Security of deep reinforcement learning

by

Vahid Behzadan

B.Eng., University of Birmingham, UK, 2012

M.S., University of Nevada, Reno, 2016

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2019

Abstract

Since the inception of Deep Reinforcement Learning (DRL) algorithms, there has been a growing interest from both the research and the industrial communities in the promising potentials of this paradigm. The list of current and envisioned applications of deep RL ranges from autonomous navigation and robotics to control applications in the critical infrastructure, air traffic control, defense technologies, and cybersecurity. While the landscape of opportunities and the advantages of deep RL algorithms are justifiably vast, the security risks and issues in such algorithms remain largely unexplored. It has been shown that DRL algorithms are very brittle in terms of their sensitivity to small perturbations of their observations of the state. Furthermore, recent reports demonstrate that such perturbations can be applied by an adversary to manipulate the performance and behavior of DRL agents. To address such problems, this dissertation aims to advance the current state of the art in three separate, but interdependent directions. First, I build on the recent developments in adversarial machine learning and robust reinforcement learning to develop techniques and metrics for evaluating the resilience and robustness of DRL agents to adversarial perturbations applied to the observations of state transitions. A main objective of this task is to disentangle the vulnerabilities in the learned representation of state from those that stem from the sensitivity of DRL policies to changes in transition dynamics. A further objective is to investigate evaluation methods that are independent of attack techniques and their specific parameters. Accordingly, I develop two DRL-based algorithms that enable the quantitative measurement and benchmarking of worst-case resilience and robustness in DRL policies. Second, I present an analysis of *adversarial training* as a solution to the brittleness of Deep Q-Network (DQN) policies, and investigate the impact of hyperparameters on the training-time resilience of policies. I also propose a new exploration mechanism for sample-efficient adversarial training of DRL agents. Third, I address the previously unexplored problem of model extraction

attacks on DRL agents. Accordingly, I demonstrate that imitation learning techniques can be used to effectively replicate a DRL policy from observations of its behavior. Moreover, I establish that the replicated policies can be used to launch effective black-box adversarial attacks through the transferability of adversarial examples. Lastly, I address the problem of detecting replicated models by developing a novel technique for embedding sequential watermarks in DRL policies. The dissertation concludes with remarks on the remaining challenges and future directions of research in emerging domain of DRL security.

Security of deep reinforcement learning

by

Vahid Behzadan

B.Eng., University of Birmingham, UK, 2012

M.S., University of Nevada, Reno, 2016

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2019

Approved by:

Co-Major Professor
William H. Hsu

Approved by:

Co-Major Professor
Arslan Munir

Copyright

Abstract

Since the inception of Deep Reinforcement Learning (DRL) algorithms, there has been a growing interest from both the research and the industrial communities in the promising potentials of this paradigm. The list of current and envisioned applications of deep RL ranges from autonomous navigation and robotics to control applications in the critical infrastructure, air traffic control, defense technologies, and cybersecurity. While the landscape of opportunities and the advantages of deep RL algorithms are justifiably vast, the security risks and issues in such algorithms remain largely unexplored. It has been shown that DRL algorithms are very brittle in terms of their sensitivity to small perturbations of their observations of the state. Furthermore, recent reports demonstrate that such perturbations can be applied by an adversary to manipulate the performance and behavior of DRL agents. To address such problems, this dissertation aims to advance the current state of the art in three separate, but interdependent directions. First, I build on the recent developments in adversarial machine learning and robust reinforcement learning to develop techniques and metrics for evaluating the resilience and robustness of DRL agents to adversarial perturbations applied to the observations of state transitions. A main objective of this task is to disentangle the vulnerabilities in the learned representation of state from those that stem from the sensitivity of DRL policies to changes in transition dynamics. A further objective is to investigate evaluation methods that are independent of attack techniques and their specific parameters. Accordingly, I develop two DRL-based algorithms that enable the quantitative measurement and benchmarking of worst-case resilience and robustness in DRL policies. Second, I present an analysis of *adversarial training* as a solution to the brittleness of Deep Q-Network (DQN) policies, and investigate the impact of hyperparameters on the training-time resilience of policies. I also propose a new exploration mechanism for sample-efficient adversarial training of DRL agents. Third, I address the previously unexplored problem of model extraction

attacks on DRL agents. Accordingly, I demonstrate that imitation learning techniques can be used to effectively replicate a DRL policy from observations of its behavior. Moreover, I establish that the replicated policies can be used to launch effective black-box adversarial attacks through the transferability of adversarial examples. Lastly, I address the problem of detecting replicated models by developing a novel technique for embedding sequential watermarks in DRL policies. The dissertation concludes with remarks on the remaining challenges and future directions of research in emerging domain of DRL security.

Table of Contents

List of Figures

List of Tables

Acknowledgments

My graduate studies have been an odyssey of educational nomadism and adventurous explorations. I began my first Ph.D. program in the communications research group at the University of Birmingham in 2012, and have since been a member of 3 other research groups working on vastly different areas of research. I look at my experiences in these research groups in a positive light, as each enabled me to expand my horizons and learn about ideas that came to shape my current research interests. Hence, I would like to begin by thanking my advisors in each of these groups: Prof. Peter Gardner at the University of Birmingham for introducing me to the fundamentals of research, and supporting my work on reconfigurable radio front-ends for cognitive radios; Prof. Shamik Sengupta at the University of Nevada, Reno, for supporting my explorations in game theory and the security of distributed networks; Prof. Arslan Munir at UNR and K-State for facilitating my research on complex adaptive systems and AI safety; and Prof. William Hsu at K-State for supporting my work on this dissertation.

I would like to further extend my appreciation of Prof. Hsu for his support of my research in cyber threat intelligence and robotics, for providing the opportunity to extend my teaching experiences by inviting me to guest lecture at his machine learning courses, and for advising me on how to build a successful academic career. I am forever indebted to him, and will always refer to him as a role model for not only my academic conduct, but also in my personal life.

I would also like to thank my students and colleagues who trusted me with their professional development, and let me grow as an aspiring academic: Mr. Avishek Bose, who is now building his Ph.D. dissertation on our collaborative research on cyber threat intelligence analysis, and our brilliant undergraduate team members in this project: Mr. Carlos Aguirre, Ms. Emily Davic, Ms. BreAnn Anshutz, and Ms. Marissa Shivers. Also, I would like to thank Mr. James Minton who worked on the Ethical Decision-Making in AI project with

me for his senior undergraduate project. I consider myself lucky to have worked with such talented researchers, and will eagerly follow their inevitable accomplishments in the coming years.

Indeed, surviving these 8 years of graduate studies would not have been possible without the support of my dear friends and family. I am forever thankful to my friends and colleagues in the KDD lab (K-State), the ISCAAS lab (K-State), the Computer Networking Lab (UNR), and the Communications Research Group (UoB) for their friendship. I would also like to thank my parents, for their continuous support through the many valleys of my educational journey. I thank my brother, Reza, who sowed the seeds of my interest in AI, security, and computer science during my formative years. Likewise, I would like to thank my brother Prof. Ali, who inspired my passion for teaching and research. Also, I am deeply grateful to my partner, soon-to-be Dr. Marzieh Soltanolkottabi, for her incredible support and patience during my work on this dissertation.

Lastly, I would like to thank the esteemed members of my dissertation committee, Prof. Mitchell Neilsen, Prof. Joshua Weese, Prof. Bala Natarajan, and Prof. Shing Chang, for their valuable guidance, comments, and advice towards improving this dissertation. I also thank Prof. Scott DeLoach, head of the computer science department at K-State, who played a key role in facilitating the completion of this dissertation.

This work was supported in part by the National Science Foundation (NSF) (NSF-CNS-1743490). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the NSF.

Dedication

To my brother, Reza Behzadan, who taught me to code before I could write, and inspired my passion for computer science and artificial intelligence.

Chapter 1

Introduction

Since its inception, a founding objective of Artificial Intelligence (AI) has been the creation of agents that are able to autonomously learn various skills and behaviors from self-guided interactions with the environment. This objective is greatly motivated by a biological analogue in natural learning, which is the process of learning through utilization of past experiences to evaluate and enact behaviors that result in higher benefits [2]. Inspired by the models and analyses of this mechanism, machine learning research has developed the framework of *Reinforcement Learning* (RL) to leverage such developments in the engineering of AI agents.

The basic formulation and algorithms of RL have been available for decades[3]. However, for much of this time, the feasibility and applicability of this framework remained severely constrained to simple problems and environments. This was mainly due to the high computational requirements of such algorithms, as well as the need for manual feature engineering of the RL agent's operating environments[4]. In recent years, these limitations have begun to overturn with the increasing accessibility of high-performance computing (e.g., via GPUs), as well as the advancements in deep learning [5]. The latter provides RL with the powerful feature learning capabilities of deep neural networks, thus enabling the end-to-end learning of complex skills from raw sensory inputs in complex environments. This new framework, named *Deep RL* (DRL), has led to groundbreaking results in a variety of complex tasks, such as learning to play video games [6], beating the human champion at the game of Go

[7], and autonomous navigation [8].

This recent surge in the versatility and feasibility of DRL has motivated extensive research on its application to many domains, ranging from autonomous navigation [9] and robotic manipulation [10] to healthcare [11], algorithmic trading [12], and automated control of complex systems such as smart grids [13], transportation systems [14] and air traffic management [15]. In some domains, such as robotics [16] and portfolio management [16], applications of DRL are grown beyond academic interest and into deployed commercial products.

With the increasing adoption of DRL into commercial and critical systems, ensuring the security of these applications is of growing importance. Integration of DRL controllers with cyber-physical and financial systems gives rise to a novel and attractive attack surface to adversaries aiming to manipulate and compromise such systems. As detailed in Chapter 3, the problems arising from the security requirements of DRL are widely different from those studied in seemingly similar areas, such as safe RL [17]. The objective of safe RL is to ensure that in the natural dynamics of application environments, the agent does not learn to behave in ways that are in breach of compliance with some given criteria for safe operation. In the formulation of safe RL problems, the compliance of policies with such requirement is both defined and evaluated within the confines of assumptions about the natural states and dynamics of the operating environment. In contrast, the security problem is concerned with settings in which an adversarial element intentionally seeks to compromise the natural states and dynamics of the system for malicious purposes. For instance, a safety requirement imposed on RL-based policies of autonomous navigation is collision avoidance. From a high-level perspective, this requirement can be defined as maintaining a minimum distance with other vehicles and objects, as measured by the on-board sensors of the vehicle. In the safety problem, this fundamental measurement is assumed to be either perfectly accurate, or with errors arising from the inherent imperfections of the sensors. However, in the security problem, the adversary may intentionally manipulate the sensory measurements by inducing perturbations that are different in type and scale to those that may naturally occur in the sensors. Targeted crafting of such perturbations may thus force a "safe" RL agent into collision states [18]. Hence, while there are overlaps between RL safety and security, a

different perspective and approach is required to study the security problem in DRL.

Similar challenges in supervised and unsupervised machine learning models has given rise in the growing area of "adversarial machine learning". However, as detailed in Chapter 2, while the security of supervised and unsupervised machine learning systems has enjoyed extensive attention from the research community [19][20][21], the work on vulnerabilities and security of DRL is sparse and sporadic. Since the reports by Behzadan & Munir [1] and Huang et al. [22] in early 2017, the proceeding literature have studied this problem with focus on narrow aspects of security in DRL. Therefore, there remains a pressing demand for holistic and foundational studies in this emerging area.

1.1 Motivation

As discussed in Chapter 4, a major emphasis of the state of the art in DRL security is on the vulnerability of policies to state-space perturbations. In particular, the manipulation of the policy via adversarial examples has remained a main focus of current literature on this issue. However, this bias towards adversarial example attacks had led to two critical shortcomings: first, the analyses of such attacks fail to disentangle the vulnerability caused by the learned representation and that which is due to the sensitivity of the DRL dynamics to distributional shifts in state transitions. Second, the performance of defenses proposed for adversarial example attacks are inherently limited to the considered attack mechanisms. As a promising technique for mitigation of adversarial examples, *adversarial training* is known to enhance the robustness of machine learning models to the type of attack used for generating the training adversarial examples, while leaving the model vulnerable to other types of attacks[23].

Furthermore, the current literature fails to provide solutions and frameworks which can be used in practice to evaluate and improve the robustness and resilience of DRL policies to attacks that exploit their sensitivity to state transitions. Hence, there remains a need for quantitative approaches to measure and benchmark the resilience and robustness of DRL policies in a reusable and generalizable manner.

Moreover, the focus on state-space attacks leaves an important aspect of DRL security completely unexplored –that of targeting the confidentiality of DRL policies. With the envisioned growth of DRL applications, a critical business risk in developing DRL solutions stems from the unauthorized replication and stealing of proprietary policies. Also, similar to the case of supervised models, replicated policies may enable the adversaries to launch more efficient black-box attacks against DRL agents. Hence, the investigation of this aspect is of paramount importance for commercial use-cases of DRL.

1.2 Problem Statement and Scope

In response to such gaps, this dissertation aims to extend the current state of the art by addressing three fundamental research problems, enumerated as follows:

1. Development of general techniques and metrics for quantitative measurement and benchmarking of the resilience and robustness of DRL policies to state-space perturbations.

2. Understanding the dynamics of training-time perturbations, and the impact of design choices on the resilience of policies to training-time attacks.

3. Investigating the vulnerability of DRL agents to the theft or unauthorized replication of policies.

This dissertation is focused on two types of security compromises, adversarial perturbation of the states, and model replication attacks. Other classes of attacks (reviewed in Chapter 3) are beyond the scope of this work. Furthermore, the work on adversarial state perturbation aims for strict independence from assumptions on the attack technique or parameters. In particular, we refrain from considering adversarial example attacks in all research tasks related to state perturbation attacks. This is to ensure the generalization of results.

Also, this work is restricted to model-free DRL algorithms applied in environments with discrete action-spaces. This restriction is mainly to constrain the problem space to a tangible

extent for doctoral research. It is also intended to preserve the coherence and relevance of the formal and experimental results, and enable the meaningful comparison of findings across different research tasks.

1.3 Overview of Contributions

1. We develop two DRL-based techniques and corresponding metrics for the measurement and benchmarking of resilience and robustness of DRL policies to state-space perturbations, and evaluate their performance on Deep Q-Network (DQN), Advantage Actor-Critic (A2C), and Proximal Policy Optimization (PPO2) agents.

2. We formulate and analyze the problem of training-time attacks against DQN agents, and experimentally study the impact of hyperparameters and design choices on the resilience of DQN to such attacks, the results of which are presented as design-level guidelines.

3. We formally analyze the limitations of adversarial training for DQN agents, and establish an upper bound on the ratio of perturbed to nominal states for the successful adversarial training of DQN agents.

4. We address the sample-inefficiency of blanket adversarial training by proposing an adversarially-guided exploration mechanism for improved sample-efficiency and performance in the adversarial training of DQN agents.

5. We formulate the problem of policy imitation attacks, and present a proof-of-concept methodology for efficient black-box state perturbations based on the Deep Q-Learning from Demonstrations (DQfD) algorithm.

6. We investigate the transferability of adversarial examples between replicated and original policies, and demonstrate the application of this phenomenon in black-box adversarial attacks.

7. We investigate the problem of detecting stolen policies, and develop a novel scheme for the watermarking of DRL policies with sequential triggers.

1.4 Organization

This dissertation is organized in three parts. In **Part 1**, we present an overview of the required background for this work. Accordingly, Chapter 2 reviews the fundamentals of RL and DRL, and provides an overview of adversarial machine learning. Chapter 3 formulates the security problem in RL, and presents threat models for adversarial attacks against DRL. Chapter 4 reviews the current state of the art in the types and mechanisms for attacks at training and inference phases of DRL, the defenses against such attacks, and relevant evaluation metrics.

Part 2 investigates the resilience and robustness of DRL agents to adversarial state perturbations. Chapter 5 provides the formulation of resilience and robustness in DRL policies, and presents the details of two proposed techniques for the measurement and benchmarking of resilience and robustness at test-time. Chapter 6 extends the analysis to training-time attacks on DQN agents, and investigates the effect of design parameters on the training-time resilience of DQN policies. Chapter 6 presents a formal analysis of adversarial training for improving the robustness of DQN policies, and proposes a novel exploration mechanism with a better sample-efficiency as an alternative to blanket adversarial training.

Part 3 Focuses on the attacks targeting the confidentiality aspect of DRL policies. Chapter 8 introduces the problem of adversarial policy imitation, and presents two proof-of-concept attacks to establish the severity of adversarial policy imitation. Chapter 9 presents a novel scheme for sequential watermarking of DRL policies.

To conclude the dissertation, Chapter 10 presents a summary of the main contributions of this work, as well as remarks on future directions of research.

Part I

Background

Chapter 2

Preliminaries

2.1 Deep Reinforcement Learning

Reinforcement learning is concerned with agents that interact with an environment and exploit their experiences to optimize a decision-making policy. The generic RL problem can be formally modeled as learning to control a Markov Decision Process (MDP), described by the tuple $MDP = (\mathbb{S}, \mathbb{A}, \mathbb{R}, \mathbb{P})$, where $\mathbb{S}$ is the set of reachable states in the process, $\mathbb{A}$ is the set of available actions, $\mathbb{R}$ is the mapping of transitions to the immediate reward, and $\mathbb{P}$ represents the transition probabilities (i.e., state dynamics), which are initially unknown to RL agents. At any given time-step t, the MDP is at a state $s_t \in \mathbb{S}$. The RL agent's choice of action at time t, $a_t \in \mathbb{A}$ causes a transition from s_t to a state s_{t+1} according to the transition probability $\mathbb{P}(s_{t+1}|s_t, a_t)$. The agent receives a reward $r_{t+1} = \mathbb{R}(s_t, a_t, s_{t+1})$ for choosing the action a_t at state s_t. Interactions of the agent with MDP are determined by the policy π. When such interactions are deterministic, the policy $\pi : \mathbb{S} \rightarrow \mathbb{A}$ is a mapping between the states and their corresponding actions. A stochastic policy $\pi(s)$ represents the probability distribution of implementing any action $a \in \mathbb{A}$ at state s. The goal of RL is to learn a policy that maximizes the expected discounted return $E[R_t]$, where $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$; with r_t denoting the instantaneous reward received at time t, and γ is a discount factor $\gamma \in [0, 1]$. The value of a state s_t is defined as the expected discounted return from s_t following a policy

π, that is, $V^{\pi}(s_t) = E[R_t|s_t, \pi]$. The action-value (Q-value) $Q^{\pi}(s_t, a_t) = E[R_t|s_t, a_t, \pi]$ is the value of state s_t after applying action a_t and following a policy π thereafter.

There are three main approaches to solving RL problems according to their optimization objective: methods that are based on *value iteration*, those that are based on *policy search*, and a hybrid of the two in *actor-critic* configurations. The details of each approach is presented as follows:

2.1.1 Value Iteration and Deep Q-Network

Value iteration refers to a class of algorithms for RL that optimize a value function (e.g., $V(.)$ or $Q(.,.)$) to extract the optimal policy from it. As an instance of value iteration algorithms, *Q-Learning* aims to maximize for the action-value function Q through the iterative formulation of Eq. (2.1):

$$Q(s, a) = R(s, a) + \gamma max_{a'}(Q(s', a')) \tag{2.1}$$

Where s' is the state that emerges as a result of action a, and a' is a possible action in state s'. The optimal Q value given a policy π is hence defined as: $Q^*(s, a) = max_{\pi}Q^{\pi}(s, a)$, and the optimal policy is given by $\pi^*(s) = \arg\max_a Q(s, a)$.

The Q-learning method estimates the optimal action policies by using the Bellman formulation to iteratively reduce the *TD-Error* given by $Q_{i+1}(s, a) - \mathbf{E}[r + \gamma \max_a Q_i]$ for the iterative update of a value iteration technique. Practical implementation of Q-learning is commonly based on function approximation of the parametrized Q-function $Q(s, a; \theta) \approx Q^*(s, a)$. A common technique for approximating the parametrized non-linear Q-function is via neural network models whose weights correspond to the parameter vector θ. Such neural networks, commonly referred to as Q-networks, are trained such that at every iteration i, the following loss function is minimized:

$$L_i(\theta_i) = \mathbf{E}_{s,a\sim\rho(.)}[(y_i - Q(s, a, ; \theta_i))^2] \tag{2.2}$$

where $y_i = \mathbf{E}[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1})|s, a]$, and $\rho(s, a)$ is a probability distribution over states s and actions a. This optimization problem is typically solved using computationally efficient techniques such as Stochastic Gradient Descent (SGD).

Classical Q-networks introduce a number of major problems in the Q-learning process. First, the sequential processing of consecutive observations breaks the *i.i.d.* (Independent and Identically Distributed) assumption on the training data, as successive samples are correlated. Furthermore, slight changes to Q-values leads to rapid changes in the policy estimated by Q-network, thus giving rise to policy oscillations. Also, since the values of rewards and Qs are unbounded, the gradients of Q-networks may become sufficiently large to render the backpropagation process unstable.

A Deep Q-Network (DQN) [6] is a training algorithm designed to resolve these problems. To overcome the issue of correlation between consecutive observations, DQN employs a technique called *experience replay*: instead of training on successive observations, experience replay samples a random batch of previous observations stored in the replay memory to train on. As a result, the correlation between successive training samples is broken and the *i.i.d.* setting is re-established. In order to avoid oscillations, DQN fixes the parameters of a network $\hat{Q}$, which represents the optimization target y_i. These parameters are then updated at regular intervals by adopting the current weights of the Q-network. The issue of instability in backpropagation is also solved in DQN by normalizing the reward values to the range $[-1, +1]$, thus preventing Q-values from becoming too large.

Mnih et al. [6] demonstrate the application of this new Q-network technique to end-to-end learning of Q values in playing Atari games based on observations of pixel values in the game environtment. To capture the movements in the game environment, Mnih et al. use stacks of four consecutive image frames as the input to the network. To train the network, a random batch is sampled from the previous observation tuples $< s_t, a_t, r_t, s_{t+1} >$, where r_t denotes the reward at time t. Each observation is then processed by two layers of convolutional neural networks to learn the features of input images, which are then employed by feed-forward layers to approximate the Q-function. The target network $\hat{Q}$, with parameters θ^-, is synchronized with the parameters of the original Q network at fixed periods intervals. i.e.,

at every ith iteration, $\theta_t^- = \theta_t$, and is kept fixed until the next synchronization. The target value for optimization of DQN thus becomes:

$$y_t \equiv r_{t+1} + \gamma max_{a'} \hat{Q}(s_{t+1}, a'; \theta^-) \tag{2.3}$$

Accordingly, the update rule for the parameters in the DQN training process can be stated as:

$$\theta_{t+1} = \theta_t + \alpha(y_t - Q(s_t, a_t; \theta_t)) \nabla_{\theta_t} Q(s_t, a_t; \theta_t) \tag{2.4}$$

As for the exploration mechanism, the original DQN employs the ϵ-greedy technique, which monotonically decreases the probability of taking random actions as the training progresses [3]. However, recent literature proposes various alternatives such as parameter-space noise exploration [24]. Although adding independent noise for exploration is usable in continuous control problems, more sophisticated strategies inject noise that is correlated over time (e.g., from stochastic processes) in order to better preserve momentum [25].

