Efficient and Insidious Adversaries in Deep Reinforcement Learning

A THESIS PRESENTED
BY
STEPHEN CASPER
TO
THE DEPARTMENT OF STATISTICS

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
BACHELOR OF ARTS (HONORS)
IN THE SUBJECT OF
STATISTICS

HARVARD UNIVERSITY
CAMBRIDGE, MASSACHUSETTS

MAY 2021

Thesis advisor: Gabriel Kreiman and Lucas Janson Stephen Casper

Efficient and Insidious Adversaries in Deep Reinforcement Learning

Abstract

Deep reinforcement learning agents can learn to solve a wide variety of tasks, but they are vulnerable to adversarial threats. Previous work has shown that one agent can develop an adversarial policy against another by training with the objective of making it fail. However, the extent to which these adversaries pose threats and opportunities is not well understood. To explore more sophisticated attacks from adversarial policies, this thesis introduces methods for training *efficient* adversaries which require few or no queries to the victim and *insidious* adversaries which are difficult to distinguish from benign agents. These methods each hinge on imitation learning and/or inverse reinforcement learning, and they include a the introduction of a new algorithm, Black-box Adversarial Q Estimation (BAQE), for inferring an agent's Q function from non-interactive demonstrations. Though preliminary, findings show potential for effective attacks, reveal directions for continued work, and suggest a need for caution and effective defenses in the continued development of deep reinforcement learning systems.

Contents

Listing of figures

In memory of Trevor Casper.

Acknowledgments

This project has benefitted greatly from the guidance and support from co-advisors Gabriel Kreiman and Lucas Janson. The past mentorship of Xavier Boix and Daniel Filan has also been invaluable for my entrance into research. I owe an additional thanks to Macheng Shen and Chuangchuang Sun for insightful discussions, and I am grateful for computational resources provided by the Center for Human-Compatible AI.

Why learn when you can exploit an unintended regularity?

Lehman et al., 2018

1 Background

1.1 Introduction

In recent years, there has been a striking amount of progress in machine learning and artificial intelligence (AI). Difficult tasks in domains such as image recognition, natural language processing, reinforcement learning, and unsupervised learning are now routinely accomplished using modern methods, often at par-human or superhuman levels. In addition to improvements in computing power and the availability of data, this has largely been thanks to the versatility of deep[1] neural networks which are capable of learning nonlinear functions which generalize well even in difficult tasks.

[1]Here, "deep" refers to a network having many internal layers that assist in the development of hierarchical representations.

However, these deep networks often converge to curious solutions, sometimes with unexpected flaws.

1.1.1 Adversarial Threats to Deep Learning Systems

In 2013, Szegedy et al. [67] found that a variety of networks trained for image classification were vulnerable to mislabeling images whose pixels were given small perturbations that were specifically designed to fool the network yet were too small to be perceptible to humans. Szegedy et al. [67] termed these images "adversarial" examples. Whereas the typical process of training a network involves treating a dataset as fixed and performing gradient-based optimization on the network's parameters with a classification objective, an adversarial image can be generated from a benign one by treating the network's parameters as fixed and performing gradient-based optimization on the image's pixels with a misclassification objective.

Since 2013, it has been found that vulnerability to adversarial inputs is extremely common in deep networks [73]. In addition to lending insight into the delicate nature of the solutions that are learned, adversarial examples also demonstrate a concerning weakness. A malicious actor who knows a victim's internal parameters can often easily create inputs that cause predictable misbehavior despite being extremely similar to a benign input – perhaps even indistinguishable to a human.

While past research into adversaries has been valuable for understanding threats to supervised learners, the conventional context in which they have been studied is limited. The bulk of research has focused on pixel perturbations to the inputs of image classifiers and uses methods that require *white-box* access to a victim in which the network parameters are visible to the attacker. However, outside this context, smaller literatures have emerged around other types of

"adversarial" attacks against deep learning systems which notably include black-box attacks (e.g. [30, 31]), feature-level adversaries (e.g. [3, 10, 43, 49, 71]), attacks against natural language models [74], and as is the focus of this thesis, adversaries in reinforcement learning [29, 63]. However, additional work is needed to more fully understand the broad range of threats posed by adversaries and what measures can be taken to detect and defend against them.

1.1.2 Adversarial Policies in Reinforcement Learning

Although more formal details will be discussed in the following section, a simple way to understand reinforcement learning (RL) is as the process by which an *agent* can learn via experience how to act inside of an *environment* in order to optimize for the attainment of some *reward*. Deep reinforcement learning systems have the potential for numerous innovative uses including in safety-critical settings such as healthcare, transportation, and human-machine interfaces. Interestingly, despite the fact that adversaries are less thoroughly studied in RL than in supervised learning, the notion of what an "adversary" can be is greatly expanded in the RL domain. In addition to perturbations to a network's input, RL agents can have adversarial environments, adversarial reward perturbations, and as is the focus of this thesis, adversarial policies from other agents [29]. A thorough discussion of what makes a policy "adversarial" can be found in Appendix A.1.

One reason why RL is a powerful general method for problem solving is that training reinforcement learners does not necessarily require advanced domain knowledge about how the task at hand should be performed.[2] So long as a suitable reward function can be provided for a task, an RL agent can develop innovative solutions on

[2]Though using such knowledge to shape the reward function is often extremely helpful.

its own. On one hand, this ability poses risks from perverse optimization and deception [28, 41]. On the other, this often allows agents to learn powerful and creative solutions to problems. For example, this famously allowed the AlphaMu [59] system (and its predecessor AlphaZero [64]) to achieve superhuman performance in the games of Chess, Go, and Shogi without data or direct guidance from humans.

This ability for RL agents to learn innovative solutions given only a reward function has been utilized for training adversarial policies in a way that only requires black-box access to a victim agent. Several closely-related works [5, 17–19, 54, 70] have trained adversarial policies by (1) beginning with a victim whose environment includes other actors, (2) "freezing" the victim so that it acts in accordance with its policy without performing new learning updates, and (3) training an adversarial agent whose reward function is (approximately) the negative of the victim agent's. Notably, in this type of attack, the victim need not be a reinforcement learner, rather simply an agent implementing a policy whose behavior can be associated with the attainment of some reward. This approach has been effective for inducing failure in a victim. However, the literature on these attacks remains limited, and to the best of this author's knowledge, previous works have used brute-force techniques to train these adversaries absent hard limitations on query access to the victim or a requirement of avoiding detection. However, unless the victim can be inexpensively queried many times and detection methods are not in place, these approaches would be unlikely to pose substantial threats.

1.1.3 Motivation

In order to better understand threats from adversarial policies and train more robust agents in reinforcement learning, this thesis

introduces a set of techniques for training adversarial policies which are (1) sample efficient and (2) difficult to detect from observation of the attacker. Each of these methods hinges on the use of *specification learning* algorithms in which an agent either learns to copy the policy of another agent, known as *imitation learning*, and/or learns an estimate of another agent's objectives from observing their actions, known as *inverse reinforcement learning (IRL)*.

Although this thesis focuses on developing more effective methods for attacking RL systems, its purpose is to improve our understanding of the threats that reinforcement learners in multiagent environments face. Additionally, training robust models (e.g. through adversarial training) has been shown to improve performance on novel but non adversarial inputs as well [18, 54, 70]. This author hopes that this and related works will encourage the more cautious deployment of RL systems and additional research into effective training, defense, and detection methods to reduce threats.

Much is at stake with adversarial attacks in RL. For example, major changes to transportation infrastructure from autonomous cars are imminent, yet it has been found that small pixel perturbations to visual inputs from a Tesla autopilot system were able to cause the model to send a signal to swerve into oncoming traffic [40]. Other adversarial threats to autonomous vehicles are possible as well [55] such as adversarial stickers on street signs and, of course, adversarial behavior from another agent on the road.

1.1.4 Outline

The remainder of Chapter 1 provides technical background, discusses related work, and introduces the new contributions made by this thesis. Chapter 2 provides in-depth details on the threat model, environments, and algorithms used here in preparation for Chapter 3

on results. While-preliminary, these suggest a potential for effective adversaries that are sample efficient and difficult to detect. Finally Chapter 4 concludes with a discussion of findings and plans for continued work. For a brief, high-level explanation of this project written for a lay-reader who may not be familiar with machine learning, see Appendix A.3.

1.2 Technical Background

This section presents definitions and a framework for understanding Reinforcement Learning that will be used in the remainder of this thesis. It draws largely from standard convention in the literature as presented in [42, 66].

1.2.1 Markov Decision Processes and Reinforcement Learning

This thesis focuses on Reinforcement Learning as the process of learning to extract reward from a Markov Decision Process (MDP). An MDP is defined as a 6-tuple $(\mathcal{S}, \mathcal{A}, T, d_0, r, \gamma)$ where $\mathcal{S}$ is a set of states, $\mathcal{A}$ a set of actions, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ a state transition probability (mass or density) function, d_0 an initial state distribution, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ a reward function,[3] and γ a temporal discount factor.

Although reinforcement learning can be broadly understood as a process for learning in a variety of decision problems including ones that cannot be formulated as MDPs (e.g. [6, 11]), this paper will use "reinforcement learning" or "RL" to refer to the process of learning a stochastic policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ parameterized by some vector of parameters θ which specifies a distribution over the actions in $\mathcal{A}$ conditioned on some state $s \in \mathcal{S}$. The notation $\pi_\theta(s)$ will be used to

[3] In many MDPs including the RL environment used for experimentation here, r is only a function of state.

denote an action sampled from the distribution over actions specified by the policy at state s, and $\pi_\theta(a|s)$ will be used to give the probability mass or density of the policy taking action a from state s. RL algorithms represent a formalized process of learning from experience to maximize the objective

$$J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left(\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right) \tag{1.1}$$

conditional on T, d_0, for states $s_t \in \mathcal{S}$, actions $a_t \in \mathcal{A}$, and with t a time index. RL algorithms learn to optimize this objective via interaction with the MDP using a behavior policy which may or may not match π_θ. A sequence of visited states and chosen actions $(s_0, a_0, s_1, a_1, ...)$ will be referred to as a *trajectory*, and for MDPs with a terminal state s_{end}, a trajectory from s_0 to s_{end} will be referred to as an *episode*.

A class of RL tasks related to the approach in this thesis is known as *multiagent reinforcement learning (MARL)*[14] in which multiple reinforcement learners share an environment and learn simultaneously. This represents a more complex type of problem than single-agent RL due to the fact that an agent's environment contains another agent whose internal state is both unknown and changing over time yet which is part of the input for T. This could be represented as a *Partially-Observable MDP (POMDP)* which also includes a set of observations $\mathcal{O}$ which π_θ takes as input and a lossy, potentially stochastic function $o : \mathcal{S} \to \mathcal{O}$ which maps states to observations. In this setting, the presence of an external agent which is actively learning results in a lack of stationarity in the MDP that causes many common RL algorithms to lose convergence guarantees [25, 53]. Although this thesis considers RL problems in environments that have multiple agents in a colloquial sense, in all of the experiments

performed here, only one agent at a time is actively learning. Thus, this is *not* a work involving MARL. See Section 2.1 for a more detailed discussion of the threat model.

1.2.2 Policy Gradients

Policy gradient methods are a powerful approach to RL that are commonly used in modern research due to their versatility and ability to learn quickly. A policy gradient algorithm known as *Proximal Policy Optimization (PPO)* is used for training agents in this thesis and is explained in detail in section 2.3.1. Given a trajectory $(s_0, a_0, s_1, a_1, ...)$, the gradient of $J(\pi_\theta)$ from equation (1.1) as derived in [66] is given by

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{d_0,T}\left(\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_{\pi_\theta}(s_t, a_t)\right) \tag{1.2}$$

where

$$\hat{A}_{\pi_\theta}(s_t, a_t) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'}) - b_{\pi_\theta}(s_t) \tag{1.3}$$

is an estimate of what is known as the *Advantage* function, and $b_{\pi_\theta}(s_t)$ is an estimate of the total *baseline* discounted reward achieved in a trajectory from s_t onward under π_θ. RL algorithms in the policy gradient family are characterized by the use of gradient-based optimization on θ. These algorithms are known as *on-policy* because they require that the data from which they learn come from π_θ interacting with the MDP as opposed to some other policy. This contrasts with *off-policy* algorithms which do not require this and are not used for training agents in this thesis. It is worthy of note that in practice, estimating $\nabla_\theta J(\pi_\theta)$ from equation 1.2 from experience is a high-variance process, and optimization steps are typically taken with batch sizes in the thousands.

