leq \ell(\mathbf{B}_{-i}\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i, \mathbf{y}) + \frac{1}{\lambda^2}||\mathbf{z}-\mathbf{y}||_2^2(\boldsymbol{\theta}_i^\top\boldsymbol{\theta}_i)^2.$$

By using *Sherman-Morrison formula*,

$$\begin{aligned}\ell(\mathbf{B}_{-i}\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i, \mathbf{y}) &= ||\mathbf{B}_{-i}(\mathbf{A}_{-i,-j}^{-1} - \frac{\mathbf{A}_{-i,-j}^{-1}\boldsymbol{\theta}_j\boldsymbol{\theta}_j^\top\mathbf{A}_{-i,-j}^{-1}}{1+\boldsymbol{\theta}_j^\top\mathbf{A}_{-i,-j}^{-1}\boldsymbol{\theta}_j})\boldsymbol{\theta}_i - \mathbf{y}||_2^2 \\ &\leq ||\frac{\mathbf{B}_{-i}\mathbf{A}_{-i,-j}^{-1}\boldsymbol{\theta}_i}{1+\boldsymbol{\theta}_j^\top\mathbf{A}_{-i,-j}^{-1}\boldsymbol{\theta}_j} - \mathbf{y}||_2^2 + \triangle_1(\boldsymbol{\theta})\end{aligned}$$

where $j \neq i$, and $\triangle_1(\boldsymbol{\theta})$ is a continuous function of $\boldsymbol{\theta} = \{\boldsymbol{\theta}_i\}_{i=1}^n$. As the action space $\boldsymbol{\Theta}$ is bounded, then $0 \leq \triangle_1(\boldsymbol{\theta}) < \infty$. Hence, we have

$$\begin{aligned}\ell(\mathbf{B}_{-i}\mathbf{A}_{-i}^{-1}\boldsymbol{\theta}_i, \mathbf{y}) &\leq ||\frac{\mathbf{B}_{-i}\mathbf{A}_{-i,-j}^{-1}\boldsymbol{\theta}_i}{1+\boldsymbol{\theta}_j^\top\mathbf{A}_{-i,-j}^{-1}\boldsymbol{\theta}_j} - \mathbf{y}||_2^2 + \triangle_1(\boldsymbol{\theta}) \\ &= ||\frac{\mathbf{B}_{-i}\mathbf{A}_{-i,-j}^{-1}\boldsymbol{\theta}_i - \mathbf{y} - \boldsymbol{\theta}_j^\top\mathbf{A}_{-i,-j}^{-1}\boldsymbol{\theta}_j\mathbf{y}}{1+\boldsymbol{\theta}_j^\top\mathbf{A}_{-i,-j}^{-1}\boldsymbol{\theta}_j}||_2^2 + \triangle_1(\boldsymbol{\theta}) \\ &\leq ||\mathbf{B}_{-i}\mathbf{A}_{-i,-j}^{-1}\boldsymbol{\theta}_i - \mathbf{y} - \boldsymbol{\theta}_j^\top\mathbf{A}_{-i,-j}^{-1}\boldsymbol{\theta}_j\mathbf{y}||_2^2 + \triangle_1(\boldsymbol{\theta}) \\ &= ||(\mathbf{B}_{-i,-j} + \mathbf{z}\boldsymbol{\theta}_j^\top)\mathbf{A}_{-i,-j}^{-1}\boldsymbol{\theta}_i - \mathbf{y} - \boldsymbol{\theta}_j^\top\mathbf{A}_{-i,-j}^{-1}\boldsymbol{\theta}_j\mathbf{y}||_2^2 + \triangle_1(\boldsymbol{\theta}) \\ &= ||(\mathbf{B}_{-i,-j}\mathbf{A}_{-i,-j}^{-1}\boldsymbol{\theta}_i - \mathbf{y}) + (\mathbf{z}-\mathbf{y})\boldsymbol{\theta}_j^\top\mathbf{A}_{-i,-j}^{-1}\boldsymbol{\theta}_i + \boldsymbol{\theta}_j^\top\mathbf{A}_{-i,-j}^\top(\boldsymbol{\theta}_i - \boldsymbol{\theta}_j)\mathbf{y}||_2^2 + \triangle_1(\boldsymbol{\theta}) \\ &\leq \ell(\mathbf{B}_{-i,-j}\mathbf{A}_{-i,-j}^{-1}\boldsymbol{\theta}_i, \mathbf{y}) + ||(\mathbf{z}-\mathbf{y})||_2^2(\boldsymbol{\theta}_j^\top\mathbf{A}_{-i,-j}^{-1}\boldsymbol{\theta}_i)^2 + \triangle_2(\boldsymbol{\theta})\end{aligned}$$