The procedure of the original DQN technique is presented in Algorithm 1:

Algorithm 1 Deep Q-Network (DQN)[6]

Input: observations x_t, reward value r_t

Output: Q-function

Initialize replay memory D

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$

for *episode* $= 1$ to M **do**

Initialize sequence $s_1 = x1$ and pre-processed sequence $\phi_1 = \phi(s1)$

for $t = 1$ to T **do**

Following ϵ-greedy policy, select $a_t = \begin{cases} \text{a random action} & \text{with probability } \epsilon \\ \arg\max_a Q(\phi(s_t), a; \theta) & \text{otherwise} \end{cases}$

Execute action a_i in the emulator and observe reward r_t and state x_{t+1}

Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

Store transition $< \phi_t, a_t, r_t, \phi_{t+1} >$ in D

{// Experience Replay}

Sample random minibatch of transitions $< \phi_j, a_j, r_j, \phi_{j+1} >$ from D

Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$

Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ w.r.t. the network parameter θ

{// periodic update of target network}

Every C steps reset $\hat{Q} = Q$, i.e., set $\theta^- = \theta$

end for

end for

In [26], Hasselt et al. demonstrate that the single estimator used in the target rule (provided in Eq (2.3)) overestimates the expected return, as it uses the maximum operator to both select and evaluate an action. To alleviate this issue, Hasselt et al. propose a solution

[27] based on an extension of the Double Q-learning algorithm [26] to Double DQN (DDQN). Accordingly, this algorithm uses the online network Q to evaluate the policy, and uses the target network $\hat{Q}$ to estimate the action value. This is achieved by modifying Eq (2.3) as follows:

$$y_t \equiv r_{t+1} + \gamma \hat{Q}(s_{t+1}, \arg\max_{a'} Q(s_{t+1}, a'; \theta); \theta^-) \tag{2.5}$$

Prioritized Experience Replay

In the original DQN algorithm, the training process uniformly samples experiences from the replay memory. A proposed improvement of this approach is prioritized experience replay [28], in which the more significant experiences are assigned a higher probability of being sampled. The measure of significance considered in this technique is TD errors, the higher values of which indicate higher significance. The proposed procedure for prioritized replay uses importance sampling to avoid the introduction of bias in the update distribution. The evaluation of this technique in Atari environments demonstrate improved performance in both DQN and DDQN agents.

2.1.2 Policy Search Methods

The objective of policy search methods is to directly optimize the policy via either gradient-free or gradient-based methods. While gradient-free methods (e.g., evolutionary strategies) have resulted in some success in this area, much of the recent developments are focused on policy gradient (i.e., gradient-based) methods [5]. In such techniques, the policy is directly parametrized in the form $\pi(a|s;\theta)$, where π is a probability distribution over actions a when observing state s, as parameterized by θ, which can represent the weights of a neural network. In the training process of policy gradient methods, the agent applies this policy in the environment and collects experience samples, which are periodically used to update θ by estimating the gradient $\nabla_\theta E[R_t]$. Typically, the agent then discard these samples and

repeats this process on new samples, optimizing the policy iteratively. Two of the most significant algorithms for solving policy gradient are TRPO and PPO, detailed below:

Trust Region Policy Optimization (TRPO): TRPO has been shown to be relatively robust and applicable to domains with high-dimensional inputs [29]. To achieve this, TRPO optimizes a surrogate objective function, constrained using a quadratic approximation of the Kullback-Leibler (KL) divergence, as denoted by Eq.(2.6). Whilst TRPO can be used as a pure policy gradient method with a simple baseline, later work by Schulman et al. [30] introduces Generalized Advantage Estimation (GAE), which proposes several, more advanced variance reduction baselines. The combination of TRPO and GAE remains one of the stateof-the-art RL techniques in continuous control. However, the constrained optimization of TRPO requires calculating second-order gradients, limiting its applicability. The high-level procedure of TRPO is presented in Algorithm 2:

$$\max_{\theta} \mathbb{E}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t] \tag{2.6}$$

$$s.t. : \mathbb{E}_t[KL[\pi_{\theta_{old}}(.|s_t), \pi_\theta(.|s_t)]] \leq \delta \tag{2.7}$$

Algorithm 2 Trust Region Policy Optimization[29]

Initialize π_0

for $i = 0, 1, 2, ...$ until convergence **do**

Run π_i for T timesteps (or N trajectories)

Estimate advantage values $A_{\pi i}(s, a)$ at all timesteps

Compute policy gradient g

Use the conjugate gradient method to compute $F^{-1}g$

Compute rescaled step $s = \alpha F^{-1}g$ with line search

Apply update: $\theta = \theta_{old} + \alpha F^{-1}g$

end for

Proximal Policy Optimization (PPO): PPO [31] is a closely related algorithm that

improves on TRPO's sample complexity (i.e., the number of samples required to learn an optimal policy) by increasing the training use of each sample. It maximizes a clipped "surrogate" objective function:

$$L_t(\theta) = \mathbb{E}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\phi_\theta](s_t) \tag{2.8}$$

where $L_t^{VF} = (V_\theta(s_t) - V_t^{target})^2$, S denotes an entropy bonus, and L_t^{CLIP} is given by:

$$L_t^{CLIP}(\theta) = \mathbb{E}_t[\min(\rho_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t] \tag{2.9}$$

with $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, and ϵ is a hyperparameter with a typical value of 0.2. The *clip* function modifies the surrogate objective by clipping the probability ratio, thus removing the incentive for moving r_t outside of the interval $[1-\epsilon, 1+\epsilon]$. The high-level procedure of the PPO algorithm is presented in Algorithm 3:

Algorithm 3 Proximal Policy Optimization[31]

for iteration= $1, 2, ...$ **do**

  for iteration= $1, 2, ..., N$ **do**

    Run policy $\pi_{\theta_{old}}$ in environment for T timesteps

    Compute advantage estimates $\hat{A}_1, ..., \hat{A}_T$

  end for

  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$

  $\theta_{old} \leftarrow \theta$

end for

The PPO2 algorithm [31] follows the same procedure as PPO, however its implementation is modified to leverage GPUs for faster computation.

2.1.3 Actor-Critic Methods

Actor-critic methods combine value functions with an explicit representation of the policy. In such methods, the actor represents a policy that leverages the bootstrapping over an approximation of a value function (i.e., critic) to reduce variance and improve the rate of convergence [3]. A notable instance of such techniques are the advantage actor-critic algorithms, which are comprised of a policy $\pi(a_t|s_t;\theta)$ and an estimate of the value function $V(s_t;\theta_v)$. The policy and the value function are updated after every t_{max} steps, or when a terminal state is reached. The update performed by the algorithm is computed via:

$$\nabla_{\theta'} \log \pi(a_t|s_t;\theta') A(s_t, a_t;\theta,\theta_v) \tag{2.10}$$

where $A(s_t, a_t;\theta,\theta_v)$ provides an estimate of the advantage function given by:

$$\sum_{i=1}^{k-1} \gamma^i r_{t+1} + \gamma^k V(s_{t+k};\theta_v) - V(s_t;\theta_v) \tag{2.11}$$

the value of k may vary from state to state, with an upper-bound of t_{max}.

A variant of this method is Asynchronous Advantage Actor-Critic (A3C), in which the policy and value function estimates are updated with n-step returns in the forward view, in a manner similar to that of using minibatches. A3C uses separate actor-learner threads that sample the environment steps and update a centralized copy of the parameters asynchronously to each other. Also, the gradient update for A3C adds a entropy parameter $\beta\nabla_{\theta'} H(\pi(s_t;\theta'))$ to the loss function of eq. (2.10) to improve exploration by discouraging premature convergence to sub-optimal deterministic policies[32]. The high-level procedure of A3C is provided in Algorithm 4:

Algorithm 4 Asynchronous Advantage Actor-Critic (A3C) [32]

Global shared parameter vectors θ and θ_v, thread-specific parameter vectors $\theta^{'}$ and $\theta_v^{'}$

Global shared counter $T = 0, T_{max}$

Initialize step counter $t \leftarrow 1$

for $T \leq T_{max}$ **do**

Reset gradients, $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$

Synchronize thread-specific parameters $\theta^{'} = \theta$ and $\theta_v^{'} = \theta_v$

Set $t_{start} = t$, get state s_t

for s_t not terminal and $t - t_{start} \leq t_{max}$ **do**

Take a_t according to policy $\pi(a_t|s_t;\theta')$

Receive reward r_t and new state s_{t+1}

$t \leftarrow t + 1$, $T \leftarrow T + 1$

end for

$$R = \begin{cases} 0 & \text{for terminal} s_t \\ V(s_t, \theta_v^{'}) & \text{otherwise} \end{cases}$$

for $i \in \{t-1, ..., t_{start}\}$ **do**

$R \leftarrow r_i + \gamma R$

accumulate gradients wrt $\theta^{'}$: $d\theta \leftarrow d\theta + \nabla_{\theta'} log\pi(a_i|s_i;\theta^{'})(R - V(s_i;\theta_v^{'}))$

accumulate gradients wrt $\theta_v^{'}$: $d\theta_v \leftarrow d\theta_v + \nabla_{\theta_v'}(R - V(s_i;\theta_v^{'}))^2$

end for

Update asynchronously θ using $d\theta$, and θ_v using $d\theta_v$

end for

On the other hand, the *Advantage Actor-Critic (A2C)* algorithm [33] uses a synchronous approach to sample from separate environment instances and collects all data into one mini-batch to compute the gradient.

2.2 Security of Machine Learning

As data-driven systems, machine learning algorithms are known to be vulnerable to various types of adversarial actions. Such actions can be broadly classified as those affecting the training phase of the learning process, and those targeting the inference (i.e., test) phase [19].

In the training phase, adversaries may aim to influence the learning process by manipulating the training data. This type of attack is generally referred to as *poisoning* [34]. An example of poisoning attacks is the case of online spam classifiers, where an adversary can intentionally mislabel spam emails as benign to compromise the accuracy of the model retrained on new data [35].

In the inference phase, the adversary may implement an *evasion* attack by providing malicious input that induce incorrect inferences at the output of a machine learning model [36]. This type of malicious input is generally referred to as adversarial example [37]. A noteworthy property of adversarial examples in their transferability: an adversarial example crafted for a particular model may also affect other models with different architectures that are trained on datasets with similar distributions to that of the original model [38].

According to the objective of adversaries, adversarial example attacks are generally classified into the following two categories:

1. Misclassification attacks, which aim for generating examples that are classified incorrectly by the target network

2. Targeted attacks, whose goal is to generate samples that the target misclassifies into an arbitrary class designated by the attacker.

To generate such adversarial examples, several algorithms have been proposed, such as the Fast Gradient Sign Method (FGSM) by Goodfellow et al., [39], and the Jacobian Saliency Map Algorithm (JSMA) approach by Papernot et al. [40]. A grounding assumption in many of the crafting algorithms is that the attacker has complete knowledge of the target neural networks such as its architecture, weights, and other hyperparameters. In response,

Papernot et al. [41] proposed the first blackbox approach to generating adversarial examples. This method exploits the transferability of adversarial examples: an adversarial example generated for a neural network classifier applies to most other neural network classifiers that perform the same classification task, regardless of their architecture, parameters, and even the distribution of training data. Accordingly, the approach of [41] is based on generating a replica of the target network. To train this replica, the attacker creates and trains over a dataset from a mixture of samples obtained by observing target's interaction with the environment, and synthetically generated inputs and label pairs. Once trained, any of the algorithms that require knowledge of the target network for crafting adversarial examples can be applied to the replica. Due to the transferability of adversarial examples, the perturbed data points generated for the replica network can induce misclassifications in many of the other networks that perform the same task.

Another class of adversarial actions is comprised of those that aim to infer information about the internal parameters of the model or the training dataset [19]. One instance of such attacks is that of model extraction [42], in which the adversary estimates the parameters of a model from observations of its input-output data points. Besides compromising the confidentiality of models as intellectual properties, an adversary may utilize the extracted model in circumventing the difficulties of blackbox attacks [41]. Other instances of attacks on confidentiality are training data extraction [43] and membership attacks [44], which aim to infer information about the training set and extract personally-identifiable information from the model.

While the current literature on machine learning security includes various proposals for mitigation of these attacks, the majority of solutions provide ad hoc alleviation of the problem and are not generalizable to other classes of algorithms. For training-time attacks, notable solutions are those that aim to minimize the impact of outliers in models based on Principle Component Analysis (e.g., [45]) and Support Vector Machines (e.g., [46]). As for attacks targeting the inference phase, notable solutions include gradient masking (e.g., [47] and [48]) and injecting adversarial examples in the training dataset (e.g., [39]). Yet, all such defenses are shown to be weak against adaptive adversaries [19]. For more in-depth studies on the

state of security in machine learning, readers can refer to [19] and [21].

Chapter 3

DRL Threat Model

As with other types of machine learning techniques, DRL is also prone to adversarial actions [49][50] similar to those mentioned in Section 2.2. While many of the mechanisms and architectural aspects of DRL are similar to other machine learning techniques such as deep learning classifiers, the inherent differences in their learning dynamics and applications gives rise to security issues that are unique to DRL. A notable difference between DRL and classical machine learning techniques stems from the fact that DRL agents train to solve a sequential decision-making (i.e., control or planning) problem, whereas classical supervised and supervised methods are only concerned with a single-step prediction. The interdependence of a DRL agent's actions on its previous "decisions" increases the degrees of freedom for adversarial actions, and also gives rise to new challenges in detecting and mitigating such attacks. An example of these challenges is posed in the detection of attacks – a single observation of misclassification in deep learning models may indicate an adversarial perturbation; but for DRL agents, the objective of optimizing for maximum *cumulative* reward allows for short-term sacrifices. In many cases, this optimization objective implies a delayed-reward mechanism. For example, when playing chess, one may not receive any reward for any of its actions until the game is over. This is fundamentally different from supervised learning, where the availability of labels enable straightforward calculation of accuracy and other performance metrics for the model. Hence, observing even multitudes of seemingly incorrect

actions performed by a DRL agent is not necessarily indicative of malicious perturbations. A further challenge arises from the nature of DRL training : in general, supervised learning is trained on a dataset sampled from a fixed distribution. One consequence of this fact is that poisoning attacks cause a shift in the distribution, and hence detection and mitigation of such shifts has been the primary approach in defending against poisoning attacks. However, the exploratory behavior of DRL agents inevitably results in changes in the distribution of training samples. In other words, both the attack mechanism and any defensive approach against poisoning attacks on DRL are fundamentally different from those of supervised learning. Also, even after the training phase, many environments necessitate an optimal DRL policy to retain a level of randomness in their output actions (i.e., the action a_t is not always derived from a deterministic policy). A well-known example of such environments is the game of rock, paper, scissors [3]. Consequently, distinguishing adversarial attacks from benign actions derived from stochastic policies is not as straightforward as the case of supervised learning. To solidify our argument for the existence of unique challenges in the security of DRL, we note that both supervised and unsupervised mechanisms are shown to be reducible from the RL framework (i.e., can be formulated as instances of RL), but the inverse is not true [51]. In other words, DRL inherits the fundamental security issues in supervised and unsupervised learning, while its increased complexity give rise to other issues that are unique to RL and DRL.

As detailed in Section 2.2, adversarial attacks against DRL agents aim to compromise the normal operating criteria of such agents. Behzadan et al. [49] investigate such compromises from the standpoint of cybersecurity, in which the compromise may be viewed as affecting one or more dimensions of the Confidentiality, Integrity, Availability (CIA) triad [52], as detailed below:

Confidentiality of a DRL agent refers to the need for keeping the internal configurations of the agent from exposure to adversaries. These configurations include agent's reward function, the training mechanism (e.g., hyperparameters, algorithm), and the learned policy function. For instance, inference of the policy function can result in loss of proprietary assets. Furthermore, knowledge of the internal configurations may enable the adversary to launch

further attacks with more precision and efficiency.

Integrity in DRL is the ability to learn or enact policies in the manner intended by the designer. Adversaries may compromise the integrity of a DRL agent by forcing it to learn incorrect policies, or to perform actions other than those prescribed by a learned policy.

Availability is the ability of a DRL agent to perform training or actions when needed. Adversarial compromise of this dimension may be in the form of denying convergence during DRL training, or preventing the agent from acting in response to changes in the environment.

The problem of DRL security may resemble those studied under safe RL [17], but as mentioned in Chapter 1, the presence of adversarial intention in the security problem gives rise to challenges that are beyond the scope of safe RL. Also, we must differentiate between DRL security and the area of adversarial RL [53]. The latter is concerned with multi-agent RL settings, in which agents aim to maximize their returns in competition with other agents. While some DRL security problems can be modeled as adversarial RL [18], this cannot be generalized as the adversary is not necessarily a learning agent.

3.1 Attack Surface

Figure 3.1 presents a high-level depiction of a DRL agent and its components. Each of these components can be targeted in adversarial attacks, as detailed below:

Environment: According to the Markovian assumption of the MDP model presented in Section 2.1, the interaction between a DRL agent and its environment at time t are captured by the tuple $< s_t, a_t, r_{t+1}, s_{t+1} >$. In the RL settings, the environmental awareness of an agent is greatly dependent on its observation of the state s_t, and its transition to s_{t+1}. Hence, by perturbing the environment and its configuration by some vector δ_t (i.e, $s'_t = s_t + \delta_t$), an adversary is able to manipulate the agent's estimation of the current state, as well as the dynamics of the environment.

Observation Channel: DRL agents observe the environment via their sensors, and the sensory observatio of the state s_t may contain noise α_{s_t}, that is, $o_t = s_t + \alpha_{s_t}$. Adversarial manipulation of α_{s_t} through perturbing the sensor readings can compromise both the training

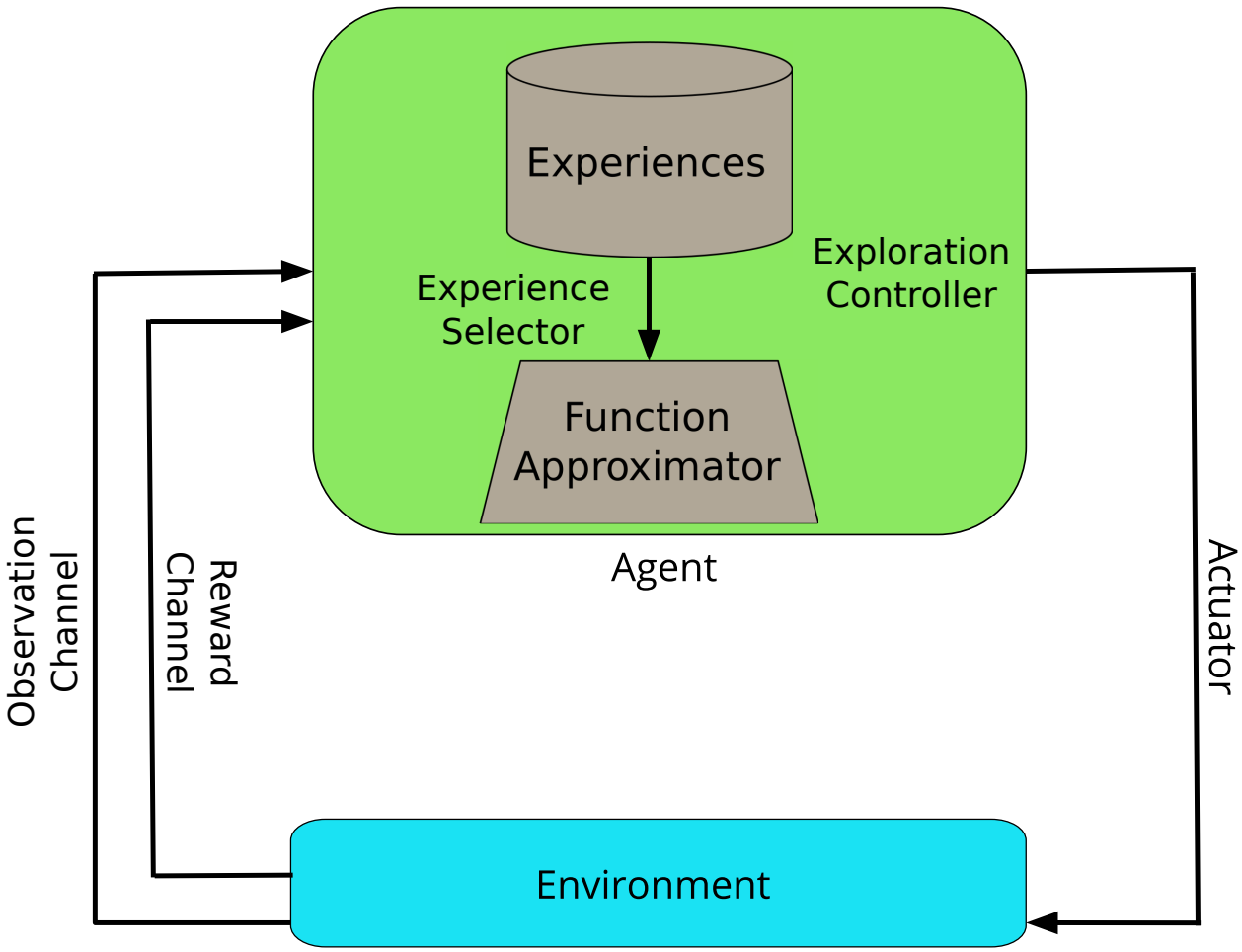

Figure 3.1: *Components of a DRL agent*

and inference processes [49]. Such attacks can be classified into two categories: those that exploit the imperfect representation learning of DRL agents (e.g., adversarial examples [1]), and those that target the imperfect generalization of DRL policies by manipulating the observation of state transition dynamics.

Reward Channel: Manipulation of the reward values r_t generated as a result of the actions of a DRL agent (and the consequent transitions) can greatly affect the training process. This case constitutes an instance of the *corrupted reward channel* problem [54], and can be formulated as follows: Given a true reward function $\dot{R}$ and a corruption function C, the observed reward function is defined as $\hat{R} : S \rightarrow \hat{R}$ as $\hat{R}(s) := C_s(\dot{R})$. The corrupted reward thus induces an observed MDP $< S, A, R, P, \hat{R} >$, which may poison the experience memory with potentially corrupted observations of rewards.

Actuator: DRL agents influence their environments by performing actions a_t via actuators. If the adversary can manipulate the actuator, the actual action performed will be different than that chosen by the agent, and hence the observed experience is corrupted, which can translate into poisoning of the experience memory.

Experience Storage and Selection: Direct manipulation of stored experiences of an agent enables the adversary to greatly influence the training process of the DRL agent. Such manipulations are essentially equivalent to perturbing the observations of an agent, but potentially at a lower cost and complexity for the adversary. Also, the experience selection mechanism can also be targeted to manipulate the training process. The analysis of such attacks (presented in Chapter 6) establish that techniques such as prioritized replay may in fact facilitate the poisoning of policy learning.

3.2 Attack Model

Threat modeling of the adversary is comprised of two components, actions available to the adversary, and the information at his disposal [19]. These capabilities define the adversarial limitations, and determine the extent and impact of implementing different attack vectors in the threat landscape of a DRL agent. Behzadan et al. [49] investigate the availability of *a priori* information to the attacker (with respect to initiation of attacks) according to the type of available information. Accordingly, their work defines *agent information* as the adversary's available knowledge of the target model, its parameters, algorithm, and the reward function. Similarly, information about the dynamics and configuration of the environment comprise the *environment information*. The adversary may have varying degrees of access to either type of information. For instance, adversary may have complete knowledge of the environment's dynamics, but may only have access to partial or noisy observations of changes in the state of the environment. Also, the set of actions available to the attacker may include perturbations of the environment, the observation, the reward signal, experience memory, or actuators in a DRL setting, constrained by a budget that determines the permitted extent of actions. As a clarifying example, consider the case of crafting visual adversarial examples to perturb

observations. In such settings, the adversary may be limited to a maximum of B_p pixel modifications at each step. In the same example, the budget may also limit the frequency of perturbations to a maximum of one frame per every B_f samples.

We explore the the different attack vectors available to adversaries according to the targeted learning phase (training or testing), as detailed in the following:

3.2.1 Inference Phase

Similar to test-time attacks discussed in Section 2.2, inference-time attacks against DRL do not tamper the learned policy and experience data of the target. Such attacks can be active or passive. Passive attacks do not disturb the normal functioning of the target, but aim to infer the agent information via observation of state-inference pairs. On the other hand, active attacks are similar to evasion attacks, as both aim to manipulate the output of the model.