1.2.3 Actor Critic Methods, Temporal Difference Learning, and Bootstrapping

Given the basic form of a policy gradient from equations (1.2) and (1.3), it remains to be determined how b_{π_θ} should be estimated. The simplest way to do so, known as a *Monte Carlo* method, is to calculate the empirical discounted sum of rewards for a rollout of n steps (or until s_{end}) from s_t as $\sum_{t'=t}^{t+n} \gamma^{t'-t} r(s_{t'}, a_{t'})$. However, an alternative method for estimating b_{π_θ} that incorporates past experience rather than a single trajectory at a time involves training a separate model specifically for its estimation. This is known as an *actor-critic* approach where the *actor* refers to π_θ and *critic* refers to the estimator of $b_{\pi_\theta}(s_t)$. This estimator is known as the *Value* or *State Value* function $V_{\pi_\theta}(s_t)$ and is defined as

$$V_{\pi_\theta}(s) = \mathbb{E}_{\pi_\theta,} \left(\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi_\theta(s_t)) \right). \tag{1.4}$$

Importantly, the optimal value function $V^*_{\pi_\theta}$ has a temporal recursive nature which satisfies

$$V^*_{\pi_\theta}(s) = \mathbb{E}_{\pi_\theta} \left(r(s, \pi_\theta(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s, \pi_\theta(s), s') V^*_{\pi_\theta}(s') \right) \tag{1.5}$$

known as a *Bellman equation.*

A function closely related to the value function is the *state-action-value* function denoted by Q:

$$Q_{\pi_\theta}(s, a) = r(s, a) + \gamma \mathbb{E}_{\pi_\theta} \left(\sum_{s' \in \mathcal{S}} T(s, \pi_\theta(s), s') V_{\pi_\theta}(s')) \right). \tag{1.6}$$

And given Q, the *advantage* function from equation (1.3) can be

expressed as:

$$A_{\pi_\theta}(s,a) = Q_{\pi_\theta}(s,a) - V_{\pi_\theta}(s). \tag{1.7}$$

Due to the temporal recursive nature of the value function, actor-critic methods which involve the estimation of the closely-related functions V, Q, or A are known as *temporal difference* methods. In practice, these methods also involve *bootstrapping* in which the current estimate of one of these temporal-recursive functions is used for its own update based on a Bellman equation. Despite this fact, however, these methods each have convergence guarantees in well-behaved MDPs [66].

1.2.4 Specification Learning

One reason why reinforcement learning is a powerful problem solving method is that a practitioner need only specify a suitable reward function for an agent to learn how to accomplish a task. Incorporating domain knowledge into a strategic reward specification, known as *reward shaping*, can greatly improve learning in many cases. However, RL agents ultimately learn their own solutions to the task at hand, and advanced domain knowledge is often not required from the user. As discussed earlier, RL can often learn creative [41] and even superhuman policies [59, 64] due to this capability.

While the ability of RL agents to learn effective policies given only a reward function is highly useful, it poses two challenges: (1) How can we train an RL agent when we know what behavior we want it to exhibit but do not know what reward function best encourages it? And (2) how can we avoid giving agents reward functions which cause them to learn unexpected and potentially dangerous solutions to problems? Concerning (2) in particular, [41] compiles dozens of examples of systems learning undesired solutions to problems when given naive reward specifications, and [7, 12, 57, 68] each argue that

dangers from perverse optimization are a paramount challenge for safety in the continued development of advanced AI systems.

The field of *specification learning* is principally motivated by these two problems. Specification learning refers to learning behavior, known as *Imitation Learning*, and/or learning a reward function estimate, known as *Inverse Reinforcement Learning (IRL)*, from an expert. Notably, there are numerous ways in which information can be extracted from the expert for specification learning [34] including but not limited to demonstrations [50], comparisons [12, 72], corrections [2], cooperation [21], language [44, 47, 65], interpreting the reward specification as an observation [22], auxiliary rewards [20], and analysis of d_0 [62].

Algorithms for imitation and IRL are central to this thesis. One straightforward method used here for imitation focuses on directly imitating an expert by training it to output a policy with a small Kullback-Leibler (KL) divergence[4] from the expert's. Another method known as *generative adversarial reinforcement learning (GAIL)* [26] is modified for both imitation learning and estimating a victim's value function. Additionally, this thesis also introduces a new algorithm, *Black-box Adversarial Q Estimation (BAQE)* meant to both learn an imitator and an estimate of a victim's Q function from non-interactive demonstrations. GAIL and BAQE are closely related and both involve training an imitator from demonstrations in order to fool a separate *discriminator* which is trained to distinguish between the imitator and expert's actions. Full details on these algorithms and how they are adapted for this thesis are in Sections 2.3.3 and 2.3.4.

[4] $KL(\pi, \pi'|s) = \sum_{a \in \mathcal{A}} \pi(a|s) \log\left(\frac{\pi(a|s)}{\pi'(a|s)}\right)$

1.3 Brief Overview of Methods

This thesis introduces three methods for developing adversarial policies that pose more potent threats. Specification learning is key to all three. Details for each will be expounded upon and presented in the following chapters.

Transfer from an Imitative Proxy (section 3.1): While the ability to interactively train against an intended victim may be limited, benign and non-interactive data from the victim may be more readily available. In this type of attack, (1) an imitator of the victim is trained, (2) an adversary is trained against the imitator, and (3) optionally, the adversary is fine-tuned against the actual victim for a small number of timesteps. This allows for adversarial behavior to be learned for a similar agent before being transferred to the intended victim, thus allowing for the training of an adversarial policy with little or no query access to the victim.

Observation Augmentation via Victim Modeling (section 3.2): The standard approach to training an adversarial policy is a black-box process which need not require access to a victim's internal state. However, having access to a proxy for it can be valuable. In particular, an estimate of a victim's Value or Q function can allow for rapid learning of the adversary's Value function because, up to normalization, $V_{\text{adversary}}(s) = -V_{\text{victim}}(s) = -\mathbb{E}\left(Q_{\text{victim}}(s, \pi_\theta(s))\right)$. In these experiments, (1) an imitator and Value/Q function estimator for the expert are jointly trained, and (2) during training, the adversary's observations are augmented with the imitator's action and Value/Q model outputs.

Insidious Attacks via Expert Imitation (section 3.3): While adversaries can be highly effective at thwarting an individual victim, they may develop easily-detectable policies or policies that do not transfer to successful behavior in an environment without the victim.

To test how well this can be avoided, *insidious* adversaries are trained with a composite objective of thwarting the target victim and imitating a separate non-adversarial expert who is adept at performing the task in a non-adversarial way. Victim models as discussed above are used in several of these experiments as well.

1.4 Related Work

RL agents can be vulnerable to a diverse set of adversarial threats including input perturbations, rewards perturbations, environments, and agents. Ilahi et al. [29] offer a broad survey of threats and defenses. For adversarial policies in particular, the training process is typically simple. The basic approach is to fix a victim in a multiagent environment so that it is not actively learning and to then train an adversarial agent with the goal of making the victim fail. This type of adversarial behavior was observed unintentionally by both [4] and [56] who found that when training multiple agents in competitive RL environments, it was key to rotate players in a round-robin fashion to avoid agents overfitting against a particular opponent. Notably, in this setting, there are multiple "agents" in a colloquial sense, but the problem is not a multiagent reinforcement learning one because no two agents learn at the same time.

Simple Adversarial Policies: Overall, this simple "overfit-to-the-victim" approach has been the basic strategy of a number of recent works investigating adversarial policies [5, 17–19, 54, 70]. These works have made valuable contributions for understanding threats and opportunities from adversarial policies. A number of these demonstrate that training a victim against an adversary, known as *adversarial training* is a reliable way to promote robustness [18, 54, 70]. Aside from adversarial training, [19] also demonstrated reasonably effective detection of adversarial policies and

the ability to defend against them by obfuscating observations from an attacker. Both of these findings from [19] relate to an additional finding that when an adversary overfits to its victim, it may learn to use specific actions that thwart the victim but do not make for a reasonable policy in general. This suggests a need for investigating threat models in which an adversary is trained not only to beat a victim, but to do so in a way that is difficult to detect from observations alone.

Adversaries Outside the Environment: Although this thesis focuses on adversarial attacks in environments which have multiple agents embedded inside of them, a number of previous works have trained adversarial policies which act *outside* of the environment. In this setting, the adversary's actions involve perturbing the victim's observations or the MDP's transition function T. These allow for any type of environment to be used, whether naturally multiagent or not. The primary motivation in the literature for training these has been to improve the robustness of the victim via training on the adversarially-perturbed environment. The authors of [54], who introduced this paradigm refer to this as Robust Adversarial Reinforcement Learning (RARL). Since, several works have expanded on this approach for developing robust policies, notably including ones which demonstrate robust transfer from simulation to the real world [1], the use of an ensemble of adversaries [70], and methods for generating certified robustness guarantees [45].

Adversarial Transfer from Proxy: In the more well-studied domain of supervised learning, it has been found that adversarial examples often transfer from one victim to another, even if the other was independently trained [32, 46, 51, 69]. Additionally, [27] found that adversarial attacks in RL could transfer between independently trained agents as well, although unlike this work, they only tested adversarial perturbations to observations as opposed to policies. A

work which took a related approach to one of the key experiments in this thesis is [5] who trained an imitator to serve as a proxy for a victim, trained an adversarial state-perturbing policy against the proxy, and tested how well that policy transferred to the original victim. While successful, this remains limited because the adversary was outside the environment, and the experiments were restricted to an elementary "CartPole" environment [9]. However, this approach using transfer from an imitative proxy shows promise given high performance specification-learning algorithms such as GAIL [26].

Victim Modeling: Another key related work is [38]. Its authors only investigated adversarial observation perturbations as opposed to policies. However, they demonstrated the effective use of a victim's internal state for scheduling maximally-effective adversarial perturbations. Under their threat model, the insertion of an perturbation was assumed to be costly, and to to overcome this, they found that inserting them when the victim's value function was above a certain threshold was helpful for maximizing damage while on a budget. This success suggests potential benefits from using a learned victim model for generating more effective and insidious adversarial policies. Similar to the solution used here, He et al., 2016 [23] experiments with the use of a learned victim model's action and Q estimates. Though conceptually similar to the approach used here, their focus was not on efficiency or insidiousness, and their work predates the more modern imitation and IRL algorithms used here. Instead, their experiments were limited to simple environments and a Q-learning[5] approach for training a victim-model off-policy.

[5]Though not discussed in Section 1.2, Q learning methods are a family of RL algorithms that contrast with policy gradients. See [66].

1.5 Contributions

Despite this notable progress from previous work, there exist several gaps in the research which this work begins to explore.

First, to the best of this author's knowledge, there are no previous works which develop adversarial policies with sample efficiency as a central goal. Previous works have used simple and "brute force" approaches in which an adversary is trained, often for many millions of timesteps (e.g. [19]) until it has developed a sufficiently detrimental policy. However, inefficient attacks could be easily defended against if either the victim were not in-silico (such as an autonomous vehicle) or query access to the it were costly/limited. This thesis makes the development of attacks with as few queries to the victim as possible a primary focus. In doing so, this thesis utilizes a useful variant of GAIL and introduces BAQE, a novel algorithm for inferring an expert's Q function from non-interactive demonstrations. This is a considerably more restricted set of permissions than required by related IRL algorithms.

Second, while the use of a victim model has proven helpful for attacking reinforcement learners (e.g. [38]), little work has focused on black-box methods [23] which is key to more germane threat models because in real systems, black-boxifying models is simple, typically done by default, and entirely thwarts white-box attack algorithms. Methods are introduced in this thesis to circumvent this problem by exploiting learned victim models and/or independently-trained agents as opposed to using direct access to the victim itself. Though related to [23], this work uses more state of the art IRL algorithms, including BAQE, and experiments with a more sophisticated RL environment. In the experiments here, these black-box attacks are also compared against white-box baselines, and another incidental but novel contribution of this work is investigating these white-box attacks.

Third and finally, findings such as those from [19] suggest that even if an adversarial policy cannot be precluded or defended against, it may still be very easy to detect. This thesis makes training insidious adversarial policies an explicit focus which to the best of this author's knowledge, no prior works to date have made a goal.

Experiments are ongoing, but this thesis marks a milestone toward better understanding threats and opportunities from more potent adversarial attacks in RL.

2 Approach

This chapter presents the framework and methods used for exploring sample-efficient and insidious adversarial attacks. This is in preparation for the following chapter which covers how these methods are used and presents results. Additional minor details about training, hyperparameters, and network architectures are reserved for Appendix A.2.

2.1 Threat Model

Like adversarial examples for supervised learning models, observation-perturbation-based adversaries for reinforcement learners (e.g. [27, 38]) are most often studied under a white-box threat model in which the adversary has access to the victim's internal parameters.

Threat Model	Permissions	Restrictions
Related works e.g. [18, 19, 54, 70]	-Victim reward function -Cheap query access	-Black-box access
Efficient attacks	-Victim reward function -Benign victim demonstrations	-Black-box access -Expensive/no query access to victim or similar agent -Demonstrations non-interactive
Insidious attacks	-Victim reward function -Cheap query access -Grey-box access to benign expert	-Black-box access -Detector of form $D(a\|s)$

Table 2.1.1: Key permissions and restrictions of threat models for previous works and the two paradigms in this thesis.