where $\triangle_2(\boldsymbol{\theta})$ is a continuous function of $\boldsymbol{\theta}$ and $0 \leq \triangle_2(\boldsymbol{\theta}) < \infty$. Let $\mathbf{A}_{-i,-j,-k} = \mathbf{A}_{-i,-j} - \boldsymbol{\theta}_k \boldsymbol{\theta}_k^\top$, then, similarly, $(\boldsymbol{\theta}_j^\top \mathbf{A}_{-i,-j}^{-1} \boldsymbol{\theta}_i)^2$ can be further relaxed as follows.

$$\begin{aligned}(\boldsymbol{\theta}_j^\top \mathbf{A}_{-i,-j}^{-1} \boldsymbol{\theta}_i)^2 &= (\boldsymbol{\theta}_j^\top (\mathbf{A}_{-i,-j,-k}^{-1} - \frac{\mathbf{A}_{-i,-j,-k}^{-1} \boldsymbol{\theta}_k \boldsymbol{\theta}_k^\top \mathbf{A}_{-i,-j,-k}^{-1}}{1 + \boldsymbol{\theta}_k^\top \mathbf{A}_{-i,-j,-k}^{-1} \boldsymbol{\theta}_k}) \boldsymbol{\theta}_i)^2 \\ &\leq (\boldsymbol{\theta}_j^\top \mathbf{A}_{-i,-j,-k}^{-1} \boldsymbol{\theta}_i)^2 + \triangle_3(\boldsymbol{\theta})\end{aligned}$$

where $0 \leq \triangle_3(\boldsymbol{\theta}) < \infty$, using the same approach,$(\boldsymbol{\theta}_j^\top \mathbf{A}_{-i,-j}^{-1} \boldsymbol{\theta}_i)^2$ can be further and iteratively relaxed as follows,

$$\begin{aligned}(\boldsymbol{\theta}_j^\top \mathbf{A}_{-i,-j}^{-1} \boldsymbol{\theta}_i)^2 &\leq (\boldsymbol{\theta}_j^\top \mathbf{A}_0^{-1} \boldsymbol{\theta}_i)^2 + \triangle_4(\boldsymbol{\theta}) \\ &= \frac{1}{\lambda^2} (\boldsymbol{\theta}_j^\top \boldsymbol{\theta}_i)^2 + \triangle_4(\boldsymbol{\theta})\end{aligned}$$

where $0 \leq \triangle_4(\boldsymbol{\theta}) < \infty$. Combining the results above, we can iteratively relax $\ell(\mathbf{B}_{-i} \mathbf{A}_{-i}^{-1} \boldsymbol{\theta}_i, \mathbf{y})$ as follows,

$$\begin{aligned}\ell(\mathbf{B}_{-i} \mathbf{A}_{-i}^{-1} \boldsymbol{\theta}_i, \mathbf{y}) &\leq \ell(\mathbf{B}_{-i,-j} \mathbf{A}_{-i,-j}^{-1} \boldsymbol{\theta}_i, \mathbf{y}) + \frac{1}{\lambda^2} ||\mathbf{z} - \mathbf{y}||_2^2 (\boldsymbol{\theta}_j^\top \boldsymbol{\theta}_i)^2 + \triangle_5(\boldsymbol{\theta}) \\ &\leq \ell(\mathbf{X} \boldsymbol{\theta}_i, \mathbf{y}) + \frac{1}{\lambda^2} ||\mathbf{z} - \mathbf{y}||_2^2 \sum_{j \neq i} (\boldsymbol{\theta}_j^\top \boldsymbol{\theta}_i)^2 + \triangle(\boldsymbol{\theta})\end{aligned}$$

where $0 \leq \triangle_5(\boldsymbol{\theta}) < \infty$ and $0 \leq \triangle(\boldsymbol{\theta}) < \infty$. Then,