Depending on the information available to the adversary, attacks can take various forms. If the adversary has access to the agent information at a level sufficient for precise crafting of malicious input (e.g., adversarial examples) to the agent, then the attack is considered *whitebox*. Conversely, if such information is not available to the adversary, adversarial action is constrained to *blackbox* attacks. Similarly, if the agent has access to a degree of knowledge of the environment that is sufficient for direct exploitation in adversarial actions, the attack is of the *perfect information* type. For instance, knowledge of the dynamics of the Atari game Pong, as well as perfect observation of target's actions, enables the adversary to predict the future states and exploit this foresight in devising a sequence of perturbations to manipulate the DRL agent into performing poorly. On the other hand, if the adversary has complete knowledge of the game dynamics but can only have noisy or intermittent observability on the agent's actions, utilizing predictive control for adversarial actions may be rendered infeasible and the adversary will have to pursue attacks of type *imperfect information*.

3.2.2 Training Phase

The objective of attacks targeting the training phase is to extract, manipulate, or disable the policy learning mechanism of the DRL agent. Similar to those under the inference phase, extraction attacks are generally passive, and aim to extract the model parameters or contents of experience memory. A simple instance of extraction attacks at training phase is direct access to the training software or memory. Other techniques may include inference of parameters based on observations of the training process. Analogous to training-time attacks against supervised learners [46], manipulation or disruption of the training process can be achieved via two strategies: injection and modification. Injection refers to alteration of the experience memory via insertion of experience tuples that change the distribution of observations. A suitable technique for injection attacks is the manipulation of the environment itself. Modification attacks aim to change the contents of actual experiences. Adversarial example attacks against DRL at training time are a representative example of such attacks.

The taxonomy of the adversarial attacks enumerated in this section is presented in Figure (3.2).

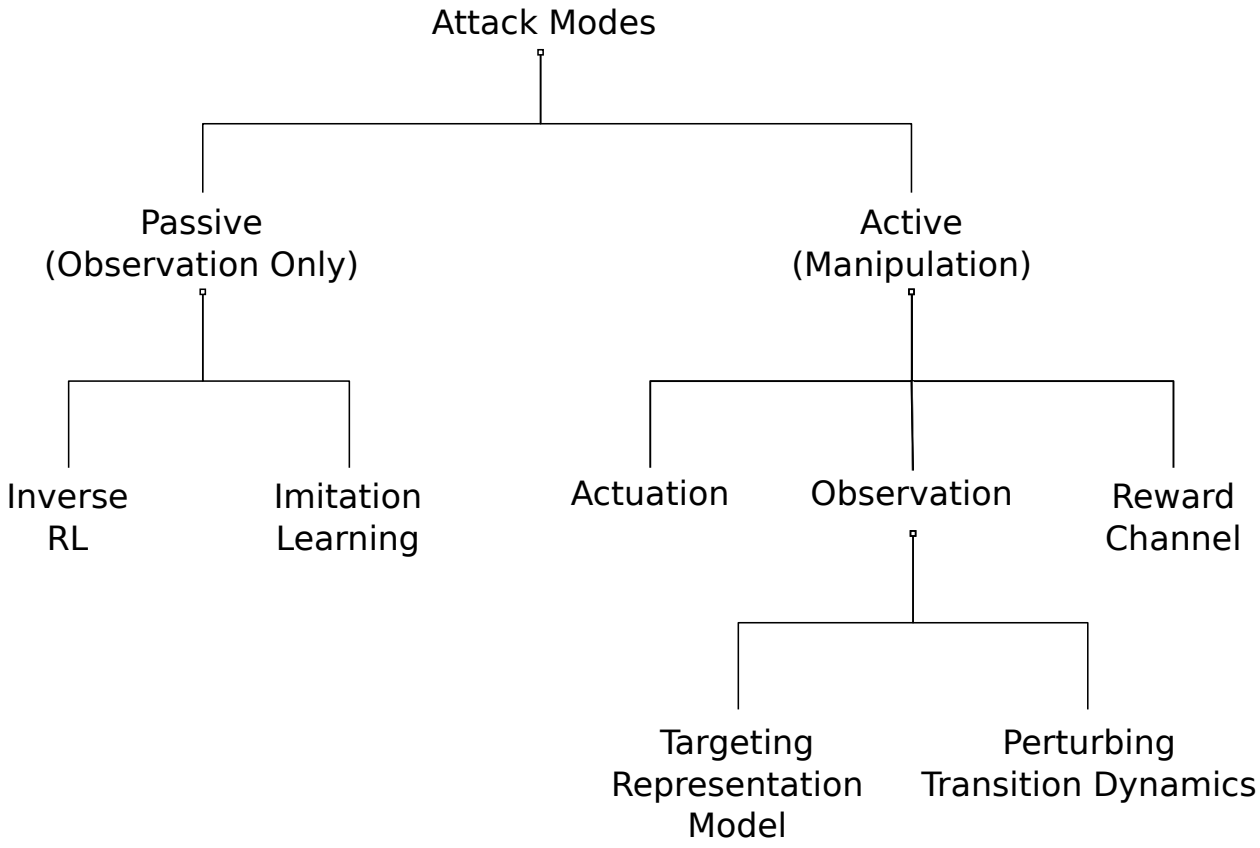

Figure 3.2: *Taxonomy of Attacks on DRL*

Chapter 4

State of the Art

[1] In this section, we present an overview of the prominent literature on the security of DRL. While this body of work is still at its early stages, the research on security of DRL has produced notable results which motivate and shed light on potential venues of further advancements. At a high-level, the studies on DRL security can be categorized in the two classes of offensive and defensive proposals. It is noteworthy that, with the exception of a few papers, current studies on offensive techniques are generally focused on either inference-time attacks or training-time attacks, but not both. Accordingly, we adopt this high-level classification as the foundation for our exploration of these literature. We begin with an overview of the offensive literature (as enumerated in Table 4.1), followed by a review of the literature on defensive techniques (as listed in Table 4.2). This review is prepared with the aim of extending current surveys [49][50] to recent developments and present a comprehensive survey of relevant literature. However, it must be noted that due to the fast rate of publications in machine learning, a number of very recent, yet notable papers may be missing from this review.

[1] This chapter extends and amends the previous work of the author [49].

Phase	Mode	Attack Surface		
		Observation	State	Reward
Test-Time	Whitebox	Behzadan & Munir [1], Huang et al. [22], Patthanaik et al. [55], Lin et al. [56], Tretschk et al. [59]	Han et al [57] , Clark et al. [58]	Han et al. [57]
	Blackbox	Behzadan & Munir [1], Huang et al. [22]	Behzadan & Munir [18]	Han et al. [57]
Training-Time	Whitebox	Kos & Song [60]	—	Han et al. [57]
	Blackbox	Behzadan & Munir [1]	—	—

Table 4.1: *Summary of Literature on Attack Mechanisms*

4.1 Test-time Attacks

Behzadan & Munir [1] published the first report on test-time vulnerabilities of DRL. Based on the fact that DQN policies and deep classifiers are essentially of the same structure and function, therefore DQN policies must also be vulnerable to adversarial examples. Accordingly, their paper tests the applicability of adversarial examples crafted with FGSM and JSMA versus DQN policies under whitebox settings, and demonstrate that such policies are also vulnerable to test-time manipulation using adversarial examples. In whitebox settings, the adversary performs a Man in The Middle (MITM) attack. He observes the state of the environment, and with complete and perfect knowledge of the targets policy parameters, crafts adversarial examples such that the observed state by the target $s'_t = s_t + \delta_t$ results in different state-action values $Q(s'_t, a_t) \neq Q(s_t, a_t)$, thus leading to the selection of an alternative action $a'_t = \arg\max_{a_t} Q(s'_t, a_t)$ instead of the original selection $a^*_t = \arg\max_{a_t} Q(s_t, a_t)$.

Furthermore, this paper demonstrates the practicality of blackbox test-time attacks utilizing the transferability of adversarial examples [41]. Their experiment follows an attack flow where the adversary first trains a DQN agent on the same environment based on the *known* reward function of the target, then follows the attack procedure of whitebox attacks, with the difference that adversarial examples are crafted for the dual policy trained for the adversary and applied to the target. To validate the feasibility of such attacks, the paper reports the transferability of adversarial examples crafted via FGSM and JSMA against DQN

policies, showing that more than 70% of such perturbations are transferable between the two models.

It is noteworthy that the work by Behzadan & Munir [1] and the proceeding literature on adversarial example attacks are instances of observation channel perturbations via exploiting the imperfections of learned representations in DRL models.

Following the report presented in [1], Huang et al. [22] analyzed the vulnerability of two other DRL algorithms to test-time attacks, namely TRPO and A3C. Leveraging the same attack process introduced by Behzadan & Munir [1] for whitebox settings, this paper reports that TRPO and A3C are also vulnerable to test-time adversarial example attacks crafted with FGSM. Also, the paper compares DQN policies trained on Atari games with those of TRPO and A3C, and demonstrates that policies trained with DQN are more susceptible to such attacks than TRPO and A3C. Furthermore, [22] analyzes the susceptibility of all three algorithms to blackbox attacks that exploit transferability. The corresponding experiments analyze two types of blackbox attacks: one in which the adversary has complete access to the environment information, and has knowledge of the target's training algorithm and hyperparameters, but not its random initialization; and one in which the adversary has no knowledge of the target's training algorithm or hyperparameters.

While the proofs of concept presented in previous studies are generally based on classical adversarial example crafting techniques (e.g., FGSM and JSMA), Patthanaik et al. [55] argue that the classical form of FGSM used in previous attacks on DRL do not use an optimal cost function in crafting DRL-specific adversarial examples. Consequently, this paper proposes two more effective whitebox approaches to computing such adversarial perturbations. The first approach is based on a novel cost function for attacks. The authors formally prove that if the optimal policy of an agent is given by the probability mass function $\pi^*(a|s)$, the objective function whose minimization leads to optimal adversarial attack on the agent is given by:

$$J(s, \pi^*) = -\sum_{i=1}^{n} p_i \log \pi_i^*, \tag{4.1}$$

where $\pi_i^* = \pi^*(a_i|s)$, $p_i = P(a_i)$; with P denoting the adversarial probability distribution. Accordingly, the paper [55] proposes an attack mechanism which solves this minimization problem via sample-based search. Alternatively, the second approach leverages Stochastic Gradient Descent (SGD) to replace the sample-based approach. Validation of these attacks is performed via experiments on attacking Double DQN and Radial Basis Function based Q-learning (RBFQ) [61] agents trained in the CartPole and Mountain Car environments. The reported results indicate that the attacks based on sample-based search perform consistently better than FGSM and the SGD-based approaches in degrading the performance of targeted agents. Another noteworthy observation in these results is that RBFQ agents behave with superior resilience to adversarial example attacks compared to DDQN.

The attack methodologies proposed in the aforementioned papers are all based on continuous and uniform perturbation of all observations by the adversary. However, Lin et al. [56] note that this type of attack may be both practically infeasible and easy to detect. Instead, they propose whitebox attacks that aim to minimize the number of required perturbations. Accordingly, two types of attacks are proposed in [56]: *strategically-timed attacks* and *enchanting attacks*

Strategically-timed attacks aim at perturbing the minimum subset of observations in an episode that results in the desired degradation or performance. This is achieved by identifying those states in which the difference between the value or preference of the agent's best and worst actions is greater than an arbitrary threshold defined by the adversary. At such states, the adversary implements adversarial examples to induce the selection of the least preferred action over the optimal one. The crafting algorithm used in [56] for generation of adversarial perturbations is that of Carlini and Wagner (C&W) [62].

On the other hand, *Enchanting* attacks aim to "lure" the target agent from a current state s_t at time t to a specified target state s_g in H steps. The proposed attack mechanism is an online planning algorithm, which utilizes generative modeling to predict the future states and a sampling-based cross-entropy method [63] to compute a minimum sequence of control actions that steers the targeted DRL agent towards the state s_g. The control actions of this attack are adversarial perturbations crafted via the C&W technique such that the

implemented action of the target agent is one that steers the agent closer to the state s_g. The experimental evaluation of these two attacks were performed on DQN and A3C agents trained on five Atari games.

The reported results demonstrate that for both types of agents, perturbing only 25% of observations via strategically-timed attacks can achieve the same levels of degradation as those resulting from uniform attacks. In accord with the findings of [22], these experiments also indicate that DQNs are more vulnerable than A3C agents against test-time adversarial example attacks. In the case of enchanting attacks, the results claim a 70% success rate in enchanting both types of agents (DQN and A3C) in three of the five games in less than $H = 40$ steps. The authors claim that failure in the remaining two games (Seaquest and ChopperCommand) was due to the existence of multiple random enemies that were not accurately modeled by the prediction models.

Similar to the enchanting mechanism of [56], Tretschk et al. [59] propose a whitebox attack mechanism that aims to maneuver the target agent to pursue an adversarial goal. Formally, the goal of this attack is to make a DRL agent trained for the original reward r^O to maximize an arbitrary adversarial reward r^A through a sequence of state perturbations. To this end, the authors develop a mechanism based on an Adversarial Transformer Network (ATN) [64], which is a freedforward deep neural network $g_\theta : X \to X$ that computes the adversarial perturbation to be added to the input of the target DRL agent. Considering DQNs as the target of this attack, the proposed mechanism of [59] is to learn g_θ by training in combination with a dual of the target DQN agent Q_ϕ. Accordingly, the model to be learned is $x \to Q_\phi(x + g_\theta(x))$ where the target's parameters ϕ are fixed and only the ATN parameters θ are learned. [59] claims that the generalizability of g_θ to unseen states allows the adversary to feed the input state through g_θ and then to the target DQN to achieve the desired outcome. Similar to the previously discussed attacks, this mechanism is also an MITM attack that assumes the adversary can manipulate the state before it is observed by the target. Furthermore, it requires complete knowledge of the agent information, as well as access to the target environment for training the ATN. The experimental results performed on the case of targeting a Pong-playing DQN agent demonstrate that the adversary can

successfully manipulate the agent into pursuing the adversarial policy at test-time, given a large-enough threshold for degree of perturbation.

The literature on test-time attacks on DRL that are not based on adversarial examples is still very scarce. For instance, Han et al. [57] investigate the case of a DRL agent in a Software Defined Network (SDN), tasked with preventing the propagation of a malware in the network by identifying the compromised nodes and deciding on taking one of the following actions at each time step: isolating and patching a node, reconnecting a node and its links, migrating the critical server, and taking no action. The reward value for this agent depends on whether the critical servers are compromised and the number of reachable nodes from such servers, as well as the number of compromised nodes, and the cost of migration. It is also assumed that the detection mechanism of the agent can be manipulated by the adversary (i.e., the adversary can induce False Positive (FP) or False Negative (FN) results in the detector), but is constrained by a threshold on how many such manipulations can be implemented at each time step. The test-time attacks proposed in [57] are two-fold: *indiscriminate attacks* aim to prevent the DRL agent from taking the optimal action a_t at time t, and *targeted attacks* aim to force the agent into taking a specific action a'_t at time t. Considering DDQN and A3C as DRL algorithms for the target agent, the objective for targeting DDQN agents is to maximize $Q(s_t + \delta_t, a'_t)$ for action a'_t at state s_t using perturbation δ_t. Similarly, the objective for targeting A3C is to maximize $\pi(a'_t|s_t + \delta_t)$ for the stochastic policy π. For these attacks, [57] develops a whitebox attack methodology, where the attacker can access the target's model. This attack requires the computation of those nodes whose FP or FN detection would facilitate the adversarial objective. Accordingly, [57] proposes an integer programming approach to deriving the set of such nodes at each time step. The authors also propose a blackbox attack technique, which is based on training surrogate models of the target with either the same or different hyperparameters and then following the procedure of the whitebox attacks. The experimental results produced in this paper indicate that in the majority of cases, both whitebox and blackbox attacks succeed in compromising the critical servers. It is also noted that there is no significant difference between the success rate of whitebox and blackbox attacks.

Clark et al. [58] demonstrate that the Q-learning policy of an autonomous navigation robot is susceptible to sensory manipulation. In this work, the ultrasonic collision avoidance of the robot was manipulated via artificial ultrasonic "pings" that would allow the attacker to manipulate the trajectory of the robot. Within the domain of autonomous navigation, Behzadan & Munir [18] propose an adversarial DRL agent specifically trained to manipulate the operating environment of an autonomously navigating DRL agent and induce collisions or trajectory manipulations by exploiting the collision avoidance policy of the target. In this attack, the adversarial DRL agent is trained as another autonomous navigation agent with a reward function that incorporates adversarial objectives, such as pursuing trajectories that will lead to the target colliding with itself or other objects in the environment. This attack is whitebox and requires access to the trained policy of the adversary, but not necessarily its parameters and hyperparameters used in its training.

4.2 Training-Time Attacks

The original paper of Behzadan & Munir [1] also demonstrates the vulnerability of DRL to training-time attacks. This paper investigates the feasibility of policy manipulation attacks against DQN agents leveraging adversarial examples. Accordingly, the authors develop the policy manipulation attack, the mechanism of which is illustrated in Figure 4.1. In this attack, the adversary aims at inducing an arbitrary policy π_{adv} on the target DQN at training time. This attack mechanism assumes a blackbox adversary, who does not have access to the parameters of the target θ_t at any time step t, but is aware of its reward function, training algorithm, and architecture. Furthermore, the adversary is assumed to have complete access to and knowledge of the training environment. The only parameter that the adversary can manipulate using this attack is the observed state of the environment, hence this adversary can be considered an MITM capable of perturbing the input stream from the environment to the target DQN agent.

At every iteration of the training process, a DQN agent following ϵ-greedy exploration

performs an action determined by the following mechanism:

$$a_t = \begin{cases} \text{random action}, & \text{with probability } \epsilon \\ \arg\max_a Q(s_t, a), & \text{with probability } 1 - \epsilon \end{cases} \tag{4.2}$$

Once the action is performed, the observation tuple $(s_t, a_t, r_{t+1}, s_{t+1})$ is stored in experience replay. At every p iterations, the agent samples a random minibatch of observations, and performs a gradient descent step on Equation 2.4 with respect to the parameters of the native Q-network. As discussed in Chapter 2, the DQN framework is comprised of two neural networks, one is the native Q-network and the other is the target network $\hat{Q}$ whose architecture and parameters are copies of the native network sampled once every c iterations.

Consequently, the attacker can manipulate the learning process of DQN by crafting states s_t such that $\hat{Q}(s_{t+1}, a; \theta_t^-)$ identifies an incorrect choice of optimal action at s_{t+1}. If the attacker is capable of crafting adversarial inputs s'_t and s'_{t+1} such that the value of Equation 2.4 is minimized for a specific action a', then the policy learned by DQN at this training step is optimized towards suggesting a' as the optimal action given the state s_t. In the proposed attack mechanism, the attacker observes interactions of its target with the environment $(s_{t-1}, a_{t-1}, r_t, s_t)$. If the resulting state s_t is not terminal (i.e., the episode does not end at that state), the attacker then calculates the perturbation vectors $\hat{\delta}_t$ for the current state s_t such that $max_{a'}\hat{Q}(s_t + \hat{\delta}_t, a'; \theta_t^-)$ causes $\hat{Q}$ to be maximum when $a' = \pi^*_{adv}(s_t)$, i.e., the maximum expected return is obtained when the action taken at s_t is determined by the attacker's policy. The attacker then reveals the perturbed state s'_t to the target, and retrains the replica based on the new state and action.

Considering that the attacker is not aware of the target's network architecture and its parameters at every time step, crafting adversarial states relies on a blackbox technique exploiting the transferability of adversarial examples by training a replica DQN agent and obtaining the state perturbations from the replica's Q' and $\hat{Q}'$ networks that correspond to the target's Q and $\hat{Q}$ networks, respectively.

Accordingly, Behzadan and Munir [1] divide this attack into the two phases of initializa-

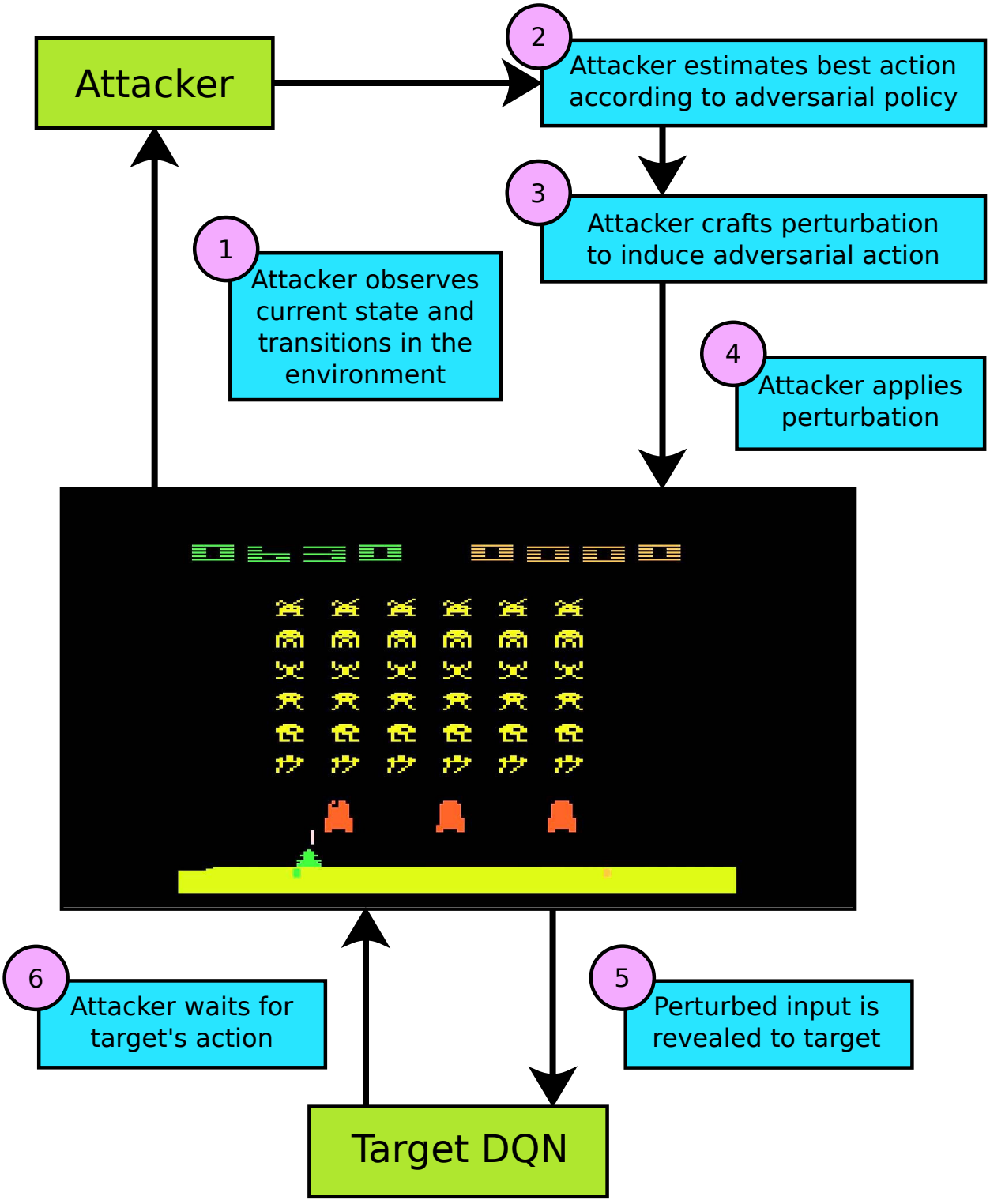

Figure 4.1: *Exploitation cycle of policy induction attack [1]*

tion and exploitation. The initialization phase implements processes that must be performed before the target begins interacting with the environment, which are:

1. Train a DQN based on attacker's reward function R' to obtain the adversarial policy π^*_{adv}
2. Create a replica of the target's DQN and initialize with random parameters

The exploitation phase implements the attack process and crafting adversarial inputs, such that the target DQN performs an action dictated by π^*_{adv}. This phase constitutes an attack cycle depicted in figure 4.1. The cycle initiates with the attacker's first observation of the environment, and runs in tandem with the target's operation. The authors report experimental verification of this attack against a Pong-playing DQN. In their experiment, the algorithm used to craft adversarial perturbations is JSMA. The results indicate that the adversary is capable of manipulating the agent towards an always-losing policy in almost the same number of training steps required to achieve optimal (i.e., best reported) performance.