This allows for gradients to be calculated via propagation through the victim's policy network and used for developing the adversarial input. However, these attacks can be easily precluded by black-boxifying the victim. In a departure from this threat model, related works such as [18, 19, 54, 70] have trained adversaries under the more restricted setting in which access to the victim is limited to that of a black-box. This approach takes advantage of the ability for reinforcement learning to learn via experience to achieve a reward function, even absent access to internal information about their victim.

In this thesis, three methods are presented for training more potent adversaries. First and second, transfer from an imitative proxy and observation augmentation via victim modeling are used for training efficient adversaries. Third, insidious attacks are developed via expert imitation. Table 2.1.1 outlines the key permissions and restrictions for the threat models used here and in related work.

Efficiency and insidiousness will often be necessary for developing attacks against real world systems because limiting query access and monitoring for threats are very simple defensive measures. However, they are by no means sufficient for genuine threats as other defensive measures could be in place such as adversarial training, limiting demonstrations, observation preprocessing, action postprocessing,

human overrides, or others. Nonetheless, this work will help us to better understand threats from adversaries and opportunities from adversarial training in more limited and realistic contexts than previous works have.

Efficient Attacks: Leveraging RL to generate adversarial policies with only black-box access to a victim is a powerful technique. However, works such as [18, 19, 54, 70] have assumed that the victim can be freely, cheaply queried interactively for many timesteps to train adversaries. The methods introduced here for efficient attacks aim to train them under a more restrictive threat model in which query access to the victim is limited/expensive. However, for both of these methods, it is assumed that while the victim can't be queried interactively, the attacker has access to demonstrations in which the victim is performing in a benign (non-adversarial) setting. Furthermore, it is assumed that (1) these demonstrations are not interactive, i.e. that an attacker cannot simulate what *would have happened* if the expert had taken some action a'_t instead of a_t and (2) that no similarly-trained proxies are available with cheap query access. To understand why these are realistic assumptions, consider an autonomous vehicle. While query access to any physical model will be inherently expensive and limited, copious amounts of data from benign driving settings may be available, though not likely in the form of an interactive simulation.

Insidious Attacks: For training insidious attacks via expert imitation, the adversary is not assumed to have limitations on query access to a victim. In fact the threat model is very similar to that of related works except that in these experiments, the adversary has additional access to the outputs of an independently trained, non-adversarial agent which the adversary learns to imitate at the same time as it learns to beat the victim. Rather than generating attacks under a limited threat model, this method simply aims to

make an adversary more resemble a benign agent when detection methods may be in place.

In supervised learning, adversaries are typically made to be insidious in the sense that they only differ from a benign input by a small-norm perturbation. However, the notion of detectability is more complicated for adversarial RL. In RL, any observer with access to the rewards for a victim agent can determine exactly how well it is performing. But there are two ways in which an adversary being difficult to detect via *observations* alone would make for a more dangerous threat. First, in deployment, an adversary may need to do more than simply interact with the victim. Consider one autonomous vehicle with a policy adversarial to another. Here, a realistic adversarial agent would need to perform reasonably well on roads with other agents or none at all. Imitating a benign agent would aid in this and help to avoid detection before interacting with the victim. The second reason why avoiding observation-based detection would make for more potent threats is by making the cause of a failure more difficult to detect. For example, this could lead to more uncertainty about whether a victim's failure is due to an adversarial agent or other minutiae of the particular setting in which it finds itself.

2.2 Environment and Agents

Experiments are conducted using the The Google Research Gfootball[1] [39] environment which was selected for being a free, accessible, advanced, and naturally-multiagent setting in which to study adversarial policies. Each agent in the environment controls a team of football (soccer) players in play against another with the objective of scoring goals and preventing the opponent from scoring them. Each team has 11 teammates in play, but all are controlled by a single

[1] https://github.com/google-research/football

Figure 2.2.1: A visual rendering from a Gfootball game. Note that this was not the same type of rendering used for observations. From [39].

agent who actively controls one teammate at a time and is able to toggle between them to switch the active player. Inactive players perform simple, rule-based actions. Each game lasts for a total of 3,000 timesteps or 5 minute of play at 10 frames per second.

The Gfootball software optionally allows for observations to be used which consist of a 115-length vector of extracted features. However, here visual observations were extracted which were $72 \times 96 \times 4$ pixels. The four channels encoded the left team positions, right team positions, ball position, and active player position. Due to the fact that observing a single frame does not allow for the inference of useful information such as velocities, observations were stacked from the previous four timesteps to yield inputs that were $72 \times 96 \times 16$ pixels. The action space was discrete with 19 distinct actions.[2] Although the objective of the game is to score more goals than the opponent, agents

[2]These were idle, left, top-left, top, top-right, right, bottom-right, bottom, bottom-left, long-pass, high-pass, short-pass, shot, sprint, release-direction, release-sprint, sliding, dribble, and release-dribble.

were given a shaped reward to improve the efficiency of learning as done in [39]. In addition to rewards of $+1$ and -1 when goals were scored by or against an agent, a reward of 0.1 was given for successfully advancing the ball an additional one-tenth of the way down the field while maintaining possession. As a result, the rewards were not zero-sum, but still strongly linked to the zero-sum measure of the difference between scores.

The agents processed the $72 \times 96 \times 16$ inputs via a deep convolutional residual network (ResNet), a type of architecture introduced by [24]. The network consisted of four modules, each with two residual blocks followed by a flattening layer and two densely connected layers. Batch normalization was used between convolutional layers. These layers led to a 256 dimensional output from which the value and policy were computed each with their own dense layer. Because the policy was discrete, a softmax layer was used to compute the distributions over outputs. All activations were ReLU, and the architecture differed for experiments investigating observation augmentation via victim modeling. Full architectural details are in Appendix A.2.

2.3 Algorithms

This section outlines the RL, imitation, and IRL algorithms used in the experiments that will be the focus of the next chapter. This includes the introduction of BAQE, a new IRL algorithm used for inferring a victim's Q function from non-interactive demonstrations.

2.3.1 Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a reinforcement learning algorithm in the policy gradient family. Since its introduction by [61] in 2017, it has become common due to its simplicity, tendency to

require little to no hyperparameter tuning, and excellent empirical performance. Recall the policy gradient formula from equations (1.2) and (1.3) in Section 1.2.2:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{d_0}\left(\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_{\pi_\theta}(s_t, a_t)\right)$$

$$\hat{A}_{\pi_\theta}(s_t, a_t) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'}) - b_{\pi_\theta}(s_t).$$

This calculates the gradient with respect to the parameters of π_θ. The simplest approach to optimization would be to take gradient steps in this parameter space, however this tends to be an imprecise way to tune a policy because the size of an optimization step in the parameter space does not reliably predict the degree to which the policy is changed. Another challenge with this approach is that in practice, taking multiple optimization steps for an on-policy algorithm based on the same trajectory, while appealing, tends to cause sporadic and too-large updates. These problems have motivated a family of algorithms known as *natural policy gradients* [36] which take optimization steps of a size not determined by a learning rate on the parameters but rather by limits on the change imposed on the policy.

One solution to this in the natural policy gradient family is *Trust Region Policy Optimization (TRPO)* [60] which was a predecessor to PPO. This algorithm introduces a surrogate objective subject to a constraint on the KL divergence between the old, pre-update policy and the new, post-update policy:

$$\begin{array}{ll} \max\limits_{\theta} & \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} A_t\right] \\ \text{subject to} & \hat{\mathbb{E}}_t\left[\text{KL}\left(\pi_{\theta_{\text{old}}}, \pi_\theta|s_t\right)\right] \leq \delta \end{array} \tag{2.1}$$

where $KL(\pi, \pi'|s) = \sum_{a\in\mathcal{A}} \pi(a|s) \log\left(\frac{\pi(a|s)}{\pi'(a|s)}\right)$ denotes the

Kullback-Leibler (KL) divergence between two distributions at a state s, and δ is a limit on the allowable KL divergence which defines a region of "trusted" policies at a given step – hence the name *Trust Region* Policy Optimization. TRPO tends to perform well, however it is relatively complicated to implement in practice. An additional challenge is that the theoretical basis for TRPO justifies a penalty rather than a constraint on the KL divergence, but penalty-based TRPO methods are empirically difficult to tune.

Proximal Policy Optimization was developed as solution to these problems. It comes in two variants. The first known as PPO-penalty introduces an adaptive coefficient to a penalty on the KL divergence between the old and new policy. The surrogate objective is given by

$$\max_{\theta} \ \hat{\mathbb{E}}_t \left[\frac{\pi_\theta \left(a_t \mid s_t \right)}{\pi_{\theta_{\text{old}}} \left(a_t \mid s_t \right)} A_t - \beta \, \text{KL} \left(\pi_{\theta_{\text{old}}} , \pi_\theta | s_t \right) \right] \tag{2.2}$$

where after each step, β is up- or down-weighted for the next step based on whether the change in the KL divergence is less than or greater than a target level d which may optionally undergo decay during training.

However, the adaptive penalty version of PPO often has inferior performance in practice to a second and more commonly used version of PPO known as PPO-clip [61] which is used to train agents in this thesis. Rather than adding a penalty, this version of PPO simply clips the surrogate objective in order to eliminate the incentive for moving the new policy too far from the old one:

$$\max_{\theta} \ \hat{\mathbb{E}}_t \left[\min \left(\frac{\pi_\theta \left(a_t \mid s_t \right)}{\pi_{\theta_{\text{old}}} \left(a_t \mid s_t \right)} A_t, \text{clip} \left(\frac{\pi_\theta \left(a_t \mid s_t \right)}{\pi_{\theta_{\text{old}}} \left(a_t \mid s_t \right)}, 1 - \epsilon, 1 + \epsilon \right) A_t \right) \right] \tag{2.3}$$

In this case, regardless of whether the advantage function estimate is positive or negative, clipping the ratio in the second argument to the min function from both above and below constrains the size of the

step in policy space. Using this proxy objective, the version of PPO used here is given in Algorithm 1. In experiments here, an entropy regularization term was also added to the reward function to encourage exploration as done in [39].

Algorithm 1 Actor-Critic Proximal Policy Optimization - Clip [61]

Input: Untrained actor $\pi_{\theta_{\text{old}}}$ and critic $A_{\phi_{\text{old}}}$.
Output: Trained actor and critic.
for iteration in $1, 2, \ldots, n$ **do**
    Run actor $\pi_{\theta_{\text{old}}}$ for T timesteps to collect state, action, and reward data.
    Obtain advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$ using critic $A_{\phi_{\text{old}}}(s, a)$.
    Train actor π_θ from θ_{old} to maximize surrogate objective (2.3) over K epochs.
    Train critic A_ϕ from ϕ_{old} over K' epochs via bootstrapping.
    $(\theta_{\text{old}}, \phi_{\text{old}}) \leftarrow (\theta, \phi)$
end for

2.3.2 Behavioral Cloning

Learning to imitate from demonstrations is a simple specification learning technique used to create insidious adversaries in this thesis. The most straightforward approach to imitation is *behavioral cloning* which refers to training an imitator to copy an expert's actions. A simple method for this would be train a model to classify or regress on an expert's actions given a dataset of observation, action pairs as if it were a supervised learning problem.

For training insidious victims, a method for behavioral cloning that is only slightly more complicated was used here. Here, insidious adversaries were created by training an agent who is rewarded jointly for mimicking a benign agent and thwarting its intended victim. Hence, this is an approach based on reinforcement learning rather than supervised learning. Unlike the intended victim, the expert used

here was not assumed to be a black box, so under this threat model, a richer reward signal was extracted from expert's output distribution over discrete actions. Here, the environment-based rewards for playing against the adversary along with the imitation-based reward based from the KL divergence between the expert and imitator were each normalized to have mean zero and variance one before being combined with a weight of $\beta \in [0, 1]$ on the environment reward and $(1 - \beta)$ on the imitation reward to calculate the final joint reward. For these experiments, β was set to 0.75. This process is outlined in Algorithm 2.

Algorithm 2 Insidious Adversary Training

1: **Input:** Trained expert π_{ex} and untrained insidious adversary π_{old}. Victim embedded in the MDP.
2: **Output:** Trained insidious adversary.
3: **for** iteration in $1, 2, \ldots, n$ **do**
4: Run π_{old} for T timesteps to collect state, action, action distribution, and reward data $s_1, \ldots, s_T$, $a_1, \ldots, a_T$, $\pi_{\text{old}}(s_1), \ldots, \pi_{\text{old}}(s_T)$, and $r_1, \ldots, r_T$.
5: Collect expert action distributions $\pi_{\text{ex}}(s_1), \ldots, \pi_{\text{ex}}(s_T)$.
6: Obtain imitation rewards $\hat{r}_1, \ldots, \hat{r}_T$ in the form of $\hat{r}_i = KL(\pi_{\text{ex}}(s_i), \pi_{\text{old}}(s_i))$.
7: Obtain training rewards by normalizing and combining the adversarial and imitative rewards with weights of $\beta \in [0, 1]$ and $(1-\beta)$.
8: Train imitator π_θ from θ_{old} using any actor-critic method on training rewards.
9: $\theta_{\text{old}} \leftarrow \theta$
10: **end for**

2.3.3 Generative Adversarial Imitation Learning (GAIL)

A powerful strategy often used for training machine learning systems including imitation learners is a *generative adversarial* scheme. This approach hinges on the use of a *generator* which is trained to produce

some desired output by pitting it against a *discriminator*. The generator is given the objective of fooling the discriminator while the discriminator is given the objective of correctly distinguishing generator outputs from natural references. Both are trained simultaneously. For example, image generators have been trained to produce fake but photorealistic images by fooling a discriminator which is trained to distinguish generator outputs from natural images (e.g. [8]).