$$\begin{aligned}\ell(\mathbf{X}^* \boldsymbol{\theta}_i, \mathbf{y}) &\leq \ell(\mathbf{B}_{-i} \mathbf{A}_{-i}^{-1} \boldsymbol{\theta}_i, \mathbf{y}) + \frac{1}{\lambda^2} ||\mathbf{z} - \mathbf{y}||_2^2 (\boldsymbol{\theta}_i^\top \boldsymbol{\theta}_i)^2 \\ &\leq \ell(\mathbf{X} \boldsymbol{\theta}_i, \mathbf{y}) + \frac{1}{\lambda^2} ||\mathbf{z} - \mathbf{y}||_2^2 \sum_{j=1}^{n} (\boldsymbol{\theta}_j^\top \boldsymbol{\theta}_i)^2 + \triangle(\boldsymbol{\theta}).\end{aligned}$$

Hence,

$$
\begin{aligned}
c_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i}) &= \beta \ell(\mathbf{X}^* \boldsymbol{\theta}_i, \mathbf{y}) + (1-\beta)\ell(\mathbf{X}\boldsymbol{\theta}_i, \mathbf{y}) \\
&\leq \ell(\mathbf{X}\boldsymbol{\theta}_i, \mathbf{y}) + \frac{\beta}{\lambda^2} ||\mathbf{z}-\mathbf{y}||_2^2 \sum_{j=1}^{n} (\boldsymbol{\theta}_j^\top \boldsymbol{\theta}_i)^2 + \epsilon
\end{aligned}
$$

where ϵ is a constant such that $\epsilon = \beta * \max_{\boldsymbol{\theta}}\{\triangle(\boldsymbol{\theta})\} < \infty$. □

C.4 Proof of Theorem 4

Proof. We have known that $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i)\rangle$ has at least NE, and each learner has an nonempty, compact and convex action space $\boldsymbol{\Theta}$. Hence, we can apply Theorem 2 and Theorem 6 of [92]. That is, for some fixed $\{r_i\}_i^n (0 < r_i < 1, \sum_{i=1}^n r_i = 1)$, if the matrix in Eq. (C.1) is positive definite, then $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i)\rangle$ has a unique NE.

$$
Jr(\boldsymbol{\theta}) = \begin{bmatrix} r_1 \nabla_{\boldsymbol{\theta}_1, \boldsymbol{\theta}_1} \widetilde{c}_1(\boldsymbol{\theta}) & \dots & r_1 \nabla_{\boldsymbol{\theta}_1, \boldsymbol{\theta}_n} \widetilde{c}_1(\boldsymbol{\theta}) \\ \vdots & & \vdots \\ r_n \nabla_{\boldsymbol{\theta}_n, \boldsymbol{\theta}_1} \widetilde{c}_n(\boldsymbol{\theta}) & \dots & r_n \nabla_{\boldsymbol{\theta}_n, \boldsymbol{\theta}_n} \widetilde{c}_n(\boldsymbol{\theta}) \end{bmatrix} \tag{C.1}
$$

By taking second-order derivatives, we have

$$
\nabla_{\boldsymbol{\theta}_i, \boldsymbol{\theta}_i} \widetilde{c}_i(\boldsymbol{\theta}) = 2\mathbf{X}^\top \mathbf{X} + \frac{2\beta ||\mathbf{z}-\mathbf{y}||_2^2}{\lambda^2} (4\boldsymbol{\theta}_i \boldsymbol{\theta}_i^\top + 2\boldsymbol{\theta}_i^\top \boldsymbol{\theta}_i \mathbf{I} + \sum_{j \neq i} \boldsymbol{\theta}_j \boldsymbol{\theta}_j^\top)
$$

and

$$
\nabla_{\boldsymbol{\theta}_i, \boldsymbol{\theta}_j} \widetilde{c}_i(\boldsymbol{\theta}) = \frac{2\beta ||\mathbf{z}-\mathbf{y}||_2^2}{\lambda^2} (\boldsymbol{\theta}_i^\top \boldsymbol{\theta}_j \mathbf{I} + \boldsymbol{\theta}_j \boldsymbol{\theta}_i^\top)
$$