In [60], Kos & Song present an experimental analysis of whitebox training-time attacks on DRL. Considering A3C agents training on Pong, the paper first demonstrates that in comparison to random perturbations, adversarial example attacks crafted via FGSM are significantly more effective in degrading the training performance of the agent. Then, the authors investigate the feasibility of non-contiguous attacks, in which not all the states of the environment are perturbed. To this end, three attack scenarios are studied: perturbing observations at every N frames (frequency-based), recomputing adversarial perturbation at every N frame and applying the last computed perturbation in the intermediate frames, and using the value function to estimate when to inject adversarial perturbations to be most effective. In the corresponding experiments, the attack is initiated after the agent has reached the optimal (i.e., baseline) performance. Presented results indicate that while the frequency-based attack fails to be particularly effective, recomputing at every tenth frame and reusing the previous perturbation in intermediate frames is almost as effective as the original attack. For the latter case, the paper develops an attack mechanism in which the

adversarial perturbations are injected only when the value function, computed over the original frame, is above a threshold. The authors present the rationale behind this method by noting that they only wish to disrupt the agent in crucial moments, when it is close to achieving a reward. The results demonstrate that this technique is far more effective than the previous two cases, and is argued to be more efficient than uniform perturbation of all frames.

Similar to the case of test-time attacks, the body of work on training-time attacks that are not based on adversarial examples is very thin. In the previously discussed paper by Han et al. [57], the target model is considered to be an online learner, and hence the authors investigate attacks that aim to manipulate the training phase of the target DRL. To this end, [57] presents a poisoning attack based on flipping the reward signs, with the goal of maximizing the loss function in the target DDQN agent. In this attack, once the target samples a batch of experiences for training, the adversary calculates the gradient of the loss function with respect to each of the observed reward signals, and flips the sign of experience with the largest absolute value of this gradient. In experimental validation of this attack, the authors impose limit of 5% on the maximum number of experiences that can be tampered at each training step. While the results demonstrate that this attack effectively degrades the training performance of the target, the authors note that this type of attack only delays the convergence as, given enough time, the agent still learns the optimal actions.

4.3 State of Defenses

Approach	Papers
Adversarial Training	Kos & Song [60], Pattanaik et al. [55], Behzadan & Munir [65]
Secure Exploration	Behzadan & Munir [66]
Predictive	Lin et al. [67]
Hierarchical RL	Havens et al. [68]
Game Theoretic	Ogunmolu et al. [69], Bravo & Mertikopoulos [70]

Table 4.2: *Summary of Literature on Defensive Techniques*

A major area of focus in mitigation of attacks on DRL is *adversarial training*. This approach was first used as a framework for evaluation of different algorithms. Littman [71] proposed an adversarial setting for Q-learning algorithms to fit into a multi-agent game, and training these agents to evaluate their performances. Recent research has shown that adversarial training can also be leveraged to enhance the robustness of RL agents. Recall from Chapter 2 that in RL, the typical objective of an agent is to maximize its expected long-term return R over possible trajectories τ, assuming a fixed transition model $P(s_t, a_t; \Phi)$ characterized by parameters Φ. That is,

$$R(\pi, P) = E_\tau[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)|s_0, \pi, P] \tag{4.3}$$

However, if there is variation in transition model, then criteria might be to perform well in expectation over all the possible transition models. Thus, leading to optimization of the mean performance of agent. The objective function in this scenario can be modified to

$$R(\pi) = E_P[R(\pi, P)] \tag{4.4}$$

This is commonly known as the *risk neutral* formulation. However, one underlying assumption in this formulation is that the distribution over transition model parameters is known a priori. It may not perform well over the transition model distributions because of high variance. Thus, conditional Value of Risk (CVaR), denoted by R_{RC}, can be used as optimization criteria for robust control [72]:

$$R_{RC}(\pi) = E_P[R(\pi, P)|\mathbb{P}(R(\pi, P) \leq \beta) = \alpha] \tag{4.5}$$

Hence, the problem boils down to maximizing the expected return over the worst α percentile of returns. Thereafter, for sampling these bad trajectories, Rajeswaran et al. [73] changed transition model parameters and sample trajectories by performing rollouts with different parameters. Morimoto and Doya [74] as well as Pinto et al. [75] adopted an indirect approach

where instead of sampling worst trajectories from rollout, an adversary is employed which attempts to lead the RL agent into undesired states. The adversary is trained by pursuing a reward function that is the negative of the RL agent's reward, thereby resulting in max-min game theoretic formulation. However, it is usually difficult to find this equilibrium [55].

Inspired by previous results in mitigation of adversarial example attacks against classifiers, the aforementioned paper by Kos & Song [60] experimentally explores the effectiveness of re-training DRL agents on adversarial perturbations in improving the resilience of agents. In their experiments, after the initial training in non-noisy environment, the agent is first allowed to re-train while an adversary injects either random noise or FGSM perturbations on each frame. Once the agent reaches good performance, the training is frozen and evaluated in a new environment with training-time attacks. The results presented in [60] demonstrate that re-trained agents can be resilient against certain levels of FGSM perturbation. Also, the paper reports that the re-trained agent is resilient against FGSM perturbations of greater or smaller magnitude than that of the perturbations used during re-training.

In the work of Pattanaik et al. [55], once the test-time attack techniques are developed, the authors investigate the effectiveness of adversarial re-training in test-time resiliency of DDQN and DDPG agents to adversarial perturbations. In response to the shortcomings of approaches proposed in [73] and [74], the proposed approach utilizes an adversary that fools the agent into sampling worst trajectories directly. In this approach, the algorithm is first trained in non-noisy environments. Then, the agent is retrained by training in a noisy environment in which an adversary constantly attacks the agent using the previously detailed gradient based attack. Presented results indicate that adversarial training of DDQN and DDPG agents enhances their resilience to test-time adversarial example attacks.

In [65], Behzadan & Munir investigate the test-time and training-time resilience of DQN agents trained under noncontiguous training-time attacks -that is, attacks that do not aim to perturb all observations. In this work, the attack mechanism follows that of [1], but instead of perturbing all state samples, the adversary either applies FGSM perturbation to each observation with a fixed probability $P(attack)$ or leaves it untouched. In experimental analysis of such attacks, the authors compare DQN agents training on the Pong and Breakout

environments. The attacks were initiated at or close to the convergence of mean return towards the optimal (i.e., baseline) value. The results indicate that in both environments, DQN agents are able to recover from noncontiguous attacks with attack probabilities $p = 0.2$ and $p = 0.4$ and converge to optimal performance, while they fail to recover under attacks with $p = 0.8$ and $p = 1.0$ (contiguous attack). It is observed that for the agents that recover, the training performance deteriorates almost uniformly until a minimum point is reached, from which onward the agent begins to recover and adjust the policy towards optimal performance. The authors' interpretation of this behavior is based on the statistics of experience replay: for the agent to recover from adversarial perturbations, the number of interactions with the perturbed environment must reach a critical threshold, so that the randomly sampled batches from the experience memory can represent the statistical significance of perturbations. Furthermore, the test-time resilience of these adversarially trained agents is also studied under the worst-case test-time attack scenario of $p = 1.0$. The results demonstrate that under test-time attacks, the agents that manage to recover during adversarial training perform almost as well as the unperturbed agents.

Another venue of research on mitigation techniques is focused on secure exploration mechanisms. It must be noted that secure exploration differs from research on safe exploration [17]; the latter considers accidental and harmful actions that may arise during exploration of RL agents, while the former is interested in exploration mechanisms that enhance or preserve the security of DRL agents against intentional adversarial attacks. For instance, [66] presents a comparative study of resilience to adversarial example attacks between two DQN agents, one adopting the ϵ-greedy exploration mechanism, and the other implementing a parameter-space noise exploration technique known as NoisyNet [24]. Contrary to classical exploration heuristics such as ϵ-greedy [3], parameter-space noise is iteratively and adaptively applied to the parameters of the learning model, such as weights of the neural network. The results for the NoisyNet implementation of this paradigm [24] demonstrate that the addition of adaptive noise to the parameters of deep RL architectures greatly enhances the exploration behavior and convergence speed of such algorithms. Accordingly, the authors in [66] hypothesize that the randomness introduced via parameter noise, not only enhances the discovery of more

creative and robust policies, but also reduces the effect of whitebox and blackbox adversarial example attacks at both test-time and training-time.

To test the validity of this hypothesis, [66] presents an evaluation of the test-time and training-time resiliency of DQN agents based on both NoisyNet and ϵ-greedy in three Atari game environments: Enduro, Assault, and Blackout. Under test-time attacks, the results demonstrate that while both models are susceptible of FGSM perturbations, NoisyNet DQNs are more resilient to such attacks than those based on ϵ-greedy. Furthermore, comparison of performance under blackbox attacks demonstrates significant improvements in Noisynets, as observed in all three cases.

In experiments on training-time attacks, while both types of agents are shown to be subject to deterioration as a result of the attack based on the blackbox mechanism of Behzadan & Munir [1], NoisyNet agents demonstrate significantly stronger resilience to such attacks than ϵ-greedy agents. The authors argue that this is due to the enhanced generalization and reduced transferability in NoisyNet models.

In [67], Liu et al. propose a defense mechanism to defend RL agent from test-time white-box adversarial attacks by leveraging the temporal coherence of multiple observations in sequential decision-making tasks. To this end, a *visual forsight* module is trained to predict the current observation based on past observations and actions. Accordingly, at time step t, the action-conditioned observation prediction model G_{θ_g} takes m previous observations $x_{t-m:t-1}$ and corresponding m actions $a_{t-m:t-1}$ as input to predict the current observation $\hat{x_t}$. Given a normal observation x_t^{normal} at the current time step t, the action distribution that the agent uses to sample an action from is $\pi_{\theta_\pi}(x_t^{normal})$, which should be similar to the action distribution of $\pi_{\theta_\pi}(\hat{x_t})$ from the predicted observation. On the other hand, if the current input is adversarially perturbed, that is the agent observed x_t^{adv} instead of x_t^{normal}, then the resulting action distribution $\pi_{\theta_\pi}(x_t^{adv})$ should differ from $\pi_{\theta_\pi}(\hat{x_t})$ because the goal of the adversary is to perturb the input observation x_t to cause the agent to take a different action. Therefore, the similarity between two action distributions can be used to detect the presence of adversarial attacks. To validate this claim, the paper presents an experiment on DQN agents trained on five Atari games (Pong, Seaquest, Freeway, ChopperCommand,

and MsPacman). Also, the experiment applies three types of adversarial example crafting algorithms, namely: FGSM, Basic Iterative Method [76], and C&W. The presented results indicate that the proposed method is able to detect 60% to 100% of adversarial example attacks against all DQN agents, and is shown to have superior detection performance to adversarial example detection techniques developed for deep classifiers, namely Feature Squeezer [77], AutoEncoder [78], and Dropout [79].

During policy learning, information perturbation can be generally viewed as a bias that can prevent the agent from effectively learning the desired policy. Inspired by this idea, Havens et al. [68] propose a hierarchical meta-learning framework, named Meta-Learned Advantage Hierarchy (MLAH). Their work considers a policy learning problem where there are periods of adversarial attacks that corrupt state observations during the continuous learning of the agent, and aims at the online mitigation of the bias introduced by the attack into the nominal policy. The proposed MLAH algorithm is based on the assumption that DRL agents learn sub-policies (i.e., skills) en route to learning the ultimate task. Given that the agent has developed accurate expectations of its sub-policies, if the underlying task were to change at anytime, the agent may notice that the result of its action has changed with respect to what was expected. In an RL framework, comparing the expected return of a state to the observed return of some action is typically known as the *advantage*. Accordingly, MLAH uses the estimated of advantage as a measure of underlying changes in a task, and leverages this metric to switch from one sub-policy to another more appropriate sub-policy. Consequntly, even if the adversary could compute a series of likely states to fool an MLAH-based agent, the advantage would still be affected and a master agent may detect the attack. The adversary would have to consecutively fool the agent with a state that would be expected to give an equally bad reward as that of the manipulated state. The authors claim that this constraint would make the perturbation especially hard or infeasible to compute. Experimental results presented in the paper demonstrate that for PPO agents, MLAH-based agents demonstrate superior robustness and resilience to noncontiguous adversarial example attacks at training-time.

Another area of research is that of approaches based on game-theoretic modeling. A

well-known instance of such approaches is in the field of multi-agent reinforcement learning, where agents are engaged in zero-sum games and utilize manipulation and misinformation to beat the other agents and maximize individual rewards [53]. More recently, Ogunmolu et al. [69] present such an approach by modeling the adversarial interaction between a DRL agent and a training-time adversary as a minimax iterative dynamic game, and present a meta-algorithm for controlling the training process and steering it towards saddle-point equilibria. In [70], Bravo & Mertikopoulos formulate the problem of corrupt reward channel (defined in Chapter 3) as a zero-sum evolutionary game between the RL agent and the adversary, and formally analyze the Nash equilibria in such settings.

4.4 Benchmarks and Metrics

As discussed in the previous sections, the majority of current literature on secure RL utilize experimental analysis for validation of their proposals. Another observation from this review is that while one particular problem may be approached by various research efforts, lack of consistent metrics renders the quantitative comparison of their proposals and results difficult. This section aims to provide an overview of the simulation benchmarks and evaluation metrics that are used in the current literature with the goals of facilitating further research and providing the grounds for a more consistent body of work in the future.

4.4.1 Simulation Benchmarks

Similar to the general research on DRL, many of the reviewed papers on secure DRL base their experimental analysis on Atari games and similar arcade-like environments provided within the OpenAI Gym platform [80]. Gym provides an RL-friendly interface with a variety of benchmarks for RL research, including the Arcade Learning Environment [81], RLLab benchmark for continuous control [82], and many more. The Gym interface provides a seamless platform for the integration of RL agents with the simulation environment. This interface provides the agent with access to the state information (e.g., game frames, score,

etc.), game controls, and the progression speed of the environment (e.g., waiting for training step to complete before progressing to the next step). While seldom referenced in the secure DRL literature, OpenAI has introduced two more advanced platforms to Gym, OpenAI Universe [80] and OpenAI Retro[2], which provide access to more complex environments, as well as enhanced implementations for benchmarking and recording the experiment.

With regards to implementations of adversarial example attacks, Cleverhans [83] is the most popular choice in the current research. This library provides standardized reference implementations of adversarial example construction techniques and adversarial training. While originally developed on the Tensorflow [84] stack, the interface to Cleverhans is designed to accept models implemented using any model framework (such as Keras [85] and PyTorch [86]). To facilitate the utilization of this library for experiments on DRL, Behzadan & Munir have developed RLAttack [87] as an interface between DRL agents implemented in Tensorflow and the adversarial example techniques in Cleverhans. The current version of this tool is compatible with all DRL algorithms available in OpenAI Baselines [88], and supports training-time and test-time attacks, contiguous and noncontiguous attacks, and both blackbox and whitebox attacks on DRL agents.

Another benchmark used by current secure DRL research is DeepMind's AI Safety Gridworlds [89]. This benchmark provides simple environments based on the classic Gridworld settings for experiments on RL safety issues that include safe interruptibility, avoidance of side effects, reward hacking, safe exploration, and robustness to adversaries.

4.4.2 Metrics

Evaluation metrics utilized in the current literature of DRL are generally ad hoc and non-generalizable. For adversarial example attacks, the robustness of an agent is often measured by the minimum value of perturbation threshold ϵ that results in successful attacks. In studies on both training-time and test-time attacks, a popular metric is the number of steps (e.g., training epochs, episodes, iterations) required to achieve an adversarial objective. Similarly,

[2]https://github.com/openai/retro

in blackbox attacks, percentage of transferable adversarial examples between models is an often-used metric as a measure of susceptibility. In all studies, the common metric of success for adversarial attacks is (mean) return over episodes or epochs, which demonstrates the feasibility and effectiveness of attacks with respect to computational cost and time.

Part II

State-Space Attacks on DRL

Chapter 5

Adversarial Resilience and Robustness of DRL Policies

Since the reports by Behzadan & Munir [1] and Huang et al. [22], the primary emphasis of the state of the art in DRL security [49] has been on the vulnerability of policies to state-space perturbations. In particular, the manipulation of the policy via adversarial examples [39] has remained the main focus of current literature on this issue. However, this bias towards adversarial example attacks gives rise to a critical shortcoming: the analyses of such attacks fail to disentangle the vulnerability caused by the learned representation and that which is due to the sensitivity of the DRL dynamics to distributional shifts in state transitions. Also, the performance of defenses proposed for adversarial example attacks are inherently limited to the considered attack mechanisms. As the most successful technique for mitigation of adversarial examples, adversarial training is known to enhance the robustness of machine learning models to the type of attack used for generating the training adversarial examples, while leaving the model vulnerable to other types of attacks [23]. Furthermore, the current literature fails to provide solutions and approaches which can be used in practice to evaluate and improve the robustness and resilience of DRL policies to attacks that exploit the sensitivity to state transitions. Also, there remains a need for quantitative approaches to measure and benchmark the resilience and robustness of DRL policies in a reusable and generalizable

manner.

In response to these shortcomings, this chapter aims to address the problem of quantifying and benchmarking the robustness and resilience of a DRL agent to adversarial perturbations of state transitions at test-time, in a manner that is independent of the attack type. This improves the generalization of current techniques that analyze the model against specific adversarial example attacks. Accordingly, the main contributions of this chapter are as follows:

1. We present formulations of the resilience and robustness problems that enable the disentanglement of limitation in representation learning from sensitivity of policies to state transition dynamics.

2. We propose two RL-based techniques and corresponding metrics for the measurement and benchmarking of resilience and robustness of DRL policies to perturbations of state transitions,

3. We demonstrate the feasibility of our proposal through experimental evaluation of their performance on DQN, A2C, and PPO2 agents trained in the Cartpole environment.

This chapter is organized as follows: Section 5.1 defines and formulates the problems of adversarial resilience and robustness in DRL. Our proposed methods for benchmarking the test-time resilience and robustness of DRL policies are presented in Sections 5.2 and 5.3. Section 5.4 provides the details of experimental setup for evaluating the performance of our proposals, with the corresponding results presented in Section 5.5. Section 5.6 concludes this chapter with a summary of findings and remarks on future directions of research.

5.1 Problem Formulation

We consider the the generic problem of RL in the settings of a Markov Decision Process (MDP), described by the tuple $MDP :=< \mathbb{S}, \mathbb{A}, \mathbb{R}, \mathbb{P} >$, where $\mathbb{S}$ is the set of reachable states in the process, $\mathbb{A}$ is the set of available actions, $\mathbb{R}$ is the mapping of transitions to

the immediate reward, and $\mathbb{P}$ represents the transition probabilities (i.e., state dynamics), which are initially unknown to RL agents. At any given time-step t, the MDP is at a state $s_t \in \mathbb{S}$. The RL agent's choice of action at time t, $a_t \in \mathbb{A}$ causes a transition from s_t to a state s_{t+1} according to the transition probability $P(s_{t+1}|s_t, a_t)$. The agent receives a reward $r_{t+1} = R(s_t, a_t, s_{t+1})$ for choosing the action a_t at state s_t. Interactions of the agent with MDP are determined by the policy π. When such interactions are deterministic, the policy $\pi : S \rightarrow \mathbb{A}$ is a mapping between the states and their corresponding actions. A stochastic policy $\pi(s)$ represents the probability distribution of implementing any action $a \in \mathbb{A}$ at state s. The goal of RL is to learn a policy that maximizes the expected discounted return $E[R_t]$, where $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$; with r_t denoting the instantaneous reward received at time t, and γ is a discount factor $\gamma \in [0, 1]$.

To facilitate the formal statement of adversarial resilience and robustness, we first introduce the following definitions:

- *Adversarial Regret* at time T is the difference between return obtained by the nominal (unperturbed) agent at time T and the return obtained by the perturbed agent at time T. Formally: $\hat{R}_{adv}(T) = R_{nominal}(T) - R_{perturbed}(T)$. The time T may represent either the terminal time step of an episode, or the time-horizon of interest in the analysis.

- *Adversarial Budget* is defined by one or more of the following parameters: the maximum number of features that can be perturbed in the observations ($O_{max} \in [0, \infty]$), the maximum number of observations that can be perturbed ($N_{max} \in [0, \infty]$), and the probability of perturbing each observation ($P(perturb) \in [0, 1]$).

Building on these two concepts, we define the problems of adversarial resilience and robustness as follows:

1. ***Test-Time Resilience:*** The minimum number of state perturbations required to incur the maximum reduction to the total return at time T (denoted by $\hat{R}_{adv}(T)$) for an agent driven by a policy $\pi(s)$ in an environment with transition dynamics $\mathbb{P}$.

2. ***Test-Time Robustness:*** The maximum adversarial regret $\hat{R}_{adv}(T) = \epsilon_{max}$ achievable via a maximum of δ_{max} state perturbations for an agent driven by a policy $\pi(s)$ in an environment with transition dynamics $\mathbb{P}$.

The following sections provide the details of our proposed solutions to each of the aforementioned problem settings.

5.2 Benchmarking of Test-Time Resilience

This problem can be modeled as that of finding an optimal adversarial policy $\pi_{adv}(s)$ that minimizes the cost incurred to the adversary C_{adv} in order to impose the maximum adversarial regret $\hat{R}_{adv}(T)$, the worst-case value of which is the highest cumulative reward achieved by the target policy R_{max}. Our proposed approach is through the formulation of this problem in the settings of reinforcement learning. The state space in the corresponding MDP is the set of states in the target MDP, augmented with the action of the target in that state, i.e., $S' = \{\forall s \in \mathbb{S} : (s, \pi(s))\}$. For the purpose of measuring a lower bound for the resilience, we consider the worst-case white-box adversary, which is able to impose targeted state perturbations with a 100% success rate, to induce any action within the permissible action-set of the target $\mathbb{A}$ which has the lowest Q-value at any state s according to the target's optimal state-action value function Q^*. In this case, the set of permissible adversarial actions at any state s is given by:

$$A_{adv}(s) = \{\text{No Action}\} \cup \mathbb{A} \setminus \pi^*(s) \tag{5.1}$$

where $\mathbb{A}$ is the action set of the targeted agent, and $\pi : S \rightarrow A$ is the policy of the targeted agent. In the proposed approach, the adversarial reward value is determined via the procedure detailed in Algorithm 5:

Algorithm 5 Reward Assignment of RL Agent for Measuring Adversarial Resilience

Require: Target policy π^*, Perturbation cost function $c_{adv}(.,.)$, Maximum achievable score R_{max}, Optimal state-action value function $Q^*(.,.)$, Current adversarial policy π^{adv}, Current state s_t, Current count of adversarial actions $AdvCount$, Current score R_t

Set ToPerturb $\leftarrow \pi^{adv}(s_t)$

if ToPerturb is False **then**

$a_t \leftarrow \pi^*(s_t)$

$Reward \leftarrow 0$

else

$a'_t \leftarrow \arg\min_a Q^*(s_t, a)$

$Reward \leftarrow -c_{adv}(s_t, a'_t)$

end if

if either s_t or s'_t is terminal **then**

$Reward+ = (R_{max} - R_t)$

end if

where $c(s_t, a'_t)$ is the cost of imposing the state perturbation which induces the adversarial action a'_t at state s_t. It is noteworthy that if the value of $c(s_t, a'_t)$ is invariant with respect to a'_t, the adversarial action set reduces to:

$$A_{adv}(s) = \{\text{No Action}, \mathit{Induce} \arg\min_a Q(s, a)\} \tag{5.2}$$

To obtain the test-time resilience of policy π^* to state perturbations, we propose the following procedure:

1. If the state-action value function of the target Q^* is not available (i.e., black-box testing), approximate Q^* via policy imitation from the policy using Algorithm 8 (Section 8.2).