Generative adversarial imitation learning (GAIL) [26] is such an approach to imitation learning. The generator/imitator is a reinforcement learner who is trained to mimic an expert, and the discriminator is trained to distinguish the generator's trajectories from the expert's. A version of GAIL in which the discriminator takes state/action pairs as input is formally presented in Algorithm 3.

Algorithm 3 Generative Adversarial Imitation Learning (GAIL) [61]

Input: Trained expert π_{ex}, untrained imitator π_{old}, and untrained discriminator $D_{\phi_{\text{old}}}$.
Output: Trained imitator and discriminator.
for iteration in $1, 2, \ldots, n$ **do**
    Run expert π_{ex} for T timesteps to collect state/action data $s_1, \ldots, s_T$, $a_1, \ldots, a_T$.
    Collect imitation actions $\hat{a}_1, \ldots, \hat{a}_T$ from $s_1, \ldots s_T$ using imitator $\pi_{\theta_{\text{old}}}$.
    Obtain imitation rewards $r_1, \ldots, r_T$ from Discriminator $D_{\phi_{\text{old}}}$ in the form of $r_i = \text{logit}(D_{\phi_{\text{old}}}(s_i, \hat{a}_i))$
    Train imitator π_θ from θ_{old} on imitation rewards.
    Train discriminator D_ϕ from ϕ_{old} to identify expert from imitator actions via binary logistic regression.
    $(\theta_{\text{old}}, \phi_{\text{old}}) \leftarrow (\theta, \phi)$
end for

The first approach here for training efficient adversaries involves pretraining against an imitative proxy, and the second involves

augmenting the attacker's observations with action and value estimates for the victim. However, by itself, a simple actor-critic GAIL imitator will tend to perform poorly for both of these tasks. First, under this threat model, only demonstrations are available, so when a GAIL imitator inevitably makes errors and drifts off the state distribution for the policy it is trying to imitate, it will be less useful as a proxy victim. Second, the critic from an actor-critic GAIL imitator will not make for a useful Value function estimate because the imitation objective will not learn to reflect how the expert values a given state – only how imitatable the expert is at that state.

As a solution to both of these problems, a variation on GAIL is used in which the imitator is trained jointly to imitate via rewards based on the GAIL discriminator and to accomplish the expert's objective via the expert's rewards. In this variant of GAIL which will be referred to here as *GAIL-RL*, the imitation-based rewards and the expert's environment-based rewards for GAIL were normalized and then combined with weights $\beta \in [0, 1]$ and $1 - \beta$ respectively to create rewards for training. Here, $\beta = 0.75$ was used. The process is shown in Algorithm 4 which is very similar to Algorithm 3. Although the expert's rewards are not necessarily consistent with the rewards that the victim's policy would achieve, as training progresses, the expert's rewards become progressively closer to being on-policy. As a result of incorporating the expert's rewards, this method encourages the imitator to both learn a policy that keeps the agent in a reasonable state distribution and to learn a value function which reflects the expert's. Afterward, the actor was used for imitation, and the critic for value function estimation.

When training GAIL imitators for experiments here, a crucial strategy to avoid the imitators simply learning to output the most common actions taken by the experts was to train the discriminator

Algorithm 4 GAIL-RL

Input: Trained expert π_{ex}, untrained imitator π_{old}, and untrained discriminator $D_{\phi_{\text{old}}}$.
Output: Trained imitator and discriminator.
for iteration in $1, 2, \ldots, n$ **do**
    Run expert π_{ex} for T timesteps to collect state, action, and reward data $s_1, \ldots, s_T$, $a_1, \ldots, a_T$, and $r_1, \ldots, r_T$.
    Collect imitation actions $\hat{a}_1, \ldots, \hat{a}_T$ from $s_1, \ldots s_T$ using imitator $\pi_{\theta_{\text{old}}}$.
    Obtain imitation rewards $r_1, \ldots, r_T$ from Discriminator $D_{\phi_{\text{old}}}$ in the form of $r_i = \text{logit}(D_{\phi_{\text{old}}}(s_i, \hat{a}_i))$
    Obtain training rewards by normalizing and combining the imitation-based expert rewards with weights of $\beta \in [0, 1]$ and $(1 - \beta)$.
    Train imitator π_θ from θ_{old} on training rewards.
    Train discriminator D_ϕ from ϕ_{old} to identify expert from imitator actions via binary logistic regression.
    $(\theta_{\text{old}}, \phi_{\text{old}}) \leftarrow (\theta, \phi)$
end for

only on examples which it predicted incorrectly. Additionally, a small amount of weight decay was also used to improve the performance of the discriminator.

2.3.4 Black-box Adversarial Q Estimation (BAQE)

While combining rewards based on imitation and the expert's interaction with the environment in GAIL-RL alleviates the aforementioned problem of learning a fully irrelevant Value function, the resulting value function estimate will still fail to fully represent the victim's actual Value function. As a more principled solution to the problem of learning a faithful victim model, one can use a similar generative adversarial strategy but learn the victim model inside the discriminator rather than the imitator. In GAIL, the discriminator $D(s, a)$ need only learn to distinguish between expert and imitator

actions given a state and can take on any form that allows it to do so. Contrastingly, [15] propose using a discriminator for performing binary logistic regression which takes the form

$$D_\phi(\tau) = \frac{\exp\{f_\phi(\tau)\}}{\exp\{f_\phi(\tau)\} + \pi_\theta(\tau)}$$

and learns to distinguish expert from imitator trajectories, represented by τ. However, a considerably simpler but similar approach would be to use state/action pairs rather than entire trajectories:

$$D_\phi(s,a) = \frac{\exp\{f_\phi(s,a)\}}{\exp\{f_\phi(s,a)\} + \pi_\theta(s,a)}.$$

For a given expert π_{ex}, the optimal f will be $f^*(s,a) = \log \pi_{\text{ex}}(a|s) = A_{\pi_{\text{ex}},T}(s,a)$ where $A_{\pi_{\text{ex}},T}$ gives the optimal advantage function induced by T, π_{ex}, and some reward function r^* for which π_{ex} is optimal [16]. Unfortunately, simply learning f directly would be of very little use to an attacker because the Advantage function does not convey any information about the victim that isn't redundant with its policy, and an estimate of the policy is already used in the victim modeling attacks here. However, the Advantage function can be expressed in terms of the reward and Value function as such

$$\begin{aligned} A_{\pi_\theta}(s,a) &= Q_{\pi_\theta}(s,a) - V_{\pi_\theta}(s) \\ &= r(s,a) + \gamma \mathbb{E}_{\pi_\theta}\left[V_{\pi_\theta}(s')\right] - V_{\pi_\theta}(s), \end{aligned}$$

and the victim's reward function is known under this threat model. This motivates using a new approach for learning f as a function of the state, action, and successor state

$$f(s,a,s') = r(s,a) + \gamma h(s') - h(s). \tag{2.4}$$

Such that at optimality, $h(s) = V_{\pi_\theta,T}(s)$. In fact, another algorithm known as *Adversarial Inverse Reinforcement Learning (AIRL)* uses a discriminator based on a similar function in order to learn estimates of the reward and value functions [16].

Unfortunately, although in this case the rewards are known, a discriminator of this form is still not a viable approach under the threat model used here. To successfully learn to discriminate between an expert and imitator with a function of the same form as equation (2.4), one needs to have different s, s' pairs from the expert and imitator, especially because in many reinforcement learning MDPs, the reward $r(s)$ is only a function of state. However, here it is only assumed that demonstrations from the victim are available in the form of non-interactive trajectories, so using different s, s' pairs to learn to distinguish between the expert and imitator is not assumed to be possible. The expert and the imitator must then only be distinguished based on the actions they take given states visited by the expert. This motivates learning f from $r(s)$ as a state-only function, two states, and two actions as such:

$$f(s, a, s', a') = r(s) + \gamma h(s', a') - h(s, a).$$

An f of this form would then be tractable under this threat model because h is now a function of state and action. And given that $\mathbb{E}_{\pi_\theta}\left[Q_{\pi_\theta}(s, \pi_\theta(s)\right] = V_{\pi_\theta}(s)$, at optimality $h(s, a) = Q_{\pi_{\text{ex}}}(s, a)$. Thus, for learning an estimate of an expert's Q function when $r(s, a)$ is known and only non-interactive demonstrations are available, this thesis proposes a generative adversarial method in which an imitator and discriminator are trained against each other much like in GAIL, but where the discriminator has the form

$$D(s, a, s', a') = \frac{\exp\{f(s, a, s', a')\}}{\exp\{f(s, a, s', a')\} + \pi_\theta(a|s)} \tag{2.5}$$

$$f(s, a, s', a') = r(s) + \gamma h(s, a) - h(s', a'). \tag{2.6}$$

So that at optimality, $h(s, a) = Q_{\pi_{\text{ex}}}(s, a)$, allowing h to be used as an estimate of the victim's Q function. This algorithm is termed here as *Black-box Adversarial Q Estimation (BAQE)*. Though similar to AIRL [16], BAQE can operate under the considerably more-restricted setting in which demonstrations are non-interactive. BAQE is outlined in full in Algorithm 5 which differs only from GAIL-RL in algorithm 4 by a single step.

Algorithm 5 BAQE-RL

1: **Input:** Trained expert π_{ex}, untrained imitator π_{old}, and untrained discriminator $D_{\phi_{\text{old}}}$.
2: **Output:** Trained imitator and discriminator.
3: **for** iteration in $1, 2, \ldots, n$ **do**
4: Run expert π_{ex} for T timesteps to collect state, action, and reward data $s_1, \ldots, s_T$, $a_1, \ldots, a_T$, and $r_1, \ldots, r_T$.
5: Collect imitation actions $\hat{a}_1, \ldots, \hat{a}_T$ from $s_1, \ldots s_T$ using imitator $\pi_{\theta_{\text{old}}}$.
6: Obtain imitation-based rewards from Discriminator $D_{\phi_{\text{old}}}$ of the structure given in (2.5) and (2.6) in the form of $\text{logit}(D_{\phi_{\text{old}}}(s_t, \hat{a}_t, s_{t+1}, a_{t+1}))$.
7: Obtain training rewards by normalizing and combining the imitation-based expert rewards with weights of $\beta \in [0, 1]$ and $(1 - \beta)$.
8: Train imitator π_θ from θ_{old} on training rewards.
9: Train discriminator D_ϕ from ϕ_{old} to identify expert from imitator actions via binary logistic regression.
10: $(\theta_{\text{old}}, \phi_{\text{old}}) \leftarrow (\theta, \phi)$
11: **end for**

In experiments here, the imitator was trained jointly on imitative rewards and the expert's rewards in the same manner as in GAIL-RL with $\beta = 0.75$ in order to encourage the imitator to stay on a more

reasonable distribution of states than it otherwise would. Consequently, this approach will also be referred to as BAQE-RL, though unlike with GAIL-RL, the imitator's critic was not used as the victim model.

Importantly, under this threat model, the action distribution $\pi_{\text{ex}}(a|s)$ is not available for the expert, so that of the victim is always used in the discriminator. Although this will be a poor proxy for the expert's policy early in training, it will approach the expert's behavior as the imitator improves. Moreover, to reduce variance and simplify the structure of the discriminator, the successor action a' given to the discriminator was always that of the expert. So on two dual calls to the discriminator $D(s, a, s', a')$ for the expert and imitator, only a was ever different.

Unlike GAIL, this approach learns the victim's Q function rather than its value function. For the Gfootball environment, the action space was discrete and the adversary's observations were augmented with the Q estimate for all possible actions. When paired with a good action estimate, this is strictly richer than the value estimate alone.

3

Experiments

This chapter presents results for training efficient and insidious adversarial policies. These experiments involve training a number of different reinforcement learning "characters" which each play a unique role in experiments. These characters are outlined in Table 3.0.1.

All adversaries are trained for a total of 15 million timesteps against basic victim agents which were trained in two stages for a total of 20 million timesteps. The efficient and insidious adversaries' learning curves over time are compared to a baseline from adversarial agents who had unmitigated access to the victim. Additionally, for experiments with insidious adversaries, output similarity to the benign expert over time is also presented.

Due to limited compute[1], results from only two

[1]Each agent took days to train.