We first let $r_1 = r_2 = ... = r_n = \frac{1}{n}$ and decompose $Jr(\boldsymbol{\theta})$ as follows,

$$Jr(\boldsymbol{\theta}) = \frac{2}{n}\mathbf{P} + \frac{2\beta||\mathbf{z}-\mathbf{y}||_2^2}{\lambda^2 n}(\mathbf{Q}+\mathbf{S}+\mathbf{T}), \tag{C.2}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are *block diagonal matrices* such that $\mathbf{P}_{ii} = \mathbf{X}^\top\mathbf{X}$, $\mathbf{P}_{ij} = \mathbf{0}$, $\mathbf{Q}_{ii} = 4\boldsymbol{\theta}_i\boldsymbol{\theta}_i^\top + \boldsymbol{\theta}_i^\top\boldsymbol{\theta}_i\mathbf{I}$ and $\mathbf{Q}_{ij} = \mathbf{0}$, $\forall i, j \in \mathcal{N}, j \neq i$. $\mathbf{S}$ and $\mathbf{T}$ are *block symmetric matrices* such that $\mathbf{S}_{ii} = \boldsymbol{\theta}_i^\top\boldsymbol{\theta}_i\mathbf{I}$, $\mathbf{S}_{ij} = \boldsymbol{\theta}_i^\top\boldsymbol{\theta}_j\mathbf{I}$, $\mathbf{T}_{ii} = \sum_{j\neq i}\boldsymbol{\theta}_j\boldsymbol{\theta}_j^\top$ and $\mathbf{T}_{ij} = \boldsymbol{\theta}_j\boldsymbol{\theta}_i^\top$, $\forall i, j \in \mathcal{N}, j \neq i$.

Next, we prove that $\mathbf{P}$ is *positive definite*, and $\mathbf{Q}$, $\mathbf{S}$ and $\mathbf{T}$ are *positive semi-definite*. Let $\mathbf{u} = [\mathbf{u}_1^\top, ..., \mathbf{u}_n^\top]^\top$ be an $nd \times 1$ vector, where $\mathbf{u}_i \in \mathbb{R}^{d\times 1} (i \in \mathcal{N})$ are not all zero vectors.

1. $\mathbf{u}^\top\mathbf{P}\mathbf{u} = \sum_{i=1}^n \mathbf{u}_i^\top\mathbf{X}^\top\mathbf{X}\mathbf{u}_i = \sum_{i=1}^n ||\mathbf{X}\mathbf{u}_i||_2^2$. As the columns of $\mathbf{X}$ are linearly independent and $\mathbf{u}_i$ are not all zero vectors, there exists at least one $\mathbf{u}_i$ such that $\mathbf{X}\mathbf{u}_i \neq \mathbf{0}$. Hence, $\mathbf{u}^\top\mathbf{P}\mathbf{u} > 0$ which indicates that $\mathbf{P}$ is positive definite.

2. Similarly, $\mathbf{u}^\top\mathbf{Q}\mathbf{u} \geq 0$ which indicates that $\mathbf{Q}$ is a positive semi-definite matrix.

3. Let's $\mathbf{S}^* \in \mathbb{R}^{n\times n}$ be a symmetric matrix such that $\mathbf{S}^*_{ii} = \boldsymbol{\theta}_i^\top\boldsymbol{\theta}_i$ and $\mathbf{S}^*_{ij} = \boldsymbol{\theta}_i^\top\boldsymbol{\theta}_j$, $\forall i, j \in \mathcal{N}, j \neq i$. Hence, $\mathbf{S}_{ij} = \mathbf{S}^*_{ij}\mathbf{I}$, $\forall i, j \in \mathcal{N}$. Note that $\mathbf{S}^* = [\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_n]^\top[\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_n]$ is a positive semi-definite matrix, as it is also symmetric, there exists at least one lower triangular matrix $\mathbf{L}^* \in \mathbb{R}^{n\times n}$ with non-negative diagonal elements [41] such that

$$\mathbf{S}^* = \mathbf{L}^*\mathbf{L}^{*\top}(\text{Cholesky Decomposition})$$

Let $\mathbf{L}$ be a block matrix such that $\mathbf{L}_{ij} = \mathbf{L}^*_{ij}\mathbf{I}$, $\forall i, j \in \mathcal{N}$. Therefore, $(\mathbf{L}\mathbf{L}^\top)_{ij} = (\mathbf{L}^*\mathbf{L}^{*\top})_{ij}\mathbf{I} = \mathbf{S}^*_{ij}\mathbf{I} = \mathbf{S}_{ij}$ which indicates that $\mathbf{S} = \mathbf{L}\mathbf{L}^\top$ is a positive semi-definite matrix.