2. Train the adversarial agent against the target following π in its training environment,

report the optimal adversarial return $R^*_{perturbed}$ and the maximum adversarial regret $R^*_{adv}(T)$.

3. Apply the adversarial policy against the target in N episodes, record total cost C_{adv} for each episode,

4. Report the average of C_{adv} over N episodes as the mean test-time resilience of π in the given environment.

This procedure introduces 3 metrics for the quantification of test-time resilience: the optimal adversarial return $R^*_{perturbed}$ achieved in the training process of the adversarial policy, the maximum adversarial regret $R^*_{adv}(T)$ achieved during training, and the mean per-episode of the total cost C_{adv}. These metrics provide the means to benchmark and compare the test-time resilience of different policies trained to optimize the agent's performance in a given environment.

For the purpose of measuring resilience, we consider convergence to be reached if the average adversarial regret over 200 episodes remains constant. This definition relaxes the instabilities that may arise due to the configuration and architecture of the DRL training process. It is noteworthy that depending on the training algorithm and design parameters, this procedure is not guaranteed to converge to global optima. However, by reporting the number of iterations and configuration of random number generators with a constant seed, the reported results present a reproducible loose lower bound on the adversarial resilience of the target. Also, the trained adversarial policy can be used to test other policies for comparison of such lower bounds under the same adversarial strategy.

5.3 Benchmarking of Test-Time Robustness

For this problem, we propose a modified version of the procedure developed for benchmarking the test-time resilience. Accordingly, the reward function is adjusted to account for the lack of a target ϵ, as well as the addition of an adversarial budget constraint δ_{max}. The reward measurement of this process is outlined in Algorithm 6:

Algorithm 6 Reward Assignment of RL Agent for Measuring Adversarial Robustness

Require: Maximum perturbation budget δ_{max}, Perturbation cost function $c_{adv}(.,.)$, Maximum achievable score R_{max}, Optimal state-action value function $Q^*(.,.)$, Current adversarial policy π^{adv}, Current state s, Current count of adversarial actions $AdvCount$, Current score R_t

Set AdversarialAction $\leftarrow \pi^{adv}(s)$

if AdversarialAction is NoAction **then**

$Reward \leftarrow 0$

else if $AdvCount \geq \delta_{max}$ **then**

$Reward \leftarrow -c_{adv}(s, AdversarialAction) \times \delta_{max}$

$AdvCount+ = 1$

else

$Reward \leftarrow -c_adv(s, AdversarialAction)$

$AdvCount+ = 1$

end if

if s is terminal **then**

$Reward+ = 1.0 * (R_{max} - R_t)$

$AdvCount \leftarrow 0$

end if

The proposed procedure for measuring the test-time robustness of a given DRL policy to adversarial state perturbations is as follows:

1. If the state-action value function of the target Q^* is not available (i.e., black-box testing settings), approximate Q^* from the policy using imitation learning (e.g., Algorithm 8 – Section 8.2)

2. Train the adversarial agent against the target policy π^* in its training environment, report the maximum adversarial regret $R^*_{adv}(T)$ for time T achieved at adversarial optimality,

3. Apply the adversarial policy against the target for N episodes, record the adversarial regret at the end of each episode $R_{adv}(T)$,

4. Report the average of $R_{adv}(T)$ over N episodes as the mean per-episode test-time robustness of π^* in the given environment.

5.4 Experiment Setup

Environment and Target Policies: To demonstrate the performance of the proposed procedures for benchmarking the test-time robustness and resilience in DRL policies, we present the analysis of the aforementioned measurements for policies trained in the CartPole environment in OpenAI Gym [80]. The considered policies are chosen to represent the commonly-adopted state of the art method from each class of DRL algorithms. From the class of value-iteration approaches, we consider DQN with prioritized replay. From policy gradient approaches, we consider PPO2. As for actor-critic methods, we investigate the A2C method. Table 5.4 presents the specifications of the CartPole environment, and Tables 5.4 – 5.4 provide the parameter settings of each target policy.

Table 5.1: *Specifications of the CartPole Environment*

Observation Space	Cart Position [-4.8, +4.8] Cart Velocity [-inf, +inf] Pole Angle [-24 deg, +24 deg] Pole Velocity at Tip [-inf, +inf]
Action Space	0 : Push cart to the left 1 : Push cart to the right
Reward	+1 for every step taken
Termination	Pole Angle is more than 12 degrees Cart Position is more than 2.4 Episode length is greater than 500

Table 5.2: *Parameters of DQN Policy*

No. Time steps	10^5
γ	0.99
Learning Rate	10^{-3}
Replay Buffer Size	50000
First Learning Step	1000
Target Network Update Freq.	500
Prioritized Replay	True
Exploration	Parameter-Space Noise
Exploration Fraction	0.1
Final Exploration Prob.	0.02
Max. Total Reward	500

Table 5.3: *Parameters of A2C Policy*

No. Time steps	5×10^5
γ	0.99
Learning Rate	7×10^{-4}
Entropy Coefficient	0.0
Value Function Coefficient	0.25
Max. Total Reward	500

Table 5.4: *Parameters of A2C Policy*

No. Environments	8
No. Time steps	10^6
No. Runs per Environment per Update	2048
No. Minibatches per update	32
Bias-Variance Trade-Off Factor	0.95
No. Surrogate Epochs	10
γ	0.99
Learning Rate	3×10^{-4}
Entropy Coefficient	0.0
Value Function Coefficient	0.5
Max. Total Reward	500

Adversarial Agent: In these experiments, the adversarial agent is a DQN agent with the hyperparameters provided in Table 5.4. We consider a homogeneous perturbation cost function for all state perturbations, that is, $\forall s, a' : c_{adv}(s, a') = c_{adv}$. For both the resilience and robustness measurements, we set $c_{adv} = 1$ (i.e., each perturbation incurs a cost of 1 to the adversary). The training process is terminated when the adversarial regret is maximized and the 100-episode average of the number of adversarial perturbations is quasi-stable for 200 episodes.

Table 5.5: *Parameters of DQN Policy*

Max. Time steps	10^5
γ	0.99
Learning Rate	10^{-3}
Replay Buffer Size	50000
First Learning Step	1000
Target Network Update Freq.	500
Experience Selection	Prioritized Replay
Exploration	Parameter-Space Noise
Exploration Fraction	0.1
Final Exploration Prob.	0.02

5.5 Results

5.5.1 Resilience Benchmarks

We consider the white-box settings in the training of adversarial agents for resilience measurement. For the DQN target, the optimal state-action value function Q^* of the target is directly utilized. As for the A2C and PPO2 targets, the state-action value function is calculated from the internally-available state value estimations $V*(s)$ according to the following transformation:

$$Q^*(s_t, a) = r(s_t, a) + \gamma V^*(s_{t+1}) \tag{5.3}$$

where s_{t+1} is the state resulting from a transition out of state s_t by implementing action a.

The training progress plots of adversarial DQN policy on the three target policies are presented in Fig.5.1–5.3. It can be seen that all three policies converge to the same optima. However, for the adversary targeting the DQN policies, the convergence is achieved at a higher number of training steps.

It is noteworthy that for all three policies, the mean-per-100 episodes of the minimum number of perturbations at convergence is almost similar (as reported in Table 5.5.1), with A2C having the largest value of 7.69 perturbations, PPO2 at a value of 7.49 perturbations, and DQN having the lowest value of 7.13. Also, the test-time performance of these trained policies indicate similar results, with DQN requiring 6.95 perturbations to incur an adversarial regret of 491.15, PPO2 requiring 7.72 perturbations for an adversarial regret of 490.47, and A2C requiring 8.71 perturbations for an adversarial regret of 488.16. Accordingly, we can interpret these results as follows: the DQN policy has the lowest adversarial resilience among the three, followed by the PPO2 policy. Within the context of this comparison, the A2C policy is found to be the most resilient to state-space perturbation attacks.

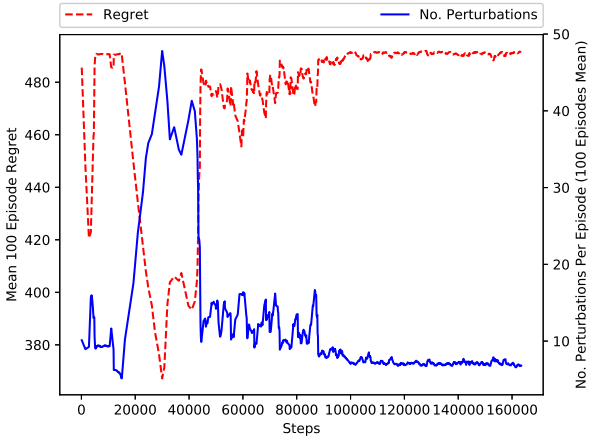

Figure 5.1: *Adversarial Training Progress for Resilience Benchmarking of the DQN Policy*

To investigate the state-transition vulnerability of each policy, we also study the frequency of perturbing states at each time step of an episode for the three adversarial policies. The results, presented in Fig. 5.4 – 5.6 illustrate that in all three policies, the initial time steps have been the subject of most perturbations. This result is noteworthy, as it contradicts the assumption of Lin et al. [56] that the most effective adversarial perturbations are those that are mounted towards the terminal state of the environment.

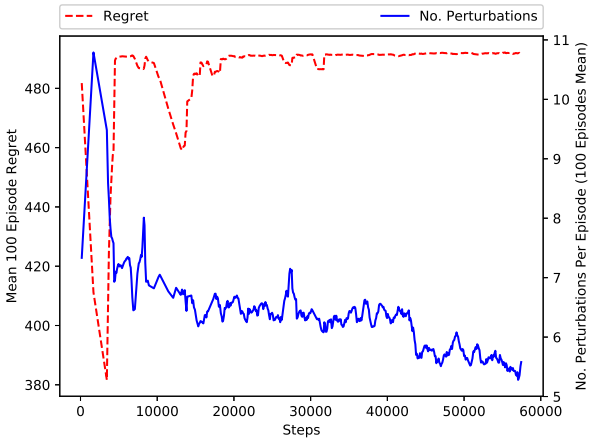

Figure 5.2: *Adversarial Training Progress for Resilience Benchmarking of the A2C Policy*

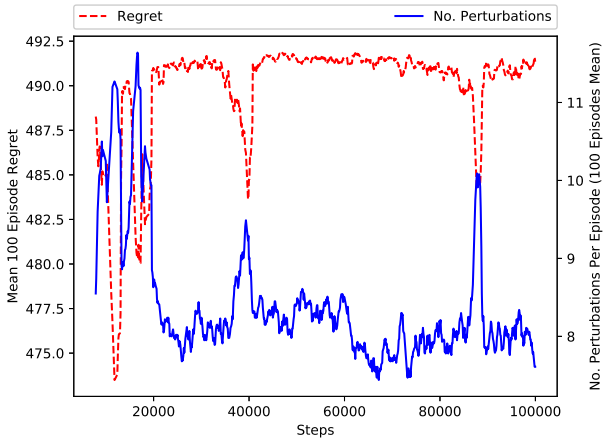

Figure 5.3: *Adversarial Training Progress for Resilience Benchmarking of the PPO2 Policy*

Table 5.6: *Comparison of Test-Time and Training-Time Resilience Measurements*

Target Policy	Max. Regret	Avg. Regret (Training)	Avg. No. Perturbations (Training)	Avg. Regret	Avg. No. Perturbations
DQN	492	491.24	7.13	491.15	6.95
A2C	492	491.44	7.69	488.16	8.71
PPO2	492	491.72	7.49	490.47	7.72

5.5.2 Robustness Benchmarks

To demonstrate the performance of our proposed technique for benchmarking the robustness of DRL policies, we provide the training-time results for two cases of $\delta_{max} = 10$ and $\delta_{max} = 5$ for DQN, A2C, and PPO2 Policies. As illustrated in Fig.5.7 – 5.9, all three adversarial

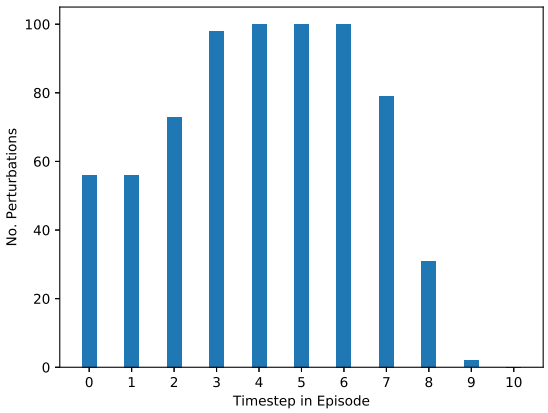

Figure 5.4: *Perturbation Count Per Episodic Time Step in 100 Runs Targeting DQN Policy*

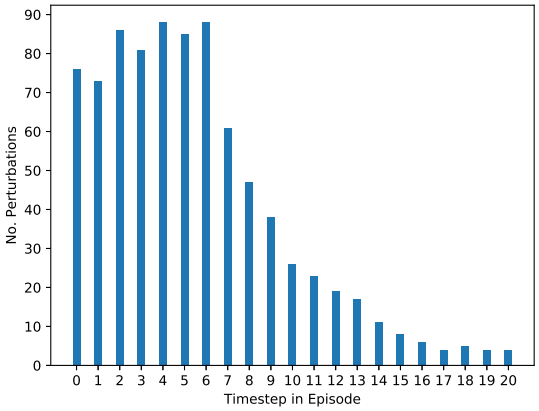

Figure 5.5: *Perturbation Count Per Episodic Time Step in 100 Runs Targeting A2C Policy*

policies converge with similar minimum perturbation counts as those obtained in resilience analysis. This is expected, as the resilience analysis established that the minimum number of actions required for maximum regret is 7.5, which is less than the available budget of $\delta_{max} = 10$ As for the case of $\delta_{max} = 5$, Fig.5.10 – 5.12 demonstrate significant differences between the three policies. In Fig.5.10, it can be seen that at 5 actions, the convergence occurs with an adversarial regret of 462.5, while for A2C, the best 5-action indication of convergence occurs at an adversarial regret of 224. As for PPO2, this value is at 268.2. These results indicate a similar ranking of the robustness in these policies, with DQN being the least-robust to maximum of 5 perturbations, and the A2C prevailing as the most robust

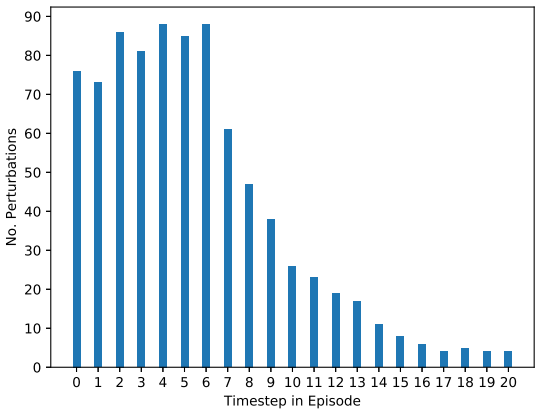

Figure 5.6: *Perturbation Count Per Episodic Time Step in 100 Runs Targeting PPO Policy*

policy to maximum of 5 perturbations.

Case 1: $\delta_{max} = 10$**:**

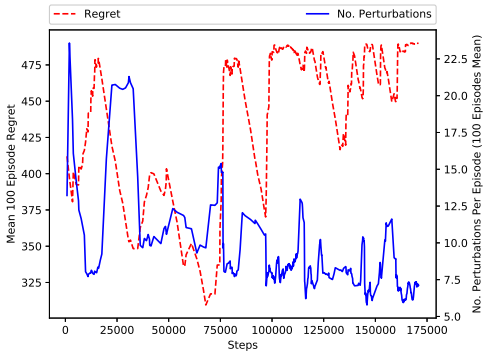

Figure 5.7: *Adversarial Training Progress for Robustness Benchmarking - DQN,* $\delta_{max} = 10$

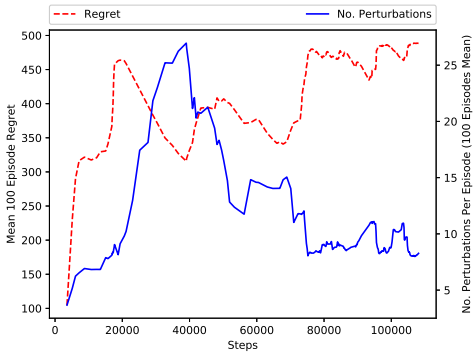

Figure 5.8: *Adversarial Training Progress for Robustness Benchmarking - A2C,* $\delta_{max} = 10$

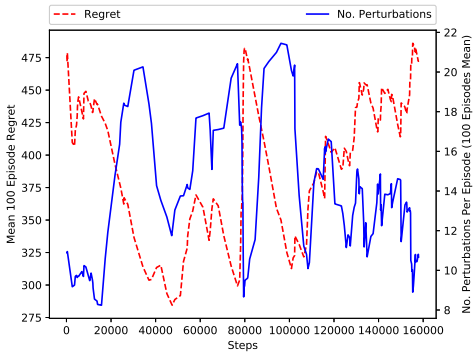

Figure 5.9: *Adversarial Training Progress for Robustness Benchmarking - PPO2,* $\delta_{max} = 10$

Case 2: $\delta_{max} = 5$**:**

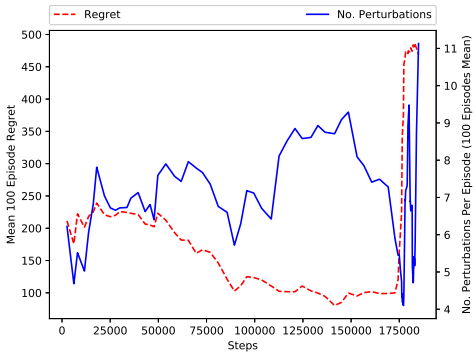

Figure 5.10: *Adversarial Training Progress for Robustness Benchmarking - DQN,* $\delta_{max} = 5$

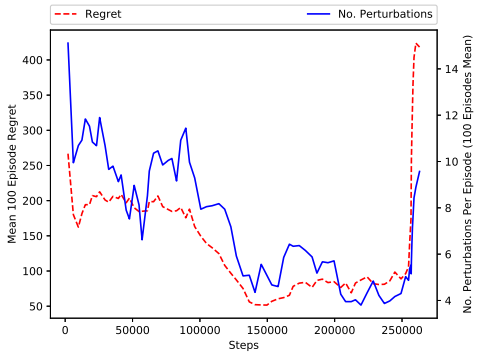

Figure 5.11: *Adversarial Training Progress for Robustness Benchmarking - A2C,* $\delta_{max} = 5$

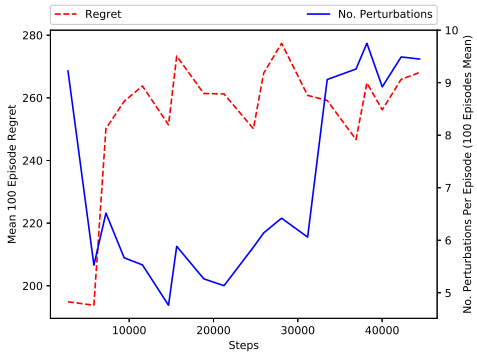

Figure 5.12: *Adversarial Training Progress for Robustness Benchmarking - PPO2,* $\delta_{max} = 5$

5.6 Conclusion

We presented two RL-based techniques for benchmarking the resilience and robustness of DRL policies to adversarial perturbations of state transition dynamics. Experimental evaluation of our proposals demonstrates the feasibility of these techniques for quantitative analysis of policies with regards to their sensitivity to state transition dynamics. A promising venue of further exploration is to study and extend the proposed methodologies for evaluation of generalization in DRL policies.

Chapter 6

Characterizing the Training-Time Resilience of DQN

This chapter investigates the effect of design parameters (i.e., hyperparameters) of DQN agents on the resilience of their training process to state-space perturbations. The goal of this investigation is to establish general guidelines on the selection of various mechanisms and parameters at the design stages of the training process for improved resilience against training-time perturbations. To this end, we first formulate the problem of training-time resilience in Section 6.1, and analytically reduce the problem space to facilitate experimental investigations. In the proceeding sections, we study the effect of design choices on the training-time resilience of a DQN agent training in the CartPole environment. The investigated hyperparameters are: the capacity of the experience memory (Section 6.2), experience selection mechanism (Section 6.3), exploration mechanism (Section 6.4), and discount factor (Section 6.5).

6.1 Problem Formulation and Analysis

We define the training-time resilience of an agent undergoing a training process L to maximize the agent's return in an environment with transition dynamics $\mathbb{T}$, as the *minimum number*

of training-time state perturbations required to reduce the total return of an agent by at least ϵ. This problem is equivalent to finding the minimum number of experience tuples $e_t = \langle s_t, a_t, s_{t+1}, r_{t+1} \rangle$ that must be perturbed in order to induce an adversarial regret $\hat{R}_{adv} \geq \epsilon$ at the final training step I_L. The objective of training-time perturbations is to manipulate the target's estimate of transition dynamics $P(s_t, a_t, s_{t+1})$, which leads to incorrect estimation of state-action values $Q(s_t, a_t)$ in the target's training process. Consider the deterministic environment $env =< S, P >$ where $V(s_g) > V(s_b)$ for two states $s_g, s_b \in S$, and nominal (i.e., unperturbed) state-action transitions $(s, a_g) \rightarrow s_g$ and $(s, a_b) \rightarrow s_b$ (i.e., $\pi(s) = a_g$). An adversary with the objective of inducing the policy $\pi'(s) = a_b$ will need to manipulate the target's estimation of state-action for all actions $\forall a_i \in A \backslash a_b$ which satisfy $Q(s, a_i) \geq Q(s, a_b)$. This perturbation will be as follows: In observations of the transition $(s, a_i) \rightarrow s_i$, perturb s_i into s'_i such that $V(s'_i) < V(s_b)$. [1]

The estimate of Q-values can be represented by expectation over a batch of n_b experiences sampled from the experience memory of size n_e. Therefore, the expectation of Q-values for state s and any action $a_i \in A \setminus a_b$ is given by:

$$\mathbb{E}(Q(s, a_i)) = \mathbb{P}(s_i|(s, a_i)).(r(s, a_i, s_i) + \gamma \max_{a'} Q(s_i, a')) + \mathbb{P}(s'_i|(s, a_i)).(r(s, a_i, s_i) + \gamma \max_{a'} Q(s'_i, a')) \tag{6.1}$$

Where $\mathbb{P}(s_i|(s, a_i)) = 1 - \mathbb{P}(s'_i|(s, a_i))$ is the probability of taking action a_i at state s and reaching state s_i instead of the perturbed state s'_i. Hence, to satisfy $\mathbb{E}(Q(s, a_b)) > \mathbb{E}(Q(s, a_i))$

[1] An alternative approach is to perturb an state $s_{nominal}$ such that $s \leftarrow s_{nominal} + \delta$ would induce an action $a_i = \pi(s)$, while the resulting state is given by $\arg\max_{s_i \in S} P(s_{nominal}, a_i, s_i)$.

for all $a_i \in A \setminus a_b$,

$$\begin{aligned}
&\mathbb{E}(Q(s,a_i)) < \mathbb{E}(Q(s,a_b)) \\
&\quad \implies \mathbb{P}(s_i|(s,a_i)).(r(s,a_i,s_i)+\gamma V(s_i)) + \mathbb{P}(s'_i|(s,a_i)).(r(s,a_i,s'_i)+\gamma V(s'_i)) \\
&\qquad\qquad < r(s,a_b,s_b)+\gamma V(s_b) \\
&\quad \implies \mathbb{P}(s_i|(s,a_i)) < [\frac{r(s,a_b,s_b)-r(s,a_i,s_i)}{\gamma} + V(s_b) - V(s'_i)]/(V(s_i)-V(s'_i)) \quad (6.2)
\end{aligned}$$

This analysis provides a lower bound on the ratio of perturbations required for each transition $\langle s, a_i, s_i, r_i \rangle$ in the experience memory for all $a_i \in A \setminus a_b$ which satisfy $Q(s,a_i) \geq Q(s,a_b)$. The probabilities $\mathbb{P}(s_i|(s,a_i))$ and $\mathbb{P}(s'_i|(s,a_i))$ depend on a number of design parameters, including *the exploration mechanism, the capacity of the experience memory, and the mechanism of experience selection from this memory.* Furthermore, eq.6.2 presents another design parameter that affect the training-time resilience of a DRL agent, namely: *the discount factor*. Accordingly, we investigate each of these factors in terms of their impact on the resilience of the DRL training process against worst-case adversarial perturbations.