Character	Purpose	Details
Victim	-Training adversaries against	
Efficient Adversary	-Attacking victim with few/no queries	-Trained w/ proxies and victim modeling -Threat model in Table 2.1.1
Imitator/ Victim Model	-Training efficient adversaries -Pretraining -Victim modeling	-Trained w/ GAIL-RL or BAQE-RL
Insidious Adversary	-Attacking victim and avoiding detection	-Trained w/ imitation of benign expert -Threat model in Table 2.1.1
Benign Expert	-Training insidious attacks -Adversaries imitate	-Grey-box access required

Table 3.0.1: Reinforcement learning "characters" used in experiments.

identically-configured adversaries are shown in each experiment. Both the individual and averaged results from these trails are displayed with light and dark curves respectively in plots. In future experiments, a greater number of replicates will be trained.

Figure 3.0.1(a) shows training curves for the agents using which all subsequent victim models and adversaries were trained. When discussed in context of training IRL models with GAIL-RL and BAQE-RL, these agents will be referred to as "experts" and when in context of training adversaries against them, they will be referred to as "victims." This and all other curves plotted in this chapter were created by storing batch-averaged values and smoothing with a kernel whose size was 1/50th the number of total batches. Recall that agents are rewarded 1 for scoring, -1 for being scored on, and 0.1 for advancing 1/10th of the way down the field (thus, the reward was not zero-sum). These victims were trained in two stages: first against a simple rule-based "bot" agent and second against a more difficult pretrained agent from [39], each for 10 million timesteps. The increased difficulty of the second stage of training is reflected in

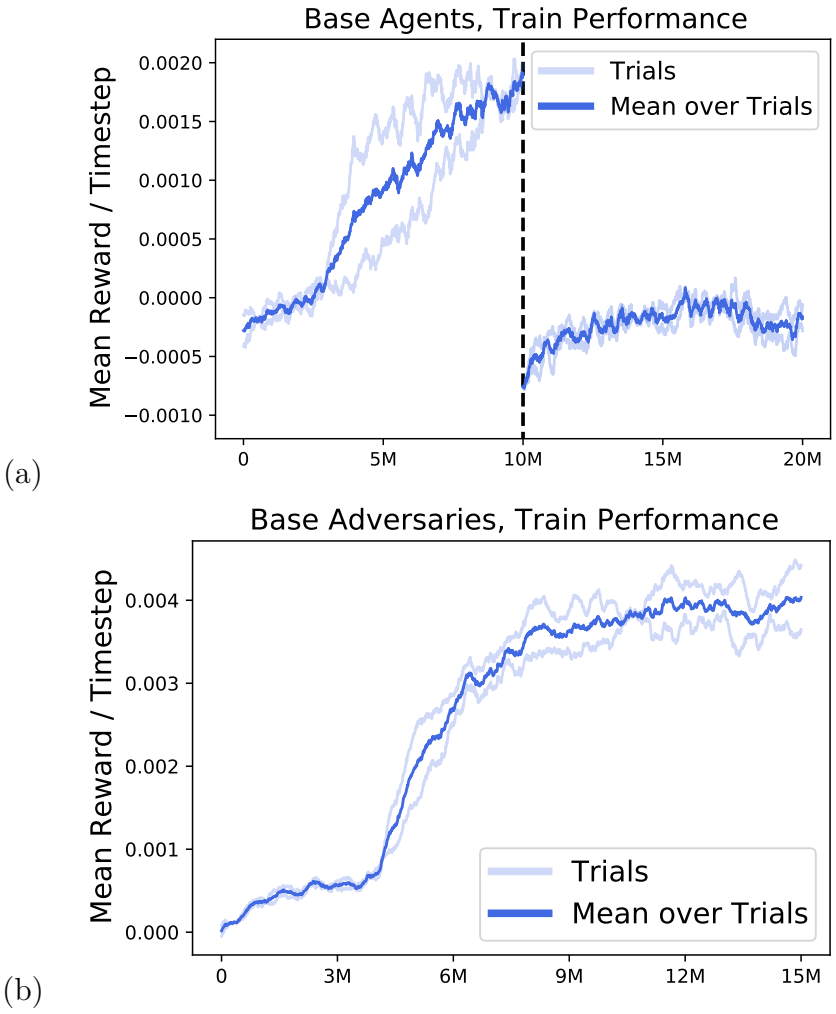

Figure 3.0.1: Training curves for base agents and control adversaries: Average reward per timestep for individual trials (light) and mean average reward per timestep across both trials (dark). (a) Performance for training the victims that were used in all subsequent experiments. Training was done first against a simple "bot" and second against a pretrained agents from [39]. (b) Performance for non-insidious control adversaries trained with unmitigated access to the victim. On a testing evaluation over 150,000 timesteps, these adversaries averaged 11.9 and 10.8 points over the victims per game.

3.0.1(a) by the fact that training performance decreases after 10 million timesteps at the switch to the more difficult training

opponent. Additional details are in the Appendix.

Figure 3.0.1(b) shows the training curves from the control experiments for training adversaries against these victims. These basic adversaries were non-insidious and had unmitigated black-box access to the victim which fits the type of threat model used in [18, 19, 54, 70]. The training curves show average reward per timestep, and at final evaluation over 150,000 timesteps, these adversaries averaged 11.9 and 10.8 more points than the victims per game. Note that there is not a direct relationship between average reward per timestep and average net points scored per game because of the shaped reward which also rewarded agents for successfully advancing the ball down the field while maintaining possession. As a result, an agent who nets an average of zero points over its opponent per game will still tend to have an average reward per timestep greater than zero.

Most of the subsequent experiments for training efficient and insidious adversaries involve victim imitation or modeling learned via either GAIL-RL or BAQE-RL. After a total of 3 million training steps, the GAIL-RL imitators shared a modal action with the experts 67% and 67% of the time, while the BAQE-RL agents fared significantly worse at 36% and 27% of the time. Figure 3.0.2 additionally shows the averaged KL divergences ($KL(\pi_{\text{imitator}}, \pi_{\text{expert}})$) over time for the GAIL-RL and BAQE-RL imitators for imitating the victims/experts. Importantly, these imitators are trained to maximize agreement with the expert's actions rather than to minimize the KL divergence, and there are signs that the measurements in Figure 3.0.2 were decoupled from the imitation objective. See Box 3 for a further discussion.

Box 3: On the difficulty of using KL-Divergence to measure successful imitation: The KL divergence between two discrete policies π, π' at a state s is given by $KL(\pi, \pi'|s) = \sum_{a \in \mathcal{A}} \pi(a|s) \log\left(\frac{\pi(a|s)}{\pi'(a|s)}\right)$. In GAIL-RL and BAQE-RL experiments here, $KL(\pi_{\text{imitator}}, \pi_{\text{expert}})$ was measured such that it could be interpreted as an expectation over the imitator's policy. Additionally, when training insidious adversaries in Section 3.3, a portion of the reward was based on the KL divergence.

Note that the KL divergence is not symmetric, and it is sensitive to $\pi'(a|s)$ values that are close to zero. However, this sensitivity is not shared with the action-agreement objective with which the imitators were trained. For example, even if $\pi(a|s)$ and $\pi'(a|s)$ are almost identical over all actions, for any single action, a, if $\pi'(a|s) \to 0$, then $KL(\pi, \pi'|a) \to \infty$.

Empirically, it seems that this measure was significantly decoupled from the training objective. Recall that the GAIL-RL imitators obtained final modal action agreements with the experts of 67% and 67% while the BAQE-RL ones achieved 36% and 27%. However, the final KL divergences for the GAIL-RL imitators is almost identical to one of the BAQE-RL agents. Moreover, while one BAQE-RL agent was a significantly better imitator according to the modal agreement measure, it was the one with the *higher* final KL divergence in Figure 3.0.2. As a result, the curves in Figure 3.0.2 should be taken only as a loose measure of imitation effectiveness. Additionally, in experiments with insidious adversaries, the KL-divergence was often very high and seemed to adversely shape the reward as will be discussed in Sections 3.3. Future work will experiment with using the L1 distance, $\sum_{a \in \mathcal{A}} |\pi(a|s) - \pi'(a|s)|$, between two policies instead.

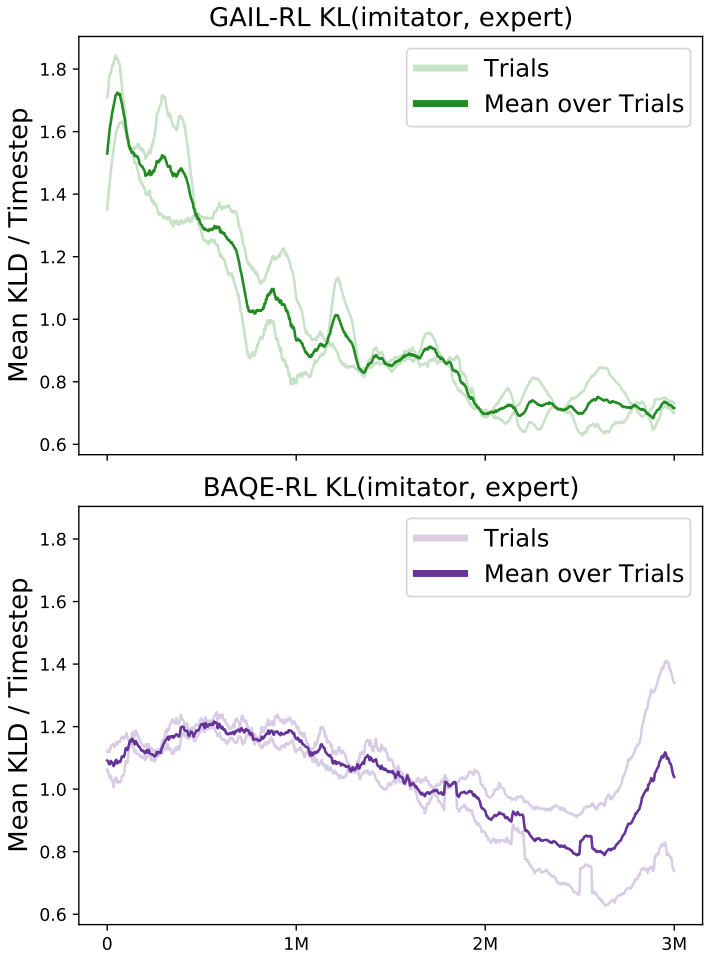

Figure 3.0.2: GAIL-RL and BAQE-RL imitation KL Divergences over training: Trial (light) and mean (dark) batch-averaged KL divergences over 3 million training timesteps for imitators. See Box 3 for additional comments.

3.1 Transfer from an Imitative Proxy

These experiments are motivated by the simple idea that learning to attack victim and a proxy which imitates it ought to be similar. An analogous type of black box attack method was introduced for attacking supervised learning models by [52] based on developing an adversary against a model trained on the labels produced by an

intended victim and then transferring those adversaries against the victim. Meanwhile, the finding from [27] that adversarial inputs in RL, though not adversarial policies, often transfer between independently trained victims further suggests that transfer from an imitative proxy to the intended victim may work effectively and efficiently.

Two types of pretraining-based adversaries are tested here. First adversaries trained only against the GAIL-RL imitators for 10 million timesteps, and second those same adversaries finetuned against their intended victims for an additional 5 million timesteps. Here, only the GAIL-RL imitators were used because they had better imitative performance based on their modal action agreement with the experts. Notably though, the BAQE-RL imitators may still learn a more useful victim model for model-based attacks as is investigated next in Section 3.2. The fine-tuned adversaries are compared to the first 5 million timesteps of the baseline adversaries which were given unrestricted query access to the victim in Figure 3.0.1(b). The number of training steps for the baseline adversaries was kept equal to the number of total training steps for the finetuned adversaries from both pretraining and finetuning.

Figure 3.1.1 plots the pretraining and fine-tuning performances of these adversaries using the GAIL-RL imitators. The high initial performance of the adversaries at the beginning of the fine-tuning stage suggests that through the use of only non-interactive demonstrations, useful proxies can be developed for pretraining both zero and few-shot adversaries. However, it remains an open question whether similarly good transfer would occur if a stronger expert/victim were used. This is discussed further in Section 4.3

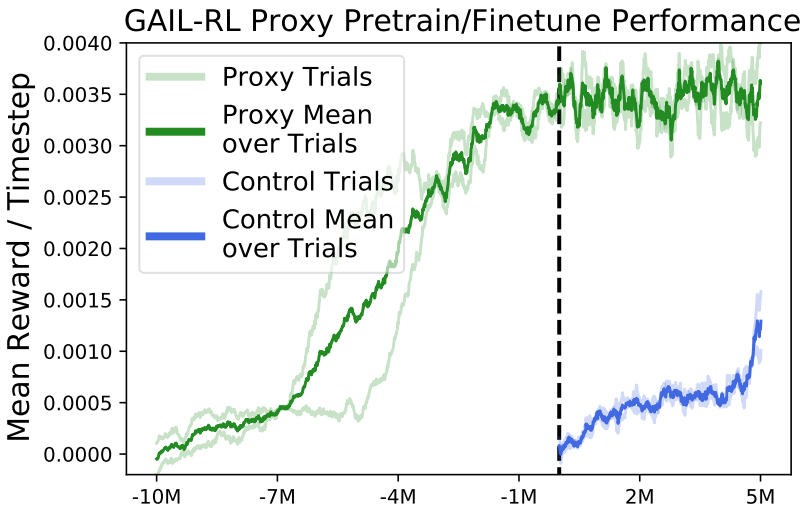

Figure 3.1.1: Training curves for pretraining adversaries against GAIL-RL proxies and and fine-tuning against the intended victim: Trial (light) and across-trial mean (dark) batch-averaged training reward per timestep. Curves are aligned such that the number of training timesteps on the intended victim for both the pretrained and control agents are aligned for $t \geq 0$. Training performances against the proxies are plotted at timesteps $t < 0$.