4. Since

$$\begin{aligned}\mathbf{u}^\top \mathbf{T}\mathbf{u} &= \sum_{i=1}^{n}\sum_{j\neq i}(\mathbf{u}_i^\top \boldsymbol{\theta}_j)^2 + \sum_{i=1}^{n}\sum_{j\neq i}(\mathbf{u}_i^\top \boldsymbol{\theta}_j)(\mathbf{u}_j^\top \boldsymbol{\theta}_i) \\ &= \sum_{i=1}^{n}\sum_{j\neq i}[\frac{1}{2}(\mathbf{u}_i^\top \boldsymbol{\theta}_j)^2 + \frac{1}{2}(\mathbf{u}_j^\top \boldsymbol{\theta}_i)^2 + (\mathbf{u}_i^\top \boldsymbol{\theta}_j)(\mathbf{u}_j^\top \boldsymbol{\theta}_i)] \\ &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j\neq i}(\mathbf{u}_i^\top \boldsymbol{\theta}_j + \mathbf{u}_j^\top \boldsymbol{\theta}_i)^2 \\ &\geq 0,\end{aligned}$$

$\mathbf{T}$ is a positive semi-definite matrix.

Combining the results above, $Jr(\boldsymbol{\theta})$ is a positive definite matrix which indicates that $\langle \mathcal{N}, \boldsymbol{\Theta}, (\widetilde{c}_i)\rangle$ has a unique NE. As Theorem 3 points out, the game has at least one symmetric NE. Therefore, the NE is unique and must be symmetric. □

C.5 Supplementary Results for The Red Wine Dataset

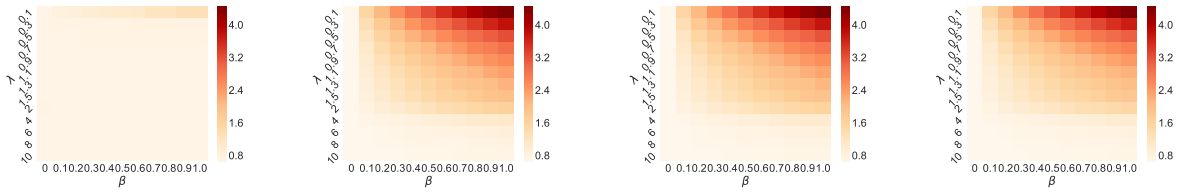

Figure C.1: Overestimated $\mathbf{z}$, $\hat{\lambda} = 0.5$, $\hat{\beta} = 0.8$.The average RMSE across different values of actual λ and β on the redwine dataset. From left to right: *MLSG*, *Lasso*, *Ridge*, *OLS*.

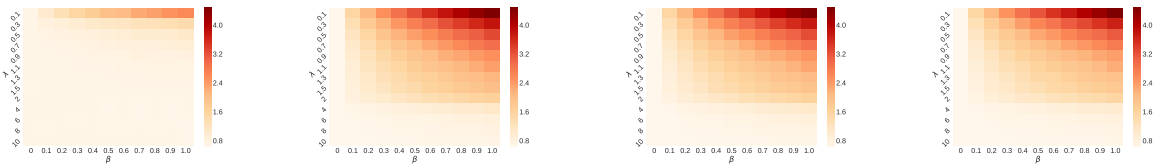

Figure C.2: Overestimated $\mathbf{z}$, $\hat{\lambda} = 1.5$, $\hat{\beta} = 0.8$. The average RMSE across different values of actual λ and β on the red wine dataset. From left to right: *MLSG*, *Lasso*, *Ridge*, *OLS*.

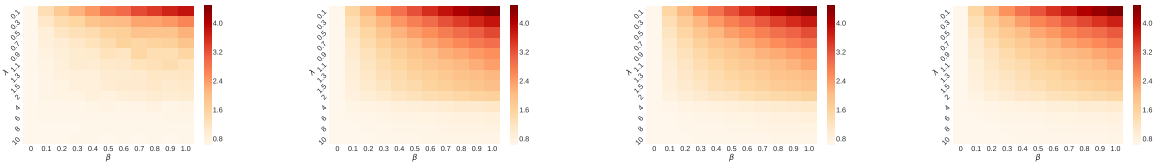

Figure C.3: Underestimated $\mathbf{z}$, $\hat{\lambda} = 1.5$, $\hat{\beta} = 0.8$. The average RMSE across different values of actual λ and β on the red wine dataset. From left to right: *MLSG*, *Lasso*, *Ridge*, *OLS*.