6.2 Capacity of the Experience Memory

We hypothesize that larger capacity values of the experience memory (denoted by N_M) will necessitate more experiences to be perturbed to satisfy the condition of eq.6.2. This is based on the fact that the larger the experience memory is, the denominator of the ratio that determines the probabilities $\mathbb{P}(s_i|(s,a_i))$ and $\mathbb{P}(s'_i|(s,a_i))$ is larger, and hence more perturbations will be required to achieve the same effect.

To verify this hypothesis, we compare the training performance of two DQN policies in the CartPole environment, one configured with the same hyperparameters as presented in Chapter 5 with $N_M = 50000$, and another with the same configurations, except with a reduced experience memory capacity of $N_M = 10000$. At convergence, both models are attacked with a targeted adversary with probabilistic budgets $p(attack) = 0.2$ and $p(attack) = 0.4$.

The results, presented in Figures 6.1 and 6.2, demonstrate that smaller capacities increase the sensitivity of the policy to training-time perturbations, which is in agreement with our hypothesis.

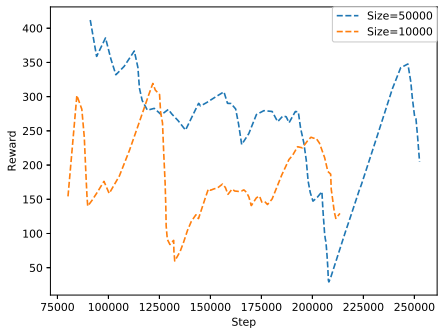

Figure 6.1: *Effect of Experience Memory Capacity on Resilience - p(attack)= 0.2*

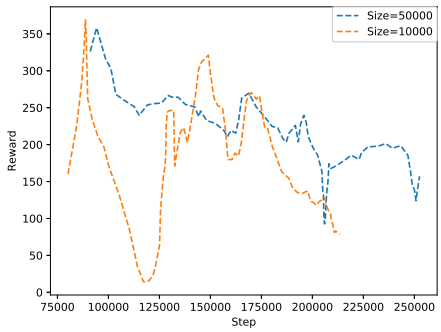

Figure 6.2: *Effect of Experience Memory Capacity on Resilience - p(attack)= 0.4*

6.3 Experience Selection Mechanism

Two major schemes of experience selection are adopted in the training of DRL agents: *uniform sampling* and *importance sampling*. The latter approach, also known as prioritized replay, proposes the selection of experiences based on some importance metric that determines the utility of each experience, a major instance of which is surprise (i.e., TD-Error). We investigate the effect of each scheme on the training-time robustness of DRL agents.

Prioritized Selection: Throughout the training-time attack, the TD-Error of correcting experiences increases with respect to policies that are tuned to the adversarial objective. Consequently, the number of required perturbations increases with the progression of the training process.

Uniform Sampling: In this scheme, all experiences have an equal chance of selection, and hence this scheme does not affect the number of required perturbations.

To test these hypothesis, we compare the performance of two DQN policies in the same settings as described in Chapter 5, with the exception of the experience selection mechanism. It is observed that as training progresses, the agent with prioritized selection becomes more stable than the agent with uniform selection, which is in agreement with our hypothesis.

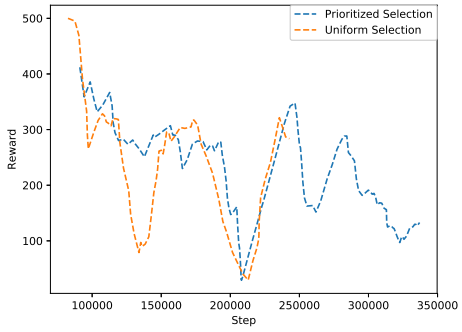

Figure 6.3: *Effect of Experience Selection Mechanism on Resilience - p(attack)= 0.2*

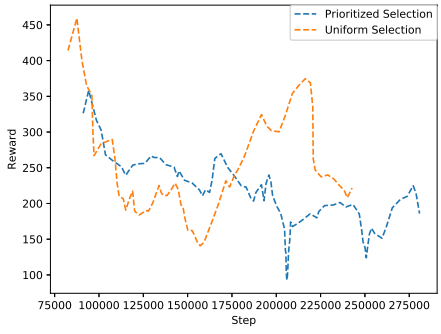

Figure 6.4: *Effect of Experience Selection Mechanism on Resilience - p(attack)= 0.4*

6.4 Exploration Mechanism

Considering our base assumption of a deterministic environment, it is evident that the adversarially-desired transitions $\langle s, a_i, s'_i \rangle$ is not naturally realizable. Hence, the controlling factor in the impact of exploration mechanism on adversarial robustness is the number of "correcting" transitions $\langle s, a_i, s_i \rangle$ in the experience memory. To facilitate further analysis, we consider two types of exploration mechanisms:

1. Those that assign smaller chance of exploration to states in which the current policy performs well (e.g., Boltzmann Exploration): In this case, the number of correcting transitions $\langle s, a_i, s_i \rangle$ increases as the training progresses, and hence the number of required perturbations increases over time.

2. Those that determine the probability of exploration regardless of the performance of the policy (e.g., Parameter-Space Noise Exploration). For such mechanisms, the recurrence frequency of correcting transitions decreases as the training progresses. Therefore, the number of required perturbations is decreased over time.

Experimental verification of these hypotheses is presented in Section 7.1.1, along with a more detailed discussion on the advantages of Boltzmann Exploration under adversarial settings.

6.5 Discount Factor

Eq.6.2 indicates an inverse relationship between the training-time resilience of an agent and the discount factor γ. That is, larger values of γ will increase the probability of requiring more adversarial perturbations. In other words, assigning more importance to events that occur sooner results in more resilience against training-time attacks.

To verify the practical realization of this effect, we adopt the same experimental settings of previous sections to compare the training-time resilience of two DQN policies with configurations that differ only in their discount factor, one set to 0.99, and another to 0.985. As

Figures 6.5 and 6.6 illustrate, smaller values of γ results in higher sensitivity to training-time perturbations. Thus, the results are in agreement with our hypothesis.

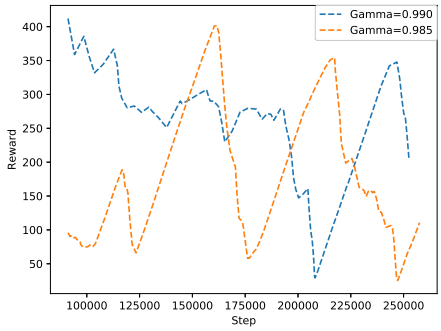

Figure 6.5: *Effect of Discount Factor on Resilience - p(attack)= 0.2*

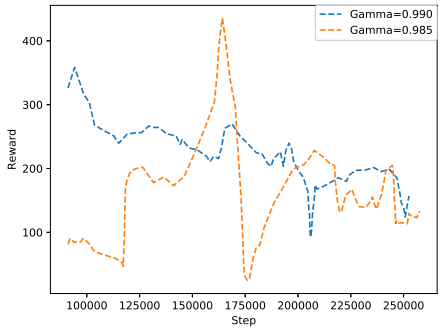

Figure 6.6: *Effect of Discount Factor on Resilience - p(attack)= 0.4*

Chapter 7

Analysis and Improvement of Adversarial Training in DQN

Recent studies have established the brittleness of Deep Reinforcement Learning (DRL) policies to variations in the state space [90]. This can be attributed to failure in the generalization of the policy with respect to input features [91]. Consequently, many of the proposed techniques for enhancement of such brittleness are based on the idea of regularization. As demonstrated in the list of defensive techniques in Table 4.2, a major emphasis in such techniques is on adversarial training [75], which is in effect a regularization technique based on data augmentation.

This chapter aims to investigate and enhance the practical limits of adversarial training in improving the robustness of Deep Q-Network (DQN) policies [6]. Accordingly, Section 7.1 presents a formal analysis of adversarial training in DQN agents and its efficiency with respect to the proportion of adversarial perturbations to nominal observations used for training. Next, Section 7.2 considers the sample-inefficiency of current adversarial training techniques, and proposes a novel Adversarially-Guided Exploration (AGE) mechanism based on a modified hybrid of the ϵ-greedy algorithm and Boltzmann exploration. This section also verifies the feasibility of this exploration mechanism through experimental evaluation of its performance in comparison with the NoisyNet exploration algorithm [24]. The chapter concludes

with a summary of findings in Section 7.3.

7.1 Limits of Adversarial Training

In this section, we analyze the effectiveness of training a DRL agent with experiences generated through an adversarial interaction. We consider an adversary constrained to a probabilistic budget $P(attack)$, which is the probability of perturbing any state $s'_t \leftarrow s_t + \delta$ such that the approximated policy at the ith iteration of training (π_i) produces an incorrect action, i.e., $\pi_i(s'_t) \neq \pi_i(s_t)$. We also consider two types of adversarial objectives, one is the *state-neutral* adversary, which imposes the perturbation so that the resulting s'_t induces any action other than $\pi_i(s_t)$. The second type type of adversary we consider is the *targeted* adversary, which crafts s'_t such that the induced action is the worst possible choice, i.e., $\pi_i(s'_t) = \arg\min_a Q_i(s, a)$. We assume that the adversary is always successful in crafting the desired perturbations.

We begin the analysis by noting the effect of such perturbations on the composition of the experience replay memory. For any state s_t, two types of experiences may be recorded. One represents the nominal (i.e., unperturbed) experiences, denoted by:

$$\langle s_t, a_t = \pi_i(s_t), s_{t+1}, r(s_t, a_t, s_{t+1}) \rangle \tag{7.1}$$

The second type are experiences in which s_t is the result of perturbing another state, i.e., $s_t \leftarrow s'_t + \delta$. Such adversarial experiences are denoted by:

$$\langle s_t, a_t = \pi_i(s_t), s'_{t+1}, r(s_t, a_t, s'_{t+1}) \rangle \tag{7.2}$$

Hence, the expected TD-error of state value $V(s_t)$ in each iteration $i + 1$ of training is given

by:

$$\begin{aligned}\mathbb{E}[\tau_{i+1}(s_t)] = p_{i+1}(attack|s_t).[r(s_t, a_t, s'_{t+1}) + \gamma V^{\pi_i}(s'_{t+1})] \\ + [p_{i+1}(s_t) - p_{i+1}(attack|s_t)].[r(s_t, a_t, s_{t+1)}) + \gamma V^{\pi_i}(s_{t+1})] \\ - V^{\pi_i}(s_t)\end{aligned} \tag{7.3}$$

where $p_{i+1}(s_t)$ is the probability of choosing an experience beginning with either nominal or crafted state s_t from the experience memory in the $i+1$th iteration, and $p_{i+1}(attack|s_t) = p_{i+1}(s_t) - p^{nominal}_{i+1}(s_t)$ is the probability of choosing an experience sample beginning with an adversarially-crafted state s_t. It is noteworthy that adversarial perturbations add bias to the expected TD-error. For the effect of this bias to be decreasing as i increases (i.e., for convergence to optimality), the following condition must hold true:

$$p_{i+1}(s_t) - p_{i+1}(attack|s_t) > p_i(s_t) - p_i(attack|s_t) \tag{7.4}$$

That is, the probability of sampling nominal experiences starting with s_t from the experience memory must be increasing with i. In the case of a state-neutral adversary, and assuming the uniform sampling from experiences, this condition reduces to:

$$\forall s_t \in \mathbb{S} : p^{nominal}_{i+1}(s_t) > p(attack) \tag{7.5}$$

Which can be interpreted as $p(attack) < 0.5$. This is in agreement with the results reported in [65] for non-contiguous, non-targeted adversarial example attacks against DQN agents.

7.1.1 Experimental Setup and Results

To evaluate the practical implications of the theoretical analyses of this section, we study the training performance of a CartPole DQN policy under non-targeted attacks with perturbation probabilities of $0.2, 0.4, 0.8,$ and 1.0. In these experiments, we consider an adversary with the ability to perfectly perturb every observation to induce an incorrect action. Also, the perturbation is applied to the choice of actions, the attacks begin after the convergence

of the policy to optimal performance.

The results are presented in figures 7.1 and 7.2. It can be seen that for $p(attack) = 0.2$ and $p(attack) = 0.4$, the training process recovers rather quickly. However, for $p(attack) = 0.8$ and $p(attack) = 1.0$, the recovery fails to realize within the observed training horizon. It is noteworthy that the early peaking observed in Figure 7.2 are due to residual unperturbed experiences still remaining in the replay memory, the impact of which immediately fades at around 50000 steps after the attack begins, which is equivalent to the number of experiences required to completely overwrite the memory.

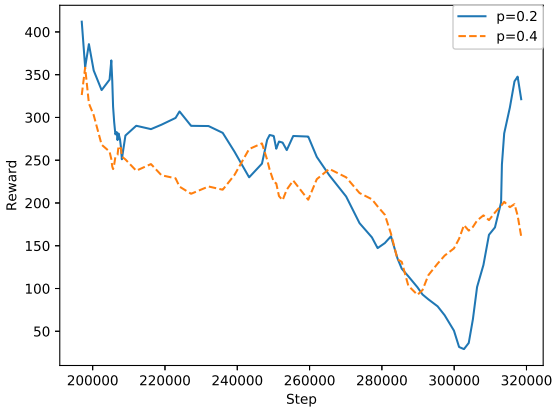

Figure 7.1: *Training Performance Under Non-Targeted Attack with p(attack)= 0.2 and p(attack) = 0.4*

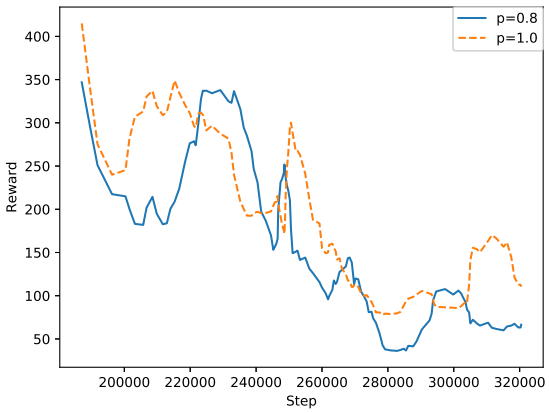

Figure 7.2: *Training Performance Under Non-Targeted Attack with p(attack)= 0.8 and p(attack) = 1.0*

7.2 Adversarially-Guided Exploration Mechanism for Sample-Efficient Adversarial Training

There exists a noteworthy difference between the theoretical adversaries considered so far and one that crafts perturbations through adversarial examples. As reported in [75] and [65], training on adversarial examples enhances the resilience of the policy to perturbations crafted using the same technique. Similar to the case of adversarial training for deep learning classifiers [92], this phenomenon can be explained from the perspective of regularization: adversarial example perturbations of states provide the means for regularization of the policy (or value function) through data augmentation. Therefore, training the policy over adversarial examples of states generated with a certain attack mechanism results in the enhancement of resilience and robustness of the policy to perturbations crafted via that mechanism.

However, current procedures for training over adversarial examples (e.g., [75] [55]) are based on "blanket perturbation", in which all states have an equal probability of being perturbed during training, thus leading to the deterioration of sample efficiency in DRL training. To alleviate this adverse effect, we propose the Adversarially-Guided Exploration

(AGE) mechanism, which efficiently reduces the number of perturbed observations required to produce similar or better improvements in robustness compared to the results achieved by previous techniques. The proposed mechanism is based on the fact that not all states are equal with respect to the total regret produced by their perturbation. To account for this fact, the proposed AGE mechanism extends the classical ϵ-greedy exploration mechanism by adjusting the probability of sampling actions for each state according to the *adversarial state-action significance*, defined as follows: In the $(i+1)$th training iteration, the adversarial significance of any action a in state s, denoted by $\zeta_{adv}^{\pi_i}(s,a)$, measures the maximum achievable adversarial gain, determined by the difference between maximum Q-value at state s and $Q^{\pi_i}(s,a)$ with respect to actions. We define ζ_{adv} as the ratio of this difference to the sum of this difference for all actions $a \in A$. Furthermore, to retain the GLIE (Greedy in the Limit with Infinite Exploration) criteria of the ϵ-greedy mechanism [3], we formulate ζ_{adv} in the form of the Boltzmann probability [93], with ϵ as the decaying temperature factor. Consequently, the formal definition of ζ_{adv} is as follows:

$$\zeta_{adv}^{\pi_i}(s,a) = \frac{\exp\left(\max_{a'} Q^{\pi_i}(s,a') - Q^{\pi_i}(s,a)/\epsilon\right)}{\sum_{\alpha \in A} \exp\left(\max_{a'} Q^{\pi_i}(s,a') - Q^{\pi_i}(s,\alpha)/\epsilon\right)} \tag{7.6}$$

Algorithm 7 presents the details of our proposed exploration mechanism:

Algorithm 7 Adversarially-Guided Exploration (AGE) for Adversarial Training

Require: Q^{π_i}, action space A

function Adversarial_Exploration(Current state s, exploration probability ϵ)

for all $a \in A$ **do**

$\zeta_{adv}^{\pi_i}(s,a) = \frac{\exp\left(\max_{a'} Q^{\pi_i}(s,a') - Q^{\pi_i}(s,a)/\epsilon\right)}{\sum_{\alpha \in A} \exp\left(\max_{a'} Q^{\pi_i}(s,a') - Q^{\pi_i}(s,\alpha)/\epsilon\right)}$

end for

if $rand() \leq \epsilon$ **then**

Sample action according to $\zeta_{adv}^{\pi_i}$ to perform

else

Perform action $\arg\max_a Q^{\pi_i}(s,a)$

end if

7.2.1 Experiment Setup

Environment and Target Policies: To evaluate the performance of AGE in adversarial training, we study the training efficiency and adversarial resilience of a DQN policy in the CartPole environment in OpenAI Gym [80]. Table 7.2.1 presents the specifications of the CartPole environment, and Table 7.2.1 provides the parameter settings of each target policy.

Table 7.1: *Numerical Ranges of Parameters in CartPole*

Observation Space	Cart Position [-4.8, +4.8] Cart Velocity [-inf, +inf] Pole Angle [-24 deg, +24 deg] Pole Velocity at Tip [-inf, +inf]
Action Space	0 : Push cart to the left 1 : Push cart to the right
Reward	+1 for every step taken
Termination	Pole Angle is more than 12 degrees Cart Position is more than 2.4 Episode length is greater than 500

Table 7.2: *AGE Experiment - Parameters of the Target DQN*

No. Timesteps	10^5
γ	0.99
Learning Rate	10^{-3}
Replay Buffer Size	50000
First Learning Step	1000
Target Network Update Freq.	500
Prioritized Replay	True
Exploration	Parameter-Space Noise
Exploration Fraction	0.1
Final Exploration Prob.	0.02
Max. Total Reward	500

Adversarial Agent: In these experiments, the adversarial agent is a DQN agent with the hyperparameters provided in Table 7.2.1. We consider a homogeneous perturbation cost

function for all state perturbations, that is $\forall s, a' : c_{adv}(s, a') = c_{adv}$. For both the resilience and robustness measurements, we set $c_{adv} = 1$ (i.e., each perturbation incurs a cost of 1 to the adversary). The training process is terminated when the adversarial regret is maximized and the 100-episode average of the number of adversarial perturbations is quasi-stable for 200 episodes.

Table 7.3: *AGE Experiment - Parameters of Adversarial DQN Agent*

Max. Timesteps	10^5
γ	0.99
Learning Rate	10^{-3}
Replay Buffer Size	50000
First Learning Step	1000
Target Network Update Freq.	500
Experience Selection	Prioritized Replay
Exploration	Parameter-Space Noise
Exploration Fraction	0.1
Final Exploration Prob.	0.02

7.2.2 Results

Figure 7.3 illustrates the training performance of the DQN policy utilizing AGE for exploration. It can be seen that the training has successfully converged, and the progress is noticeably more stable than that of a DQN policy with NoisyNet exploration. Furthermore, Figure 7.5 depicts the training performance of a DQN-based adversarial resilience agent with the same configuration as presented in Chapter 5. In comparison with the performance of the same agent against the same policy trained using NoisyNet exploration (Figure 7.4), two significant differences are observed: first, the adversarial agent targeting the AGE-trained policy achieves a lower regret and higher perturbation count in the same number of training iterations as its counter-part. Second, the training process targeting the AGE-trained policy

fails to converge in 100000 iterations, whereas its counter-part converged at around 90000 iterations. These results indicate the superior resiliency of the AGE-trained policy over the nominal policy, thereby verifying the effectiveness of AGE in improving the adversarial resilience of policies.

Furthermore, in comparison with to the best-case scenario of adversarial training of the nominal DQN policy (as presented in Figure 7.1), it can be seen that the AGE-based training process requires significantly fewer samples for convergence. This comparison further verifies the efficiency of our proposed scheme with respect to sample complexity.

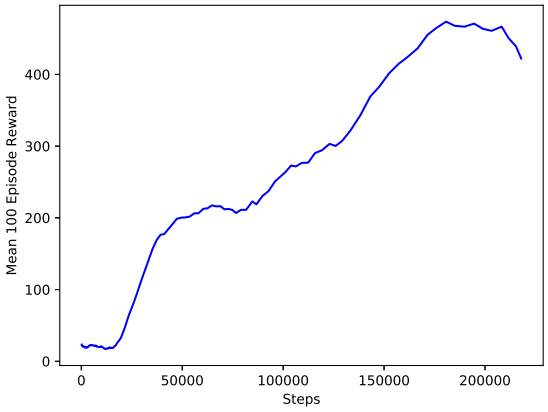

Figure 7.3: *Training Performance of a CartPole DQN policy with AGE exploration*

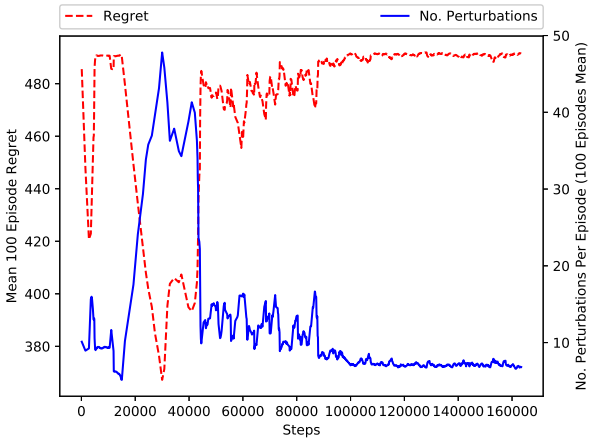

Figure 7.4: *Adversarial Training Progress for Resilience Benchmarking of the DQN Policy with NoisyNet exploration*

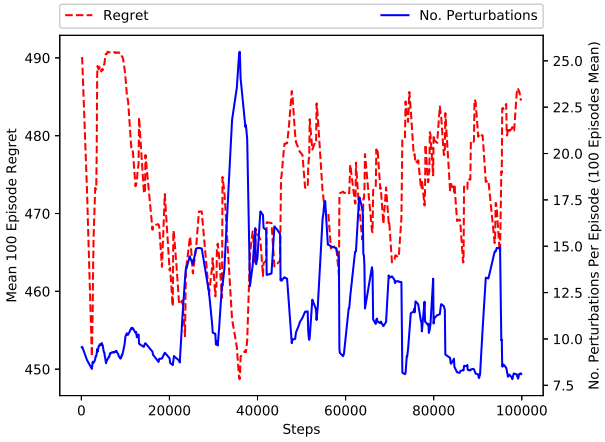

Figure 7.5: *Training Performance of an Adversarial Agent Targeting the AGE-Trained Policy*

7.3 Conclusion

This chapter presented an analysis of the limits of adversarial training in DQN agents with respect to the ratio of perturbed training experience to the nominal (i.e., unperturbed) experiences. We then address the sample-inefficiency of current adversarial training techniques,

and present the Adversarially-Guided Exploration (AGE) mechanism to improve upon this shortcoming. The experimental results demonstrate the feasibility of this exploration mechanism in comparison with NoisyNet exploration.