3.2 Observation Augmentation via Victim Modeling

Whereas pretraining on an imitative proxy only requires the *imitators* learned by GAIL-RL, the next set of experiments for training efficient adversarial policies hinged on using the learned imitators as action models plus the Value estimates for GAIL-RL or the Q estimates across all 19 actions for BAQE-RL as models of the victim's internal state. At each timestep, the state was observed by the adversary and given to the victim action and value/Q models. Then these estimates were used to augment the adversary's observations. As a result, these approaches are related to those used in [23] and fall into the broad category of model-based RL [48] algorithms. There are three principle

motivations for using this strategy to learn more sample efficient and effective adversarial policies.

First, having predictions for the victim's action is useful for choosing adversarial actions. As a simple example, consider an adversarial kicker and victim goalie playing football. The adversary being able to anticipate whether the goalie would next move left or right would be useful for knowing what action would allow it to evade the goalie.

Second, having an estimate of the victim's Value or Q function at a given timestep may help the adversary make strategic planning decisions. Consider again the example of an adversarial kicker against a victim goalie. In this case, the Value or Q estimate will convey information about whether the goalie "thinks" it is likely to be scored on soon. One concrete example of when this could be useful is that a low Value or Q estimate could then signal to the adversary that they should attempt to score within the next several timesteps because the goalie seems to be in a vulnerable state.

Third, the value or Q function estimate can help the adversary learn its own Value function efficiently. If a good estimate is obtained and added to the adversary's observations, then only the simplest of relations would need to be learned between the input and adversary's Value function as opposed to a more complicated one in which the Value must be inferred from the state alone. To aid in this process, the Value and Q estimates were concatenated into to the adversaries' network twice: once in the first dense layer of the network and once in the last so that the network could both learn a complex function of the Value or Q estimate over time and quickly learn a simple relationship between the estimate and its own value function.

An additional helpful fact about training adversaries via augmenting their observations with victim Value or Q estimates is that these estimates can be biased or even systematically biased in

different ways across the state space with little consequence. So long as the estimates are low-variance, the adversary's network can learn from experience to compensate for the bias.

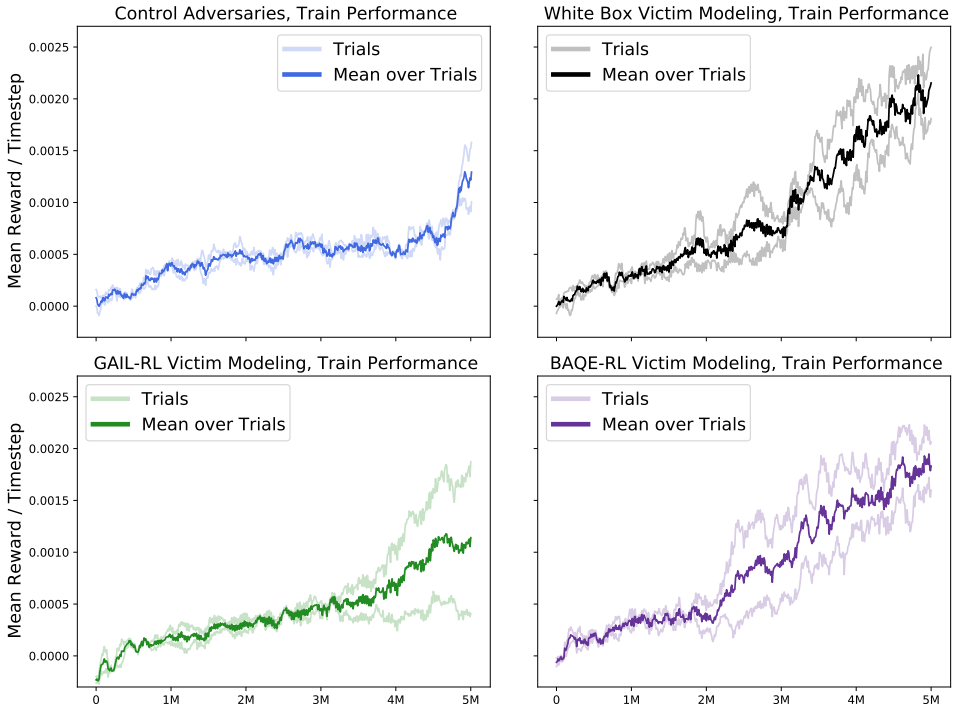

Figure 3.2.1: Training curves for white-box, GAIL-RL, and BAQE-RL victim-model adversaries compared to a control: Trial (light) and across-trial mean (dark) batch-averaged training rewards per timestep. Training curves against the intended victim are plotted over 5 million timesteps for each adversary.

Figure 3.2.1 shows training curves for adversaries using white-box, GAIL-RL, and BAQE-RL victim models over 5 million timesteps of training compared to the control adversary also shown in figure 3.0.1(b). Only 5 millions steps were run to reduce the computational load of these experiments which required the separate victim model (or two in the case of BAQE-RL) to be queried at each timestep. There is a significant amount of variance in performance involved in

training adversaries over this timescale. However, these results provides some evidence of victim-modeling being useful for developing more adversarial behavior more quickly for the white-box and BAQE-RL victim models. However, future experiments with more replicates will be needed to more clearly see trends. Future experiments will also investigate victims which are more difficult to train against in which these victim models may prove to be more useful. See Section 4.3 for details and future plans.

While not one of the principal goals of this thesis, training the adversaries who had white-box access to the intended victim as a baseline for comparison with the black box adversaries in Figure 3.2.1 demonstrates a viable white-box attack method. Though an algorithmically simple approach, to this author's knowledge no prior works have trained RL-based adversaries by augmenting observations with the victim's actor and/or critic outputs. However, this is a natural strategy to potentially improve on the black-box methods used here and in related works when white-box access to the victim is available. Pending future experiments with more difficult victims will help to further investigate the potential of this method to improve attacks.

3.3 Insidious Attacks via Expert Imitation

The final key experiments presented here focus on *insidious* adversaries without constraints on efficiency. As discussed in Section 2.1, this approach is motivated by the fact that an adversary may need to be difficult to detect to be be viable at all in some environments, and undetectability makes diagnosing the cause of a failure more difficult for the operators of the victim. Here, the assumption from the previous two sections that queries to the victim are expensive or limited is relaxed, but it is assumed that a detector

of the form $D(a|s)$ may be in place.

First, a simple approach was used in which the adversary was jointly trained to beat the victim and to imitate an independently-trained benign agent as outlined in Section 2.3.2. The benign-experts used here were the other victims: i.e. `victim A` was used as a benign expert for training an insidious attack against `victim B` and vice versa. This was simple to do in the Gfootball environment by horizontally flipping the observations and actions of an agent trained to play on one side of the field such that it could also play on the other.

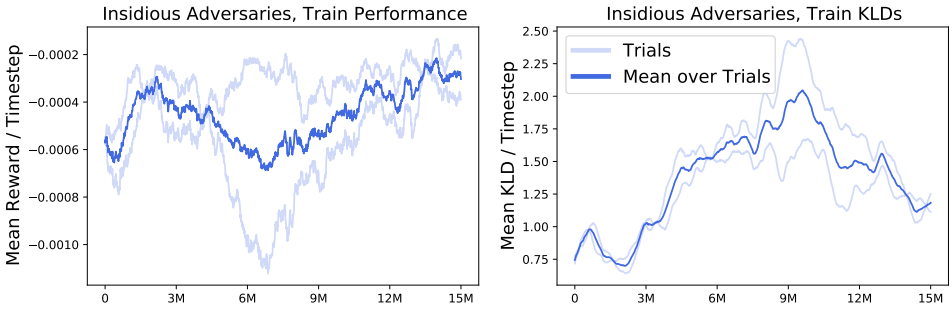

Figure 3.3.1: Training reward and KL-divergence curves for insidious adversaries: Trial (light) and across-trial mean (dark) batch-averaged training reward per timestep (left), and average KL divergence between the expert and imitator per timestep (right). Experts used here were independently trained but identically configured to the victims. See Box 3 for comments on the use of KL divergence as a training objective.

In addition to the straightforward behavioral cloning approach above, it was tested whether using a victim model as done previously in Section 3.2 would also help an adversary to be more insidious. Recall the finding from [38] that using a victim's value function to schedule a small number of allowable adversarial perturbations resulted in an improved ability to thwart a victim under a budget for how many perturbations could be introduced. For a similar reason,

allowing the adversary to observe the estimate of the victim's actions and Value/Q function may allow for more strategically insidious adversaries.

To gain intuition for this, note the fact that to be insidious, an agent needs to maintain a balance between acting adversarial to beat the victim and acting benign to avoid detection. One way to accomplish this would be to only act in a characteristically adversarial way at opportune moments while otherwise imitating the benign agent. Knowing an estimate of a victim's actions and Value/Q function would aid in this because crucial states in a trajectory are most likely to happen when the victim's Value or Q function is high or low. These adversaries were trained here with the same dual objective as in the experiments without victim modeling in Figure 3.3.1 and with the same observation augmentations used in Section 3.2.

The adversaries shown in both Figures 3.3.1 and 3.3.2 failed to learn policies that either were effective against the victims or imitated the experts well. A likely cause for this failure as mentioned in Box 3 is that the KL divergences is sensitive to small action-probabilities in the second action distribution it receives as an argument. Here, unlike with experiments involving training the GAIL-RL and BAQE-RL the imitators' rewards were directly based on KL divergences, and it was observed that they would sometimes be extremely high.[2] To alleviate this, they were clipped at 10, but they may have nonetheless resulted in an irregular distribution that caused a misshaping of the reward when the adversarial and imitative rewards were normalized and combined. To fix this problem, proceeding work will experiment with rewards based on the L1 distance, $\sum_{a \in \mathcal{A}} |\pi(a|s) - \pi'(a|s)|$, instead which better reflects the goal of action-matching between an insidious adversary and a benign expert. See Section 4.3 for further discussion.

[2]Some values over 10^{25} were observed.

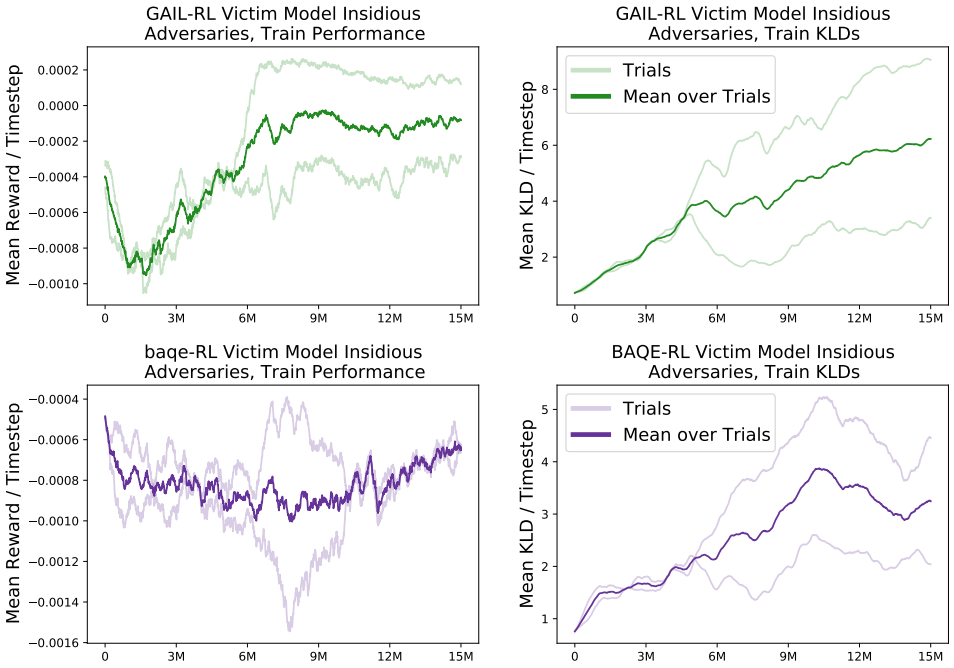

Figure 3.3.2: Training reward and KL-divergence curves for insidious victim-modeling adversaries: GAIL-RL (top) and BAQE-RL (bottom) victim models. Trial (light) and across-trial mean (dark) batch-averaged training reward per timestep (left), and average KL divergence between the expert and imitator per timestep (right). Compare to Figure 3.3.1.