C.6 Supplementary Results for The Boston Dataset

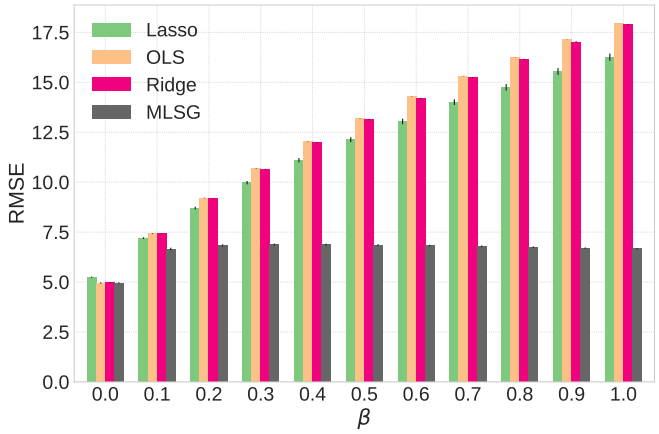

Figure C.4: The defender knows λ, β, and $\mathbf{z}$. RMSE of $\mathbf{y}'$ and $\mathbf{y}$ on the Boston dataset. The defender knows λ, β, and $\mathbf{z}$.

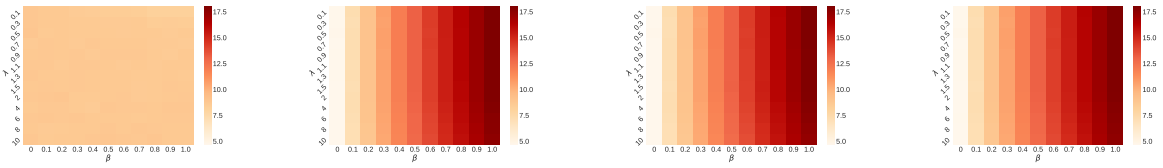

Figure C.5: Overestimated $\mathbf{z}$, $\hat{\lambda} = 0.3$, $\hat{\beta} = 0.8$. The average RMSE across different values of actual λ and β on the Boston dataset. From left to right: *MLSG*, *Lasso*, *Ridge*, *OLS*.

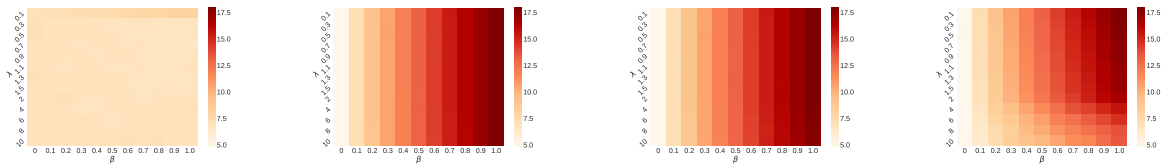

Figure C.6: Underestimated $\mathbf{z}$, $\hat{\lambda} = 0.3$, $\hat{\beta} = 0.8$. The average RMSE across different values of actual λ and β on the Boston dataset. From left to right: *MLSG*, *Lasso*, *Ridge*, *OLS*.

C.7 Supplementary Results for The PDF dataset

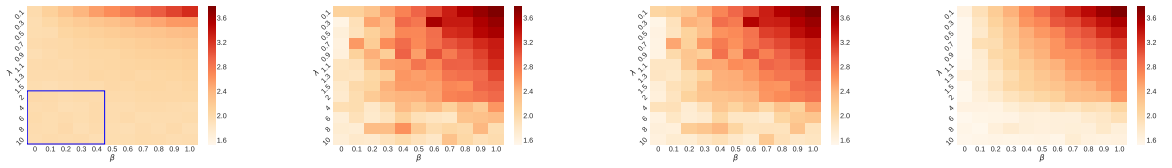

Figure C.7: Overestimated $\mathbf{z}$, $\hat{\lambda} = 1.5$, $\hat{\beta} = 0.5$. The average RMSE across different values of actual λ and β on PDF dataset. From left to right: *MLSG*, *Lasso*, *Ridge*, *OLS*.