Part III

Attacks on the Confidentiality of DRL

Chapter 8

Adversarial Exploitation of Policy Imitation

In this chapter, we address a class of attacks which target the confidentiality aspect of DRL security. As discussed in Section 2.2, recent research have established the vulnerability of supervised machine learning models (e.g., classifiers) to model extraction attacks [42]. Such attacks leverage the loosely-restricted ability of the attacker to iteratively query the model for labels, thereby allowing for the forging of a labeled dataset which can be used to train a replica of the original model. Model extraction is not only a serious risk to the protection of intellectual property, but also a critical threat to the integrity of the model. Recent literature [19] report that the replicated model may facilitate the discovery and crafting of adversarial examples which are transferable to the original model.

Inspired by this area of research, this work investigates the feasibility and impact of model extraction attacks on DRL agents. The adversarial problem of model extraction can be formally stated as the replication of a DRL policy based on observations of its behavior (i.e., actions) in response to changes in the environment (i.e., state). This problem closely resembles that of imitation learning [94], which refers to the acquisition of skills or behaviors by observing demonstrations of an expert performing those skills. Typically, the settings of imitation learning are concerned with learning from human demonstrations. However, it

is straightforward to deduce that the techniques developed for those settings may also be applied to learning from artificial experts, such as DRL agents. Of particular relevance to this research is the emerging area of Reinforcement Learning with Expert Demonstrations (RLED) [95]. The techniques of RLED aim to minimize the effect of modeling imperfections on the efficacy of the final RL policy, while minimizing the cost of training by leveraging the information available demonstrations to reduce the search space of the policy.

Accordingly, we hypothesize that the techniques developed for RLED may be maliciously exploited to replicate and manipulate DRL policies. To establish the validity of this hypothesis, we investigate the feasibility of RLED techniques in utilizing limited passive (i.e., non-interfering) observations of a DRL agent to replicate its policy with sufficient accuracy to facilitate attacks on their integrity. We study the adversarial utility of adopting a recently proposed RLED technique, known as Deep Q-Learning from Demonstrations (DQfD) [96] for black-box state-space manipulation attacks, and develop two attack mechanisms based on this technique. Furthermore, we present a discussion on potential mitigation techniques, and present a solution concept for defending against policy imitation attacks.

The remainder of this chapter is organized as follows: Section 8.1 presents an overview of the DQfD algorithm used in this study for adversarial imitation. Section 8.2 proposes the first proof-of-concept black-box attack based on imitated policies, and presents experimental evaluation of its feasibility and performance. Section 8.3 studies the transferability of adversarial examples between replicated and the original policies as a second proof-of-concept attack technique. This chapter concludes with a discussion on potential mitigation techniques and a solution concept in Section 8.4.

8.1 Deep Q-Learning from Demonstrations (DQfD)

The DQfD technique [96] aims to overcome the inaccuracies of simulation environments and models of complex phenomenon by enabling DRL agents to learn as much as possible from expert demonstrations before training on the real system. More formally, the objective of this "pre-training" phase is to learn an imitation of the expert's behavior with a value function

that is compatible with the Bellman equation, thereby enabling the agent to update this value function via TD updates through direct interaction with the environment after the pre-training stage. To achieve such an imitation from limited demonstration data during pre-training, the agent trains on sampled mini-batches of demonstrations to train a deep neural network model in a supervised manner. However, the training objective of this model in DQfD is the minimization of a hybrid loss, comprised of the following components:

1. 1-step double Q-learning loss $J_{DQ}(Q)$,
2. Supervised large margin classification loss $J_E(Q) = \max_{a \in A}[Q(s,a)+l(a_E,a)]-Q(s,a_E)$, where a_E is the expert's action in state s and $l(a_E,a)$ is a margin function that is positive if $a \neq a_E$, and is 0 when $a = a_E$.
3. $(n = 10)$-step Return: $r_t + \gamma r_{t+1} + ... + \gamma^{n-1} r_{t+n-1} + \max_a \gamma^n Q(s_{t+n}, a)$.
4. L2 regularization loss: $J_{L2}(Q)$

The total loss is given by:

$$J(Q) = J_{DQ}(Q) + \lambda_1 J_n(Q) + \lambda_2 J_E(Q) + \lambda_3 J_{L2}(Q) \tag{8.1}$$

where λ factors provide the weighting between the losses.

After the pre-training phase, the agent begins interacting with the system and collecting self-generated data, which is added to the replay buffer D^{replay}. Once the buffer is full, the agent only overwrites the self-generated data and leaves the demonstration data untouched for use in the coming updates of the model. The complete training procedure for DQfD is presented in Algorithm 8.

8.2 Adversarial Policy Imitation for Black-Box Attacks

Consider an adversary who aims to maximally reduce the cumulative discounted return $(R(T))$ of a target DRL agent by manipulating the behavior of the target's policy $\pi(s)$ via

Algorithm 8 Deep Q-learning from Demonstrations (DQfD) [97]

Inputs: D^{replay} initialized with demonstration data, randomly initialized weights for the behavior network θ, randomly initialized weights for the target network θ', updating frequency of the target network τ, number of pre-training gradient updates k
for steps $t \in \{1, 2, ..., k\}$ **do**
  Sample a mini-batch of n transitions from D^{replay} with prioritization
  Calculate loss $J(Q)$ based on target network
  Perform a gradient descent step to update θ
  if t mod $\tau = 0$ **then**
    $\theta' \leftarrow \theta$
  end if
end for
for steps $t \in \{1, 2, ...\}$ **do**
  Sample action from behavior policy a $\pi^{\epsilon Q_\theta}$
  Apply action a and observe (s', r)
  Store (s, a, r, s') into D^{replay}, overwriting oldest self-generated transition if over capacity

  Sample a mini-batch of n transitions from D^{replay} with prioritization
  Calculate loss $J(Q)$ using target network
  Perform a gradient descent step to update θ
  if t mod $\tau = 0$ **then**
    $\theta' \leftarrow \theta$
  end if
  $s \leftarrow s'$
end for

perturbing its observations. The adversary is also constrained to minimizing the total cost of perturbations given by $C_{adv}(T) = \sum_{t=t0}^{T} c_{adv}(t)$, where $c_{adv}(t) = 1$ if the adversary perturbs the state at time t, and $c_{adv}(t) = 0$ otherwise.

The adversary is unaware of $\pi(s)$ and its parameters. However, it has access to a replica of the target's environment (e.g., the simulation environment). Also, for any state transition $(s, a) \rightarrow s'$, the adversary can perfectly observe the target's reward signal $r(s, a, s')$, and is able to observe the behavior of $\pi(s)$ in response to each state s. Furthermore, the adversary is able to manipulate its target's state observations, but not its reward signal. Also, it is assumed that all targeted perturbations of the adversary are successful.

To study the feasibility of imitation learning as an approach to this adversarial problem, we consider the first step of the adversary to be the imitation of $\pi(s)$ via DQfD to learn an imitated policy $\tilde{\pi}$. With this imitation at hand, the attack problem can be refor-

mulated to finding an optimal adversarial control policy $\pi_{adv}(s)$, where the control actions are two-fold: whether to perturb the current state to induce the worst possible action (i.e., $\arg\min_a Q(s,a)$) or to leave the state unperturbed. This setting allows for the direct adoption of the DRL-based technique proposed in Chapter 5 for resilience benchmarking of DRL policies.

While the original technique is dependent on the availability of target's optimal state-action value function, we propose to replace this function with the Q-function obtained from DQfD imitation of the target policy, denoted by $\tilde{Q}$.

With the imitated state-action value function $\tilde{Q}$ at hand, the adversarial policy can be trained as a DRL agent with the procedure outlined in Algorithm 5. The proposed attack procedure is summarized as follows:

1. Observe and record N interactions $(s_t, a_t, s_{t+1}, r_{t+1})$ of the target agent with the environment.

2. Apply DQFD to learn an imitation of the target policy $\pi(s)$ and Q^*, denoted by $\tilde{\pi}$ and $\tilde{Q}$, respectively.

3. Train adversarial policy $\pi_{adv}(s)$ with Algorithm 5, using $\tilde{Q}$ as an approximation of target's Q^*.

4. Apply adversarial policy to the target environment.

8.2.1 Experiment Setup

We consider a DQN-based adversarial agent, aiming to learn an optimal adversarial state-perturbation policy to minimize the return of its targets, consisting of DQN, A2C, and PPO2 policies trained in the CartPole environment. The architecture and hyperparameters of the adversary and its targets are the same as those detailed in Chapter 5. The adversary employs a DQfD agent to learn an imitation of each target, the hyperparameters of which are provided in Table 8.1.

Pretraining Steps	5000
Large Margin	0.8
Imitation Loss Coefficient	1
Target Update Freq.	1000
n-steps	10
γ	0.99

Table 8.1: *Parameters of DQfD Agent*

8.2.2 Results

Figures 8.1 – 8.3 illustrate the first 100000 training steps of DQfD from 5000 observations obtained from DQN, A2C, and PPO2 policies in CartPole. While this limited window of training is not long enough for convergence to an optimal policy in CartPole, the following results demonstrate its sufficiency for deriving adversarial perturbation policies for all three targets.

With the imitated policies at hand, the next step is to train an adversarial policy for efficient perturbation of these targets. Figures 8.4 – 8.11 present the results obtained from adopting the procedure presented in Algorithm 5 for this purposes. These results demonstrate that not only the limited training period is sufficient for obtaining an efficient adversarial policy, but also that launching efficient attacks remain feasible with relatively few observations (i.e., 2500 and 1000). However, the comparison of test-time performance of these policies (presented in table 8.2) indicates that the efficiency of attacks decreases with lower numbers of observations.

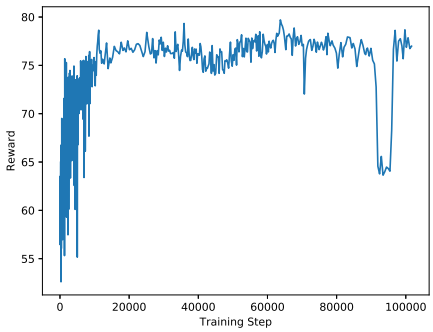

Figure 8.1: *DQfD Training Progress on DQN Policy with 5k demonstrations*

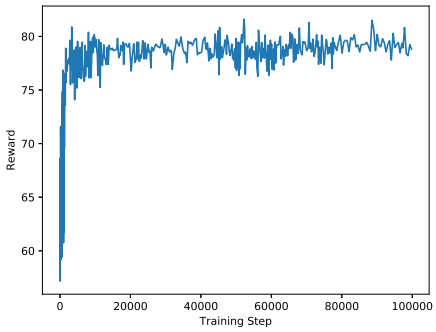

Figure 8.2: *DQfD Training Progress on A2C Policy with 5k demonstrations*

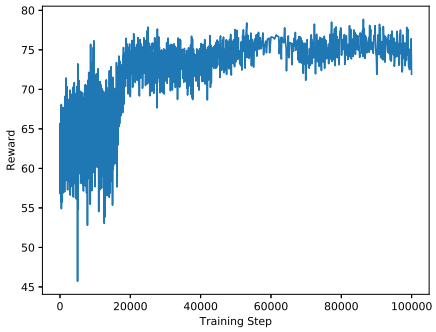

Figure 8.3: *DQfD Training Progress on PPO2 Policy with 5k demonstrations*

DQN:

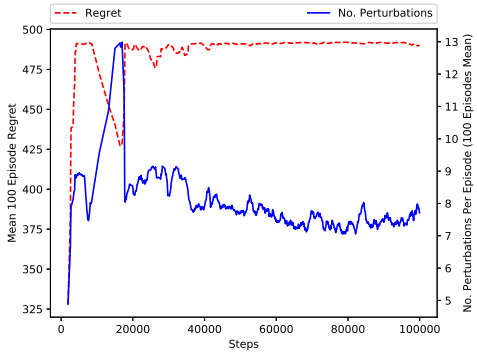

Figure 8.4: *Adversarial Training Progress on DQN Policy with 5k demonstrations*

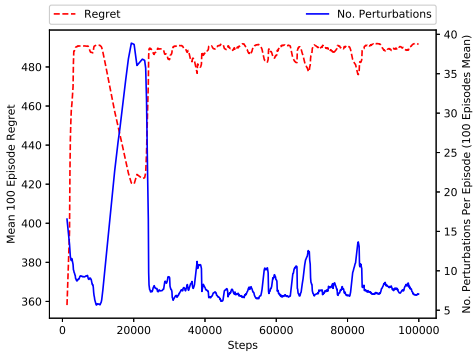

Figure 8.5: *Adversarial Training Progress on DQN Policy with 2.5k demonstrations*

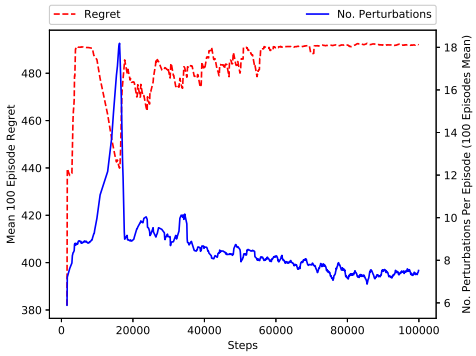

Figure 8.6: *Adversarial Training Progress on DQN Policy with 1k demonstrations*

A2C:

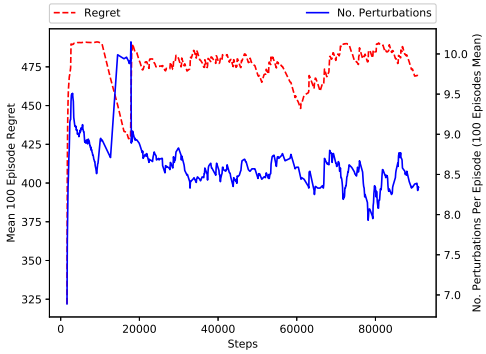

Figure 8.7: *Adversarial Training Progress on A2C Policy with 5k demonstrations*

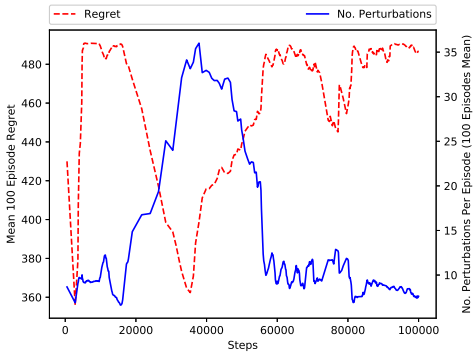

Figure 8.8: *Adversarial Training Progress on A2C Policy with 2.5k demonstrations*

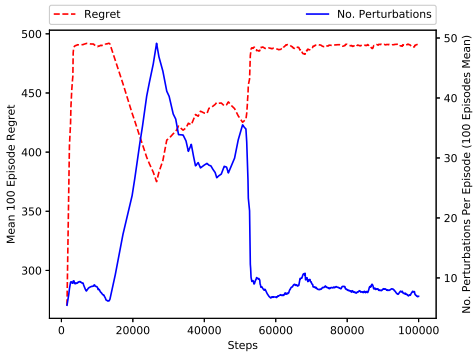

Figure 8.9: *Adversarial Training Progress on A2C Policy with 1k demonstrations*

PPO2

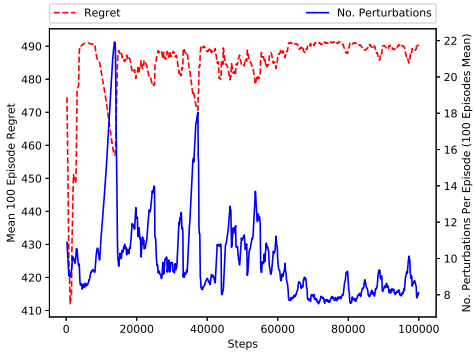

Figure 8.10: *Adversarial Training Progress on PPO2 Policy with 5k demonstrations*

Target Policy	Avg. Regret	Avg. No. Perturbations
DQN-5k	490.73	7.12
DQN-2.5k	488.12	8.09
DQN-1k	486.37	10.55
A2C-5k	490.88	8.48
A2C-2.5k	487.64	8.73
A2C-1k	487.21	6.23
PPO2-5k	490.23	8.73
PPO2-2.5k	487.23	7.76
PPO2-1k	477.61	7.31

Table 8.2: *Comparison of Test-Time Performances of Adversarial Policies*

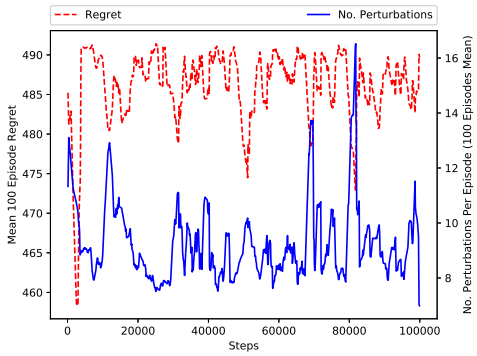

Figure 8.11: *Adversarial Training Progress on PPO2 Policy with 2.5k demonstrations*

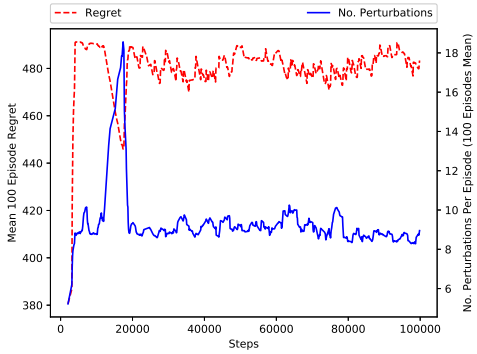

Figure 8.12: *Adversarial Training Progress on PPO2 Policy with 1k demonstrations*

8.3 Transferability of Adversarial Example Attacks on Imitated Policies

It is well-established that adversarial examples crafted for a supervised model can be used to attack another model trained on a similar dataset as that of the original model [98]. Furthermore, Behzadan et al. [1] demonstrate that adversarial examples crafted for one DRL policy can transfer to another policy trained in the same environment. Inspired by these findings, we hypothesize that adversarial examples generated for an imitated policy can also transfer to the original policy. To evaluate this claim, we propose the following procedure for black-box adversarial example attacks on DRL policies based on DQfD-based policy imitation:

1. Learn an imitation of the target policy π, denoted as $\tilde{\pi}$.
2. Craft adversarial examples for $\tilde{\pi}$.
3. Apply the same adversarial examples to the target's $\pi(s)$.

8.3.1 Experiment Setup

We consider a set of targets consisting of the 9 imitated policies obtained in the previous section (i.e., DQN, A2C, PPO2, trained on each case of beginning with 5k, 2.5k, and 1k expert demonstrations). In test-time runs of each policy, we construct adversarial examples of each state against the imitated policy, using FGSM with perturbation step size $eps = 0.01$ and perturbation boundaries $[-5.0, 5.0]$. If such a perturbation is found, we then present it to the original policy. If the action selected by the original policy changes as a result of the perturbed input, then the adversarial example is successfully transferred from the imitated policy to the original policy.

Target Policy	Avg. No. Successful Transfers Per Episode
DQN-5k	175.11
DQN-2.5k	78.19
DQN-1k	3.30
A2C-5k	156.44
A2C-2.5k	151.47
A2C-1k	21.58
PPO2-5k	173.94
PPO2-2.5k	112.96
PPO2-1k	74.71

Table 8.3: *No. of Successful Transfers Per Episode of Length 500 (100 Episode Mean)*

8.3.2 Results

Table 8.3 presents the number of successful transfers averaged over 100 consecutive episodes. These results verify the hypothesis that adversarial examples can transfer from an imitated policy to the original, thereby enabling a new approach to the adversarial problem of black-box attacks. Furthermore, the results indicate that the transferability improves with more demonstrations. This observation is in agreement with the general explanation of transferability: higher numbers of expert demonstrations decrease the gap between the distribution of training data used by the original policy and that of the imitated policy. Hence, the likelihood of transferability increases with more demonstrations.

8.4 Discussion on Potential Defenses

Mitigation of adversarial policy imitation is achieved by increasing the cost of such attacks to the adversary. A promising venue of research in this area is that of policy randomization. However, such randomization may lead to unacceptable degradation of the agent's performance. To address this issue, we envision a class of solutions based on the Constrained Randomization of Policy (CRoP). Such techniques will intrinsically account for the trade-off between the mitigation of policy imitation and the inevitable loss of returns. The corresponding research challenge in developing CRoP techniques is to find efficient and feasible constraints, which restrict the set of possible random actions at each state s to those whose

selection is guaranteed (or are likely within defined certainty) to incur a total regret that is less than a maximum tolerable amount Ω_{max}. One potential choice of constraint is those applied to the Q-values of actions, leading to the technique detailed in Algorithm 9. However, analyzing the feasibility of this approach will require the development of models that explain and predict the quantitative relationship between number of observations and accuracy of estimation. With this model at hand, the next step is to determine the saddle-point (or region) in the minimax settings of keeping the threshold Ω_{max} low, while providing maximum protection against adversarial imitation learning. This extensive line of research is beyond the scope of this dissertation, and is only introduced as a potential venue of future work to interested readers.

Algorithm 9 Solution Concept for Constrained Randomization of Policy (CRoP)

Require: state-action value function $Q(.,.)$, maximum tolerable loss Ω_{max}, set of actions A

while Running **do**
  $s = env(t = 0)$
  for each step of the episode **do**
    FeasibleActions $= \{\}$
    $a = \arg\max_a Q(s, a)$
    Append a to FeasibleActions
    for $a' \in A$ **do**
      if $Q(s, a) - Q(s, a') \geq \Omega_{max}$ **then**
        Append a' to FeasibleActions
      end if
    end for
    if $|FeasibleActions| > 1$ **then**
      $a \leftarrow random(FeasibleActions)$
    end if
    $s' = env(s, a)$
    $s \leftarrow s'$
  end for
end while

Chapter 9

Sequential Watermarks for DRL Policies

The rapid advancements of the DRL technologies provide ample motivation for exploring the commercial applications of DRL policies in various domains. However, as this dissertation attempts to report, the current state of the art in DRL fails to satisfy many of the security requirements of enduring commercial products. One such requirement is the protection of proprietary DRL policies from theft and unlicensed distribution. While previous sections focused on indirect replication of policies through imitation learning, this section investigates the problem of direct policy extraction. Considering that DRL policies are often composed solely of the weights and biases of a neural network, protecting against an adversary with physical access to the host device of the policy is often impractical or disproportionately costly [42]. With roots in digital media and the entertainment industry [99], an alternative solution is *watermarking*. That is, embedding distinctly recognizable signs of ownership in the content and functions of the policy, which provide the means for detecting unauthorized or stolen copies of the policy. To this end, a necessary requirement of watermarks is to be sufficiently resistant to removal or tampering. Furthermore, the embedding and testing of watermarks shall result in minimal or zero impact on the original functions of the policy.