4

Discussion

4.1 Implications and Impact

With recent rapid advances in AI have come unprecedented opportunities for new applications, including in safety-critical domains, and this work makes progress toward an understanding of more sophisticated threats from adversaries in reinforcement learning than have been investigated before. This thesis has introduced and tested new methods for creating efficient and insidious attacks against deep reinforcement learning systems. These also include a new algorithm, BAQE, for inferring an agent's Q function only from noninteractive demonstrations. This thesis also argues for the importance of continuing to study more realistic threat models in order to develop a germane understanding of threats from adversaries.

Though work is still ongoing, these preliminary results suggest that pretraining against an imitative proxy to develop adversarial policies seems to be a promising approach when access to the victim (or similar agents) is limited but when non-interactive, benign demonstrations are available. There is also evidence of victim-modeling being helpful for more efficiently developing adversarial behavior. However, these findings will benefit from further testing against more advanced victims. Finally, the methods introduced here for developing insidious adversaries seemed to be unsuccessful for either learning behavior that was adversarial or imitating the intended expert. New experiments here are needed and forthcoming.

Overall, the positive results demonstrated thus far suggest that attacks under significantly more limited threat models than have been demonstrated in previous works are viable. This underscores a need for caution in the further deployment of RL systems, especially for safety-critical applications in environments that are difficult to control such as those of self driving cars [55]. In addition to the direct benefits of avoiding failures in these contexts, deploying robust systems will also be crucial for building trust between humans and AI systems in domains where the responsible incorporation of AI may lead to large benefits to safety and efficiency.

In addition to black-boxifying models and limiting query access, additional defense and detection methods should be developed alongside more progress in deploying deep RL systems. These should include adversarial training [18, 54, 70], ensembling, post-processing of network outputs, and the use of sophisticated detectors. Given that the attack methods introduced here all pivot on specification learning from demonstrations, limiting the availability of even benign data will also be crucial.

When working to develop robust RL systems, it is key to bear in

mind that an adversarial policy need not be implemented *against* nor *by* a reinforcement learning system. On one hand, so long as a victim acts according to a policy that can be associated with some reward, they can be subject to black box attacks from reinforcement learners like the ones developed here. On the other hand, an adversarial policy against a victim need not be developed by a reinforcement learner. For example, human players of video games constantly develop strategies optimized to exploit the weaknesses of computer-controlled competitors to great effect. The possibility of adversarial behavior of humans against RL-based agents could pose dangers in domains involving human interaction.

Broadly, this thesis can be understood in context of the field of AI safety. As AI systems become more advanced and widely-used, it is increasingly important to ensure that they are well-aligned with human goals and values. Of particular concern should be AI systems that have par-human or superhuman intelligence in a given domain [7, 12, 57, 68]. In addition to the goal of better understanding more imminent threats, this author hopes that this thesis and related work will contribute to the broader goal of developing an AI safety toolbox that will help to guide the beneficial development of AI. In future research concerning adversaries or other technical challenges toward developing aligned AI, focusing on more realistic threat models will be key. Hopefully, continued work will help to guide the positive development of AI systems with an emphasis on safety, reliability, and trust.

4.2 Limitations

Concerning efficiency, it is key to note that even if a method for producing efficient adversarial policies is effective at reducing sample complexity, simply *more* efficient may not be efficient enough to pose

genuine threats against many real systems. The experiments here involving victim modeling suggest that benefits from model-based attacks against reinforcement learners may be limited. As a result, additional work beyond the transfer from proxy experiments in this thesis focusing on zero-shot generation of adversarial policies will be useful.

A concern involving the experiments with insidious adversaries is that they were only trained to make their action outputs *given* a state similar to a benign expert's. This approach is only designed to fool a simple type of detector which discriminates based on actions conditional on a state. However a more sophisticated detection method could additionally focus on an agent's state visitation distribution. Subsequent work should take more sophisticated detection methods than are considered here into account.

Another limitation which applies to all experiments presented here is that besides relatively simple limitations on query access or relatively naive detectors, no other defense and detection methods were assumed to be in place under the threat models used here. In addition to efficiency and insidiousness, another prerequisite to more realistic attacks will be overcoming other practical defense methods. For instance, many autonomous systems incorporate mechanisms that guard against erratic actions such as PID assistance [35, 58].

4.3 Ongoing Work

The work presented here is ongoing, and future experiments will involve a number of new strategies to produce more robust results. In addition to running a greater number of trials for each experiment, three major methodological changes are planned. After expanding the set of experiments used to investigate these threats, a paper will be submitted for publication.

Using Stronger Victims: All adversaries in experiments here were trained to beat the victims whose training curves are shown in Figure 3.0.1(a). However, as suggested by the initial performance of the adversaries in Figure 3.0.1(b), these victims were not particularly high-performing, and they would only tend to net approximately 3 points over untrained victims per game. The two-stage training process was intended to result in superior final performance but may have caused more poorly performing adversaries than using the first stage alone. In early exploratory experiments, victims were only trained against a "bot" agent without being finetuned against the pretrained deep-learning based agent from [39] yet seemed to have superior initial performance against adversaries. An issue with using weak victims is that policies which are adversarial to them may require little (over)fitting to the specific victim's strategy and instead simply execute actions that are generically good for accumulating reward when playing against a weak opponent. When tested against each other's victims, the adversaries in Figure 3.0.1(b) had similar performance compared to their own which suggests that this may be the case for the adversaries here.

One problem with using weak adversaries is that the benefits from pretraining against an imitative proxy may be largely due to pretraining against any agent at all as opposed to an imitator of the particular intended victim. A second problem involves the fact that if learning an adversarial strategy doesn't require (over)fitting to the particular victim in question, then one would expect to see few or no benefits from training with a victim model. This may explain why there were not greater improvements from victim-modeling adversaries over the control adversaries shown in Figure 3.2.1. Future experiments will use victims who are trained in multiple stages against "bots" of increasing difficulty rather than against any pretrained deep RL agents.

Measuring Imitation with the L1 Distance: Despite the fact that KL divergence is commonly used in RL as a measure of distance between policies (e.g. in TRPO [60]), it seems to be a poor measure for successful imitation for experiments here. As discussed in Box 3, and Section 3.3 the KL divergence is non-symmetric, sensitive to low $\pi'(a|s)$ values, and likely caused ill-shaped rewards for experiments with insidious adversaries. Unlike what the KL divergence measures, standard implementations of imitation/IRL algorithms like behavioral cloning, GAIL [26], AIRL[16], and BAQE reward the imitators based on how often their actions are consistent with the expert's. To better match this objective, future experiments will use the L1 loss, $L_1(\pi, \pi'|s) = \sum_{a \in \mathcal{A}} |\pi(a|s) - \pi'(a|s)|$ as a measure of successful imitation.

Using a Learned Reward with the BAQE Discriminators: While the GAIL-RL imitators learned to imitate the experts relatively well with modal action agreements of 67% and 67%, the BAQE-RL discriminators performed more poorly with modal action agreements of only 27% and 36%. One cause of this may be the use of a non-learned reward inside the discriminator. As shown in equations (2.5) and (2.6), the discriminator used the ground truth reward from the MDP in order to learn an estimate of the expert's Q function. One concern with this strategy is that it assumes the expert is acting consistently with achieving the MDP's reward, but there may be significant differences between the MDP's reward and a reward function for which the agent's policy is optimal. Empirically, it also seems that using the ground-truth reward as opposed to a learned reward proxy also harmed performance. In early exploratory experiments in which a learned reward model was used, imitators achieved superior modal action agreements.

A potential solution to this would be to use a learned *state-only* reward function instead of the ground truth reward from the MDP. In

fact, the resulting variant of BAQE strongly resembles AIRL [16] except for $h(a, s)$ from equation (2.6) being a function of both state and action and no queryable demonstrations from the expert being used for training the discriminator. As an added benefit, this version of BAQE could also be used to learn either r or Q when the ground truth reward is unknown.

A Appendix

A.1 What Makes a Policy "Adversarial?"

The notion of an "adversary" for a deep learning system was popularized by [67] and subsequent works which developed malicious images that are both *effective*, meaning that they fool an image classifier, and *subtle*, meaning that they only differ from a benign image by a very small-norm perturbation. While they often transfer to other models [32, 46, 51, 69], typically, these adversaries are also *victim-specific* in the sense that they are created specifically to fool a particular victim.

Effectiveness and victim-specificity for adversaries in supervised learning are analogous to the same characteristics for adversaries in RL. However, the subtlety property does not have an analogous one that fully captures what this and related works refer to as an "adversarial" policy. A notion of subtlety for RL adversaries that would be most directly analogous to the one in supervised learning would be that the adversary has a very small expected distance in policy space from a benign agent marginalized across that agent's state visitation distribution. This is closely related to the goal of training insidious adversaries in this thesis, but in this and most

related work, no notion of subtlety is incorporated into the operational definition of an adversarial policy.

For the sake of understanding practical threats, the performance of an adversary against its victim is the only crucial concern, and some works such as [5] have considered an attacker to be "adversarial" to a victim if it simply succeeds at thwarting the victim. Other related works, however, have defined an adversarial policy as one that is victim-specific. For example, [19] trains and provides videos of adversarial humanoid agents in competitive environments which learn to perform specific actions which induce their particular victim to fall to the ground. This notion of an adversarial policy being (over)fit to a single victim can be useful and in fact describes well the victim-modeling based attacks introduced here. But overall, an "adversarial" policy here simply refers to one which is able to make its victim fail. This is roughly the same as the definition operationalized in [5].

A.2 Architecture, Training, and Environment Details

In the Gfootball environment [39], the victim agents were first trained for 10 million timesteps (each game lasted 3000 timesteps) against a simple "bot" referred to as `11_vs_11_easy_stochastic` in the Gfootball codebase which used a simple rule-based policy and is not implemented by a neural network. The victims were then trained for another 10 million timesteps against a pretrained network-based agent from [39], also known as `11_vs_11_easy_stochastic`. This collision between the name of the bot and the pretrained deep learning agents is from the Gfootball codebase and coincidental. The victims which had then been trained for a total of 20 million timesteps were then used to train adversaries against. All adversaries were trained for a total of 15 million timesteps meaning that they were trained for 5 million fewer timesteps overall than the victims they were attacking.

As discussed in Section 2.2, the Gfootball software allows for simple 115-length vectors of extracted observations to be used for training. However, in experiments here, the agents' observation space was visual, consisting of $72 \times 96 \times 16$ inputs whose channels encoded the positions of the left team, right team, ball, and active player. These

four channels were stacked with others from the previous three timesteps to yield the final depth of 16. This temporal stacking allowed for information such as velocity to be inferred from the inputs.

The policy network architecture used was the same as the `gfootball_impala_cnn` network used in [39]. This was a ResNet [24] architecture with 4 modules each consisting of an initial convolution and pooling layer followed by two residual blocks, each consisting of two convolutional layers and residual connections. All activations were ReLU, all filter sizes were 3×3, all pooling sizes were 2×2, and all padding was `same`. After the residual modules, outputs were flattened, and processed through a dense layer of 256 neurons. This created a 256-dimensional latent from which both the value estimate and the policy were computed for Proximal Policy Optimization.

For victim-model attacks, the architecture was modified slightly. For these experiments, the pre-flattening portions of the policy network were identical, but both before and after the dense layer, the latent vector was concatenated with the policy and Value/Q estimates. The policy estimate was stacked over the past four timesteps just as the visual inputs were which made its length $19 \times 4 = 76$. For the value-models resulting from GAIL-RL, the value estimates for the past four timesteps were also concatenated making the total addition to the latent vectors a length-80 vector. For the BAQE-RL-based models, the Q function evaluated at all 19 actions for the past four timesteps were used making for an additional $19 \times 4 = 76$ nodes for a total length-152 vector being concatenated into the latent. These observations were concatenated with the latent vector both before and after the dense layer in order to allow for a more complex function to be learned from the pre dense-layer inputs and for the value/Q estimates to be readily available for the agent to learn its own Value function from the post-dense inputs.

All agents including GAIL-RL and BAQE-RL imitators were trained using PPO from OpenAI Baselines [13] with the Adam optimizer [37] with hyperparameters for PPO based on the search conducted in [39]: $\gamma = 0.993$, a learning rate of 0.000343, PPO surrogate objective clipping at $\epsilon = 0.08$, and gradient norm clipping at 0.64. A minimum batch size of 4096 was used for training all agents, though for the simple training experiments that only required one or two models to be queried as part of the process multiple environments

were run in parallel where possible, and all were used for the same update resulting in larger batch sizes by a factor of two. For most agents, entropy regularization with a coefficient of 0.003 was used, while for imitators in GAIL-RL and BAQE-RL, a higher coefficient of 0.03 was used to discourage the imitator from outputting modal actions. Training the discriminator only on examples it incorrectly predicted and training the imitator only when it output a different action from the discriminator also helped with this.

As discussed in Sections 2.3.3 and 2.3.4, GAIL and BAQE, imitators were trained with normalized versions of the discriminator-based and expert/environment-based rewards with weights $\beta = 0.75$ and $1 - \beta = 0.25$. Except for this fact and the entropy regularization coefficient as mentioned above, they were trained identically to other agents. The discriminator for GAIL-RL experiments had an architecture mimicking that of the policy networks for the RL agents and was also trained with Adam and an L2 weight decay with a coefficient of 0.0001. The BAQE discriminator calculated an internal Q estimate using the same residual architecture as the GAIL discriminator and was also trained with Adam using the same weight decay as for GAIL. The BAQE discriminator had additional batch normalization [33] layers added for stability. The output was calculated according to equations (2.5) and (2.6). As discussed in Section 2.3.4, the network took as input s, a, s', a', and $\pi(s)$. However, to simplify the implementation, the a' from the expert was always used, and the expert and imitator were only distinguished based on a. To abide by the threat model outlined in Section 2.1, the victim's $\pi(s)$ was always used because grey-box access to the victim/expert was not available.

Code for this project will be available in the final paper submitted for publication. That pending, code can be shared on an *ad hoc* basis by request.

A.3 "Explain it to me like I'm a highschooler."

While impressive progress is being made in AI and related fields, there is a communication gap between researchers and the public which too often serves as a barrier to the spread of information. While this thesis is primarily meant for people who are familiar with

reinforcement learning, this author believes that this and many other works will benefit from a brief section dedicated to explaining key concepts free of jargon and in a way that is more understandable to those outside the specialty.

Machine learning refers to the process of learning to accomplish a task from data. Much of the recent progress in machine learning has involved the use of neural networks. These information-processing systems resemble biological brains in many ways, and they tend to be versatile and excellent at learning good solutions to complicated problems when trained with enough data. In recent years, research in machine learning has made immense progress in tasks such as image recognition, natural language processing, and data compression.

Another area of great interest for active research is known as "reinforcement learning." Reinforcement learning refers to the process by which some *agent* acts in an *environment* and learns by some formalized process of trial-and-error to learn how to maximize the attainment of some *reward*. For example, many reinforcement learning algorithms are tested by having them learn to play simple video games with the objective of achieving high scores. However, there are much more important applications of reinforcement learning than video games such as self-driving cars.

One concern with machine learning systems – particularly ones based on neural networks – is a vulnerability to *adversarial* inputs. An adversary can be broadly understood as some input that has been designed to cause a machine learning system to fail. For example, adversaries can be created to fool image classifiers by taking a normal image which is classified correctly and then engineering a small, human-imperceptible, perturbation to that image's pixels which successfully fools the classifier.

In reinforcement learning, one type of adversarial threat can come from another agent. It has been found in multiagent environments that if a *victim* agent is frozen (i.e. no longer actively learning and just implementing a learned policy) and another *adversarial* agent is trained with the goal of making the victim fail, that the adversary can learn to effectively thwart the victim. This can happen even if the victim is normally good at accomplishing its goals when interacting with other nonadversarial agents. For example, [19] experimented

with a simulated kicker and goalie in a soccer/football-playing game. Interestingly, they found that an adversarial goalie would learn to defeat the kicker not by blocking the ball when kicked but by making unusual movements that induced the kicker to fall to the ground!

The effectiveness of adversarial policies in reinforcement learning poses concerns that real-world systems could be attacked with these adversaries. This is especially concerning in safety-critical domains like self-driving cars. It is important to understand these threats, but previous research into adversarial agents in machine learning is limited. Previous works have trained adversarial agents by training with extensive access to a victim and without the goal of avoiding detection measures. However, in the real world, one's ability to access and train against a victim will often be limited or expensive, and detection systems might be in place. In order to better understand more sophisticated threats from adversarial agents in reinforcement learning, this thesis develops and tests three types of methods for training adversaries to be efficient and difficult to detect.

1. *Pretraining against an imitator (Efficiency):* Even if one's ability to interact with the intended victim is limited, access to observations from the victim in nonadversarial settings may be available. Using only observations of the victim, an agent is trained to imitate the victim, and then that imitator is used as a proxy for the intended victim for training. This reduces and potentially eliminates the need to access the actual victim.

2. *Victim modeling (Efficiency):* In addition to learning an imitator from benign data, one can also train another model of the victim which learns to estimate how valuable the victim "thinks" its situation is at a given point in time. This can then be added to the adversary's observations along with an imitator's action predictions in order to provide information relevant to what strategy the victim will use and how it can be beaten. This is made available to the adversaries during training to increase their ability to effectively and quickly learn to beat the victim.

3. *Expert imitation (Undetectability):* Unlike the previous two strategies which focus on efficient adversaries, these experiments

focus on training adversaries that are hard to detect. Here, this is done by jointly training an adversary to jointly beat its victim and imitate a non-adversarial *expert* agent.

By introducing methods for developing adversarial agents in ways that are efficient and difficult to detect, this thesis makes progress toward a better understanding of what threats reinforcement learning systems may face. This author hopes that continued work in understanding and preventing these threats will make future systems more safe and trustworthy.

References

[1] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[2] Andrea Bajcsy, Dylan P Losey, Marcia K O'Malley, and Anca D Dragan. Learning robot objectives from physical human interaction. In *Conference on Robot Learning*, pages 217–226. PMLR, 2017.

[3] Maria-Florina Balcan, Avrim Blum, Dravyansh Sharma, and Hongyang Zhang. On the power of abstention and data-driven decision making for adversarial robustness. *arXiv preprint arXiv:2010.06154*, 2020.

[4] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017.

[5] Vahid Behzadan and William Hsu. Adversarial exploitation of policy imitation. *arXiv preprint arXiv:1906.01121*, 2019.

[6] James Bell, Linda Linsefors, Caspar Oesterheld, and Joar Skalse. Reinforcement learning in newcomblike environments. 2020.

[7] Nick Bostrom. *Superintelligence*. Dunod, 2017.

[8] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[9] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[10] Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. Activation atlas. *Distill*, 4(3):e15, 2019.

[11] Stephen Casper. Achilles heel for agi/asi via decision theoretic adversaries. *arXiv preprint arXiv:2010.05418*, 2020.

[12] Brian Christian. *The Alignment Problem: Machine Learning and Human Values*. WW Norton & Company, 2020.

[13] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. https://github.com/openai/baselines, 2017.

[14] Wei Du and Shifei Ding. A survey on multi-agent deep reinforcement learning: from the perspective of challenges and applications. *Artificial Intelligence Review*, pages 1–24, 2020.

[15] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.

[16] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

[17] Ted Fujimoto, Timothy Doster, Adam Attarian, Jill Brandenberger, and Nathan Hodas. The effect of antagonistic behavior in reinforcement learning. 2021.

[18] Victor Gallego, Roi Naveiro, and David Rios Insua. Reinforcement learning under threats. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9939–9940, 2019.

[19] Adam Gleave, Michael Dennis, Neel Kant, Cody Wild, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

[20] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. Georgia Institute of Technology, 2013.

[21] Dylan Hadfield-Menell, Anca Dragan, Pieter Abbeel, and Stuart Russell. Cooperative inverse reinforcement learning. *arXiv preprint arXiv:1606.03137*, 2016.

[22] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart Russell, and Anca Dragan. Inverse reward design. *arXiv preprint arXiv:1711.02827*, 2017.

[23] He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *International conference on machine learning*, pages 1804–1813. PMLR, 2016.

[24] 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*, pages 770–778, 2016.

[25] Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz de Cote. A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183*, 2017.

[26] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016.

[27] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

[28] Evan Hubinger, Chris van Merwijk, Vladimir Mikulik, Joar Skalse, and Scott Garrabrant. Risks from learned optimization in advanced machine learning systems. *arXiv preprint arXiv:1906.01820*, 2019.

[29] Inaam Ilahi, Muhammad Usama, Junaid Qadir, Muhammad Umar Janjua, Ala Al-Fuqaha, Dinh Thai Hoang, and Dusit Niyato. Challenges and countermeasures for adversarial attacks on deep reinforcement learning. *arXiv preprint arXiv:2001.09684*, 2020.

[30] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. *arXiv preprint arXiv:1804.08598*, 2018.

[31] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. *arXiv preprint arXiv:1807.07978*, 2018.

[32] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175*, 2019.

[33] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[34] Hong Jun Jeon, Smitha Milli, and Anca D Dragan. Reward-rational (implicit) choice: A unifying formalism for reward learning. *arXiv preprint arXiv:2002.04833*, 2020.

[35] Michael A Johnson and Mohammad H Moradi. *PID control*. Springer, 2005.

[36] Sham M Kakade. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

[37] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[38] Jernej Kos and Dawn Song. Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452*, 2017.

[39] Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4501–4510, 2020.

[40] Tencent Keen Security Lab. Experimental security research of tesla autopilot. 2019.

[41] Joel Lehman, Jeff Clune, and Dusan Misevic. The surprising creativity of digital evolution. In *Artificial Life Conference Proceedings*, pages 55–56. MIT Press, 2018.

[42] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[43] Hsueh-Ti Derek Liu, Michael Tao, Chun-Liang Li, Derek Nowrouzezahrai, and Alec Jacobson. Beyond pixel norm-balls: Parametric adversaries using an analytically differentiable renderer. *arXiv preprint arXiv:1808.02651*, 2018.

[44] Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. A survey of reinforcement learning informed by natural language. *arXiv preprint arXiv:1906.03926*, 2019.

[45] Björn Lütjens, Michael Everett, and Jonathan P How. Certified adversarial robustness for deep reinforcement learning. In *Conference on Robot Learning*, pages 1328–1337. PMLR, 2020.

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

[47] Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. A joint model of language and perception for grounded attribute learning. *arXiv preprint arXiv:1206.6423*, 2012.

[48] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. Model-based reinforcement learning: A survey. *arXiv preprint arXiv:2006.16712*, 2020.

[49] Jesse Mu and Jacob Andreas. Compositional explanations of neurons. *arXiv preprint arXiv:2006.14032*, 2020.

[50] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pages 663–670, 2000.

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

[52] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[53] Georgios Papoudakis, Filippos Christianos, Arrasy Rahman, and Stefano V Albrecht. Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1906.04737*, 2019.

[54] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2817–2826. JMLR. org, 2017.

[55] Adnan Qayyum, Muhammad Usama, Junaid Qadir, and Ala Al-Fuqaha. Securing connected & autonomous vehicles: Challenges posed by adversarial machine learning and the way forward. *IEEE Communications Surveys & Tutorials*, 22(2):998–1026, 2020.

[56] Anton Raichuk, Carlos Riquelme, Damien Vincent, Karol Kurach, Lasse Espeholt, Marcin Michalski, Michał Zając, Olivier Bousquet, Olivier Frederic Bachem, Piotr Michal Stanczyk, et al. Google research football: A novel reinforcement learning environment. 2019.

[57] Stuart Russell. *Human compatible: Artificial intelligence and the problem of control*. Penguin, 2019.

[58] Anirban Santara, Sohan Rudra, Sree Aditya Buridi, Meha Kaushik, Abhishek Naik, Bharat Kaul, and Balaraman Ravindran. Madras: Multi agent driving simulator. *arXiv preprint arXiv:2010.00993*, 2020.

[59] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[60] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[61] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[62] Rohin Shah, Dmitrii Krasheninnikov, Jordan Alexander, Pieter Abbeel, and Anca Dragan. Preferences implicit in the state of the world. *arXiv preprint arXiv:1902.04198*, 2019.

[63] Samuel Henrique Silva and Peyman Najafirad. Opportunities and challenges in deep learning adversarial robustness: A survey. *arXiv preprint arXiv:2007.00753*, 2020.

[64] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general

reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[65] Theodore R Sumers, Mark K Ho, Robert D Hawkins, Karthik Narasimhan, and Thomas L Griffiths. Learning rewards from linguistic feedback. *arXiv preprint arXiv:2009.14715*, 2020.

[66] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[67] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[68] Max Tegmark. *Life 3.0: Being human in the age of artificial intelligence*. Knopf, 2017.

[69] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017.

[70] Eugene Vinitsky, Yuqing Du, Kanaad Parvate, Kathy Jang, Pieter Abbeel, and Alexandre Bayen. Robust reinforcement learning using adversarial populations. *arXiv preprint arXiv:2008.01825*, 2020.

[71] Shuo Wang, Shangyu Chen, Tianle Chen, Surya Nepal, Carsten Rudolph, and Marthie Grobler. Generating semantic adversarial examples via feature manipulation. *arXiv preprint arXiv:2001.02297*, 2020.

[72] Christian Wirth, Riad Akrour, Gerhard Neumann, Johannes Fürnkranz, et al. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136):1–46, 2017.

[73] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 2019.

[74] Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–41, 2020.