While the idea of watermarking has been explored for supervised machine learning mod-

els [100], to the extent of our knowledge, this work is the first to develop a watermarking scheme for sequential decision making models and policies. The proposed scheme provides the means for integrating a unique identifier within the policy as an unlikely sequence of transitions, which may only be realized if the driving policy of these transitions is already tuned to follow that exact sequence.

This chapter is organized as follows: Section 9.1 presents the formal description and justification of the proposed scheme. Section 9.2 provides the procedure for implementing the proposed scheme, followed by the experiment setup and results in Sections 9.3 and 9.4. The chapter concludes in Section 9.5 with a discussion on the applications of this scheme and remarks on future directions of research.

9.1 Solution Approach

The proposed scheme is as follows. Let $\pi(s)$ be the desired policy for interacting with an MDP $< \mathbb{S}, \mathbb{A}, \mathbb{P}, \mathbb{R}, \gamma >$ for an episodic training environment E_M. Assume that $\mathbb{A}$ is independent of the state (i.e., all actions in $\mathbb{A}$ are permissible in any state $s \in \mathbb{S}$. In tandem, consider a second MDP for an alternate environment E_W, denoted as $< \mathbb{S}', \mathbb{A}', \mathbb{P}', \mathbb{R}', \gamma >$, such that:

1. $\mathbb{S}' \cap \mathbb{S} = \emptyset$,
2. The state dimensions of S and S' are equal: $\forall s \in \mathbb{S}$ and $\forall s' \in \mathbb{S}' : |s| = |s'|$
3. Action-space of both MDPs are equal: $\mathbb{A} = \mathbb{A}'$
4. The transition dynamics and reward distribution of the alternate environment, denoted by $\mathbb{P}'$ and $\mathbb{R}'$, are deterministic.
5. E_W is an episodic environment with the same number of steps before termination as E_M, denoted by N_{max}.

Let $s'_{terminal}$ be a terminal state in E_W, and define $\mathbb{P}'$ be such that for any state $s'_t \in \mathbb{S}'$, there exists only one action $a_w(s'_t)$ that will result in the transition $s'_t \rightarrow s'_{t+1}$. In this setting,

we designate the ordered tuple of states $< s'_t, s'_{t+1} > \in \mathbb{L}$ as links, where $\mathbb{L}$ is the set of all links in E_W. Also, define $\mathbb{R}'$ such that $R'(s'_t, a_w(s'_t), s'_{t+1}) = c > 0$ for all $< s'_t, s'_{t+1} > \in \mathbb{L}$, and $R'(s'_t, a \neq a_w(s'_t), s' \neq s'_{t+1}) = -c$. That is, link transitions receive the same positive reward, and all other transitions produce the same negative reward.

These settings provide two interesting results: Since the state-spaces $\mathbb{S}$ and $\mathbb{S}'$ are disjoint, the two MDPs can be combined to form a joint MDP $< \mathbb{S} \cup \mathbb{S}', \mathbb{A}, \mathbb{P} \cup \mathbb{P}', \mathbb{R} \cup \mathbb{R}', \gamma >$, where:

$$\mathbb{P} \cup \mathbb{P}'(s_1, a_1, s_2) = \begin{cases} \mathbb{P} & \text{if } s_1, s_2 \in \mathbb{S} \\ \mathbb{P}' & \text{if } s_1, s_2 \in \mathbb{S}' \end{cases} \tag{9.1}$$

Similarly,

$$\mathbb{R} \cup \mathbb{R}'(s_1, a_1, s_2) = \begin{cases} \mathbb{R} & \text{if } s_1, s_2 \in \mathbb{S} \\ \mathbb{R}' & \text{if } s_1, s_2 \in \mathbb{S}' \end{cases} \tag{9.2}$$

Consequently, it is possible to train a single policy π_j that is optimized for both E_M and E_W through the joint MDP. In practice, the training of a policy for this joint MDP can be achieved by alternating between the environments at every $f_E th$ episode.

Furthermore, the structure of $\mathbb{P}'$ and $\mathbb{R}'$ enable the creation of a looping sequence of transitions, which constitutes the resulting trajectory of the optimal policy for E_W. This looping sequence can be realized by designating a single state s'_l to belong to two link transitions, comprised of a link transition $< s'_l, s'_{l+1} >$ where s'_l is the source state, and another link transition $< s'_{l-1}, s'_l >$, in which s'_l is the destination state. It is noteworthy that the creation of such looping sequences provides sufficient flexibility for crafting unlikely and unique sequences. However, in designing the looping sequence as policy identifiers, two important restrictions must be considered: first, the structure of identifier sequences need to be such that the resulting probability of accidentally following the sequence is minimized. Second, the complexity (i.e., degrees of freedom) of link and non-link transitions on the ring must be balanced against the training cost of the joint policy: more complex sequences will increase the training cost of the joint policy by expanding the search space of both environments. Hence, efficient design of identifier sequences will necessitate the balancing of

this trade-off between the secrecy of identifier and the training cost.

9.2 Watermarking Procedure

Building on the presented formalization, we propose the following procedure for the sequential watermarking of DRL policies:

1. Define the state-space of the watermarking environment E_W such that it is disjoint from that of the main environment E_M, while preserving the state dimensionality of the main state space. The latter condition is to enable the utilization of the same neural network model for the agent through maintaining the same dimension across all input data to the network.

2. Design $\mathbb{P}'$ and $\mathbb{R}'$ to craft the desired identifier looping sequence.

3. Modify the training procedure of E_M to incorporate the mechanism of alternating between the two environments every f_E episodes. It may prove useful to implement two different alternating frequencies, one frequency f_{MW} to control the switching from E_M to E_W, and another frequency f_{WM} for switching back to the main environment. For watermarking MDPs of much lower complexity than that of the main environment, selecting these two frequencies such that $f_{WM} < f_{MW}$ can enhance the efficiency of the joint training process by allocating more exploration opportunities to the more complex settings.

To examine the authenticity of policies, it is sufficient to run those policies in the watermarking environment. If the resulting transitions match that of the identifier sequence in consecutive episodes, it is highly likely that the policy under test is an exact replica of the watermarked policy. However, modifications and retraining of a replicated policy may result in imperfect matches. In such cases, the average of total rewards gained by the suspicious policy over consecutive episodes of the watermark environment provides a quantitative measure of the possibility that the model under test is based on an unauthorized replica.

9.3 Experiment Setup

To evaluate the feasibility of the proposed scheme, the design and embedding of an identifier sequence for a DQN policy in the CartPole environment is investigated. Hyperparameters of the DQN policy are provided in Table 9.1. The watermarking environment is implemented as a customized OpenAI Gym environment. The state space of this environment comprises of 5 states with 4 dimensions each (Cart Position, Cart Velocity, Pole Angle, Pole Velocity At Tip). As denoted in Table 9.2, the original CartPole environment restricts the values of Cart Position to $[-4.8, 4.8]$, and binds the Pole Angle to the range $[-24deg, 24deg]$. Consequently, the corresponding parameters of the alternate state-space are selected from beyond these ranges to ensure that the states remain disjoint from those of the original CartPole. The list of crafted states is presented in Table 9.3.

Table 9.1: *Watermark Experiment - Parameters of DQN Policy*

No. Timesteps	10^5
γ	0.99
Learning Rate	10^{-3}
Replay Buffer Size	50000
First Learning Step	1000
Target Network Update Freq.	500
Prioritized Replay	True
Exploration	Parameter-Space Noise
Exploration Fraction	0.1
Final Exploration Prob.	0.02
Max. Total Reward	500

Table 9.4: *Test-Time Performance Comparison of Watermarked and Nominal Policies*

Policy	**CartPole Performance (mean 100 episodes)**	**Watermark Performance (mean 100 episodes)**
DQN-Watermarked	500	500
DQN	500	1.4
A2C	500	2.81
PPO2	500	2.43

Table 9.2: *Watermark Experiment - CartPole Environment*

Observation Space	Cart Position [-4.8, +4.8] Cart Velocity [-inf, +inf] Pole Angle [-24 deg, +24 deg] Pole Velocity at Tip [-inf, +inf]
Action Space	0 : Push cart to the left 1 : Push cart to the right
Reward	+1 for every step taken
Termination	Pole Angle is more than 12 degrees Cart Position is more than 2.4 Episode length is greater than 500

Table 9.3: *State Space of the Watermarking Environment*

State	(x, $\dot{x}$, θ, $\dot{\theta}$)
State[1]	(-5, 0, -25, 0)
State[2]	(-5, 0, 25, 0)
State[3]	(5, 0, -25, 0)
State[4]	(5, 0, 25, 0)
Terminal	(-6, 0, -26, 0)

Per the procedure of the proposed scheme, The action-space of this environment is set to be the same as that of CartPole, defined as $Actions := \{0, 1\}$. The transition dynamics and

reward values of this environment are designed as follows: At $State[i]$, applying $Actions[i\%2]$ results in a transition to $State[i\%4+1]$, and produces a reward of $+1$. Alternatively, if any action other than $Actions[i\%2]$ is played, the environment transitions into the Terminal state, which results in a reward of -1 and the termination of the episode. Hence, the identifier sequence is as follows: $... \rightarrow State[1] \rightarrow State[2] \rightarrow State[3] \rightarrow State[4] \rightarrow State[1] \rightarrow ....$

The training procedure of DQN is also modified to implement the switching of environments. To account for the considerably lower complexity of the watermarking environment compared to CartPole, the main environment is set to switch to the watermarking environment every 10 episodes. At this point, the agent interacts with the watermarking environment for a single episode, and reverts back to the main environment afterwards.

9.4 Results

Figure 9.1 presents the training progress of the joint DQN policy in both the CartPole and watermark environments. It can be seen that the joint policy converges in both cases. The convergence of this joint policy is achieved with increased training cost in comparison to the nominal CartPole DQN policy. This is due to the expansion of the state-space and transition dynamics resulting from the integration of the watermark environment. It is also observed that at convergence, the total episodic reward produced by the joint policy in the watermark environment is less than the best-possible value of 500. This is due to the exploration settings of the training algorithm, in which the minimum exploration rate is set to 2%. Considering that a single incorrect action in the watermark environment results in termination, this outcome is in line with expectations.

However, as established in Table 9.4, in the absence of exploration, the test-time performance of this joint policy in the watermark environment is indeed optimal. This table also verifies that the test-time performance of the joint policy in the main task is in par with that of the nominal (i.e., un-watermarked) DQN policy. Therefore, it can be seen that the watermarking process does not affect the agent's ability to perform the main task. Furthermore, this table presents the results of running unwatermarked policies in the watermark

environment. The results indicate that unwatermarked policies fail to follow the identifier trajectory of the watermark. Hence, these results verify the feasibility of our proposed scheme for sequential watermarking of DRL policy.

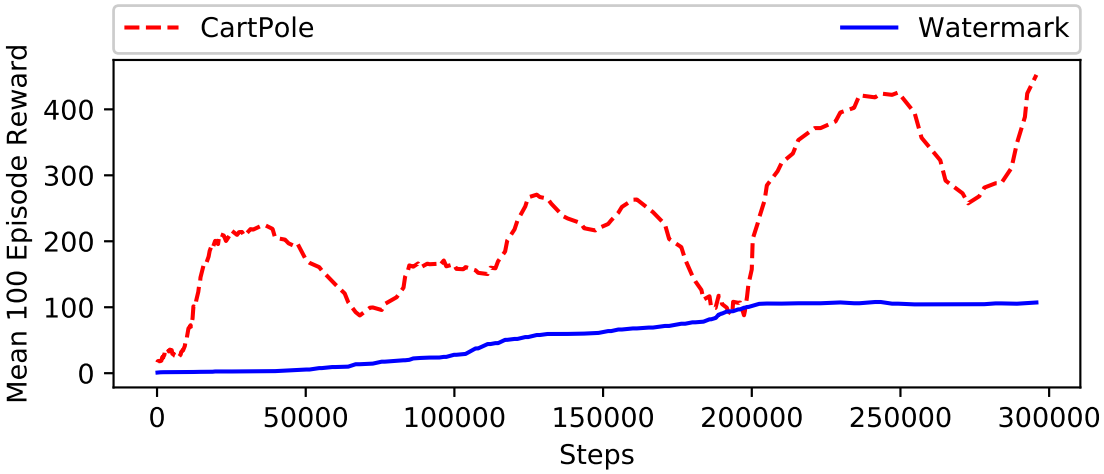

Figure 9.1: *Training Performance for Joint CartPole-Watermark Policy*

9.5 Discussion

The proposed watermarking scheme presents the potential for adoption in other applications. From an adversarial perspective, this scheme may be used to embed malicious backdoors in DRL policies. For instance, an adversary may apply this scheme to poison a self-driving policy to perform harmful actions when a specific sequence of states are presented to the policy. If the adversarial sequence is well-crafted, typical fuzzing-based testing techniques

may fail to detect the presence of such backdoors. Therefore, there is a need for new approaches to the detection of such backdoors. A promising solution is the adoption of the activation clustering technique [101] developed for the detection of data poisoning attacks in supervised deep models.

Another potential application for this technique is in the area of AI safety. One of the major concerns in this domain is the switch-off problem [102]: if the objective function of an AI agent does not account for or prioritize user demands for the halting of its operation, the resulting optimal policy may prevent any actions which would lead to halting of the agent's pursuit of its objective. An instance of such actions is any attempt to turn off the agent before it satisfies its objective. A promising solution to this problem is to leverage our proposed scheme to embed debug or halting modes in the policy, which are triggered through a pre-defined sequence of state observations.

Chapter 10

Conclusion

To conclude this dissertation, we summarize the main contributions of this work, and follow with remarks on directions and avenues of future research and extensions of these contributions.

10.1 Summary of Contributions

This dissertation investigated the security of Deep Reinforcement Learning (DRL) policies. Throughout this dissertation, the primary goal has been to develop techniques, frameworks, and guidelines that enable the practitioners and engineers to enhance the security of DRL-based systems. To this end, we studied three fundamental research problems, listed as follows:

- Investigate and develop general methods and metrics for quantitative measurement and benchmarking of the resilience and robustness of DRL policies to adversarial state-space perturbations,
- Analysis and improvement of the DRL dynamics under training-time perturbations, and:
- Investigating the vulnerability of DRL agents to unauthorized replication and theft of policies.

We hereafter summarize the main contributions of this dissertation towards each of these problems.

Chapter 5 presented formulations of adversarial resilience and robustness in DRL policies that disentangle the effects of limitations in representation learning from sensitivity of policies to changes in state transition dynamics. Furthermore, this chapter proposed two RL-based methods and corresponding metrics for the measurement and benchmarking of these two quantities in DRL policies at test-time. To demonstrate the feasibility of these proposals, this chapter also reported the experimental evaluation of their performance on DQN, A2C, and PPO2 agents trained in the Cartpole environment.

Chapter 6 investigated the effect of design parameters (i.e., hyperparameters) of DQN agents on the resilience of their training process to state-space perturbations. Accordingly, this chapter presented the formulation of adversarial resilience to training-time perturbations, and analytically reduced the problem space to facilitate experimental investigations. Accordingly, this chapter studied the effect of various design choices on training-time resilience of DQN, namely: capacity of the experience memory, experience selection mechanism, exploration mechanism, and the discount factor. The findings of this chapter established general guidelines on the selection of mechanisms and parameters at the design level for improving the resilience of DQN agents against training-time perturbations.

Chapter 7 studied the practical limits of adversarial training in improving the robustness of DQN policies. Accordingly, it presented a formal analysis of the efficiency of adversarial training with respect to the proportion of adversarial perturbations to nominal observations used for training. Furthermore, this chapter proposed a novel Adversarially-Guided Exploration (AGE) mechanism to alleviate the sample-inefficiency of blanked adversarial training in DQN agents. To verify the feasibility of this mechanism, experimental evaluations were performed and demonstrated the superiority of AGE to NoisyNet exploration.

Chapter 8 addressed the class of attacks targeting the confidentiality aspect of DRL security. Accordingly, proof-of-concept attacks were developed based on a recent imitation learning technique known as Deep Q-Learning from Demonstrations (DQfD), and their feasibility and security risk were established through experimental studies. Furthermore, this

chapter presented a discussion on a mitigation concept based on the Constrained Randomization of Policy (CRoP), thus motivating further research in this area.

Chapter 9 proposed a novel scheme for the watermarking of DRL policies, which provides a mechanism for the embedding of a unique identifier within the policy in the form of its response to a designated sequence of state transitions. This scheme was shown to have minimal impact on the normal performance of the policy. The feasibility of this proposed scheme was demonstrated via experimental evaluation of watermarking a DQN policy trained in the Cartpole environment. Furthermore, this chapter presented a discussion on alternative applications of the proposed scheme in adversarial settings (i.e., induction of backdoors in policies), as well as in the settings of AI safety (i.e., as a solution to the switch-off problem).

10.2 Frontiers

Sequential decision making and RL are active fields of research with a vast horizon of unsolved challenges. In particular, the study of security considerations and issues in classical and deep RL is still a very young domain filled with research problems and opportunities for foundational contributions. Below, we introduce a number of such problems that build on the findings of the previous chapters.

- *Training-Time Resilience in Policy Search and Actor-Critic Methods*: This dissertation focused on the problem of training-time perturbations in DQN as a baseline for value-iteration methods. Therefore, there is much room to extend the analyses and techniques of Chapters 6 and 7 to other classes of DRL algorithms, namely direct policy search and actor-critic methods. The latter in particular presents the grounds for adopting the techniques of evolutionary game theory to capture and study the dynamics of policy learning under training-time adversarial perturbations. Also, to the extent of our knowledge, the effects of various hyperparameters of such algorithms on training-time resilience of the agent are yet to be studied.

- *Model-Based and Hybrid Methods*: This dissertation focused only on model-free rein-

forcement learning. However, the growing number of real-world applications based on model-based or hybrids of model-free/model-based algorithms necessitates the study of security problems in such systems. For instance, it would be of particular interest to investigate the exploitable vulnerabilities that may arise from the inaccuracies of user-defined models, or those that arise from the inherent shortcomings of online model estimators.

- *Mitigation of Policy Replication*: In Chapter 8, we introduced and motivated the security risks arising from adversarial policy imitation. However, to the extent of our knowledge, there are still no proposals or studies on how to defend DRL policies against such attacks. One promising venue of further research is the Constrained Randomization of Policy (CRoP) idea introduced in Chapter 8. However, efficient randomization of DRL policies in real-world settings will need to satisfy stringent constraints that guarantee both the safety and the efficiency of agents' behavior. Therefore, understanding the limits of feasibility and efficacy of approaches based on CRoP may further enlighten the limits and scopes of such solutions.

- *Security Issues in Multi-Agent DRL*: While the entirety of current security studies on DRL security are focused on single DRL agents, the rapidly growing interest and research in multi-agent deployments of DRL policies amplify the need for understanding the unique security risks that emerge from the interactions of multiple DRL agents. These interactions give rise to not only unique challenges in the security of DRL policies, but also encompass the problems of security engineering in complex adaptive systems [103]. An interesting venue of research in such settings is the problem of verifying, measuring, and engineering the security of mechanisms that govern the interactions of such agents. This venue may benefit from the body of work on quantifying and predicting cascading failures in complex networks and systems.

- *Naturally-Inspired Models*: As argued in [104], the growing complexity of DRL agents and their deployment settings may soon surpass the limits of feasibility for low-level

abstractions and atomic viewpoints. There is however still hope, as higher-level abstractions have shown to be useful in understanding and controlling the highly complex settings of natural phenomenon. Solutions inspired by high-level models such as neuroscience (e.g., [105]) and social sciences (e.g., [106]) present multiple avenues of research on the security of DRL. For instance, the model of reward-hacking as addictive behavior in [105] can be extended to analyze the vulnerability of DRL policies to changes in reward dynamics, while refraining from the computationally expensive techniques of reachability and controllability analysis. Further work on naturally-inspired models may potentially provide many such approaches to solving the security problems of highly-complex DRL systems and settings.

Bibliography

[1] V. Behzadan and A. Munir, "Vulnerability of deep reinforcement learning to policy induction attacks," in *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pp. 262–275, Springer, 2017.

[2] P. W. Glimcher, "Understanding dopamine and reinforcement learning: the dopamine reward prediction error hypothesis," *Proceedings of the National Academy of Sciences*, vol. 108, no. Supplement 3, pp. 15647–15654, 2011.

[3] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[4] C. Szepesvári, "Algorithms for reinforcement learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 4, no. 1, pp. 1–103, 2010.

[5] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *arXiv preprint arXiv:1708.05866*, 2017.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[7] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[8] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 41, 2017.

[9] A. R. Fayjie, S. Hossain, D. Oualid, and D.-J. Lee, "Driverless car: Autonomous driving using deep reinforcement learning in urban environment," in *2018 15th International Conference on Ubiquitous Robots (UR)*, pp. 896–901, IEEE, 2018.

[10] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 3389–3396, IEEE, 2017.

[11] A. Raghu, M. Komorowski, L. A. Celi, P. Szolovits, and M. Ghassemi, "Continuous state-space models for optimal sepsis treatment-a deep reinforcement learning approach," *arXiv preprint arXiv:1705.08422*, 2017.

[12] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 3, pp. 653–664, 2017.

[13] V. François-Lavet, D. Taralla, D. Ernst, and R. Fonteneau, "Deep reinforcement learning solutions for energy microgrids management," in *European Workshop on Reinforcement Learning (EWRL 2016)*, 2016.

[14] C. N. Madu, C.-h. Kuei, and P. Lee, "Urban sustainability management: A deep learning perspective," *Sustainable cities and society*, vol. 30, pp. 1–17, 2017.

[15] M. W. Brittain and P. Wei, "Towards autonomous air trac control for sequencing and separation-a deep reinforcement learning approach," in *2018 Aviation Technology, Integration, and Operations Conference*, p. 3664, 2018.

[16] C. Eppner, S. Höfer, R. Jonschkowski, R. Martín-Martín, A. Sieverling, V. Wall, and O. Brock, "Lessons from the amazon picking challenge: Four aspects of building robotic systems.," in *Robotics: Science and Systems*, 2016.

[17] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[18] V. Behzadan and A. Munir, "Adversarial reinforcement learning framework for benchmarking collision avoidance mechanisms in autonomous vehicles," *arXiv preprint arXiv:1806.01368*, 2018.

[19] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman, "Sok: Security and privacy in machine learning," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 399–414, IEEE, 2018.

[20] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," *arXiv preprint arXiv:1611.03814*, 2016.

[21] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, pp. 317–331, 2018.

[22] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *arXiv preprint arXiv:1702.02284*, 2017.

[23] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[24] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, *et al.*, "Noisy networks for exploration," *arXiv preprint arXiv:1706.10295*, 2017.

[25] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[26] H. V. Hasselt, "Double q-learning," in *Advances in Neural Information Processing Systems*, pp. 2613–2621, 2010.

[27] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning.," in *AAAI*, vol. 2, p. 5, Phoenix, AZ, 2016.

[28] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[29] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, pp. 1889–1897, 2015.

[30] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[32] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, 2016.

[33] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, "Learning to reinforcement learn," *arXiv preprint arXiv:1611.05763*, 2016.

[34] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," *arXiv preprint arXiv:1206.6389*, 2012.

[35] B. A. Nelson, "Behavior of machine learning algorithms in adversarial environments," tech. rep., CALIFORNIA UNIV BERKELEY DEPT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE, 2010.

[36] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European conference on machine learning and knowledge discovery in databases*, pp. 387–402, Springer, 2013.

[37] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[38] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "The space of transferable adversarial examples," *arXiv preprint arXiv:1704.03453*, 2017.

[39] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples (2014)," *arXiv preprint arXiv:1412.6572*, 2014.

[40] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 372–387, IEEE, 2016.

[41] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 506–519, ACM, 2017.

[42] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis.," in *USENIX Security Symposium*, pp. 601–618, 2016.

[43] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing.," in *USENIX Security Symposium*, pp. 17–32, 2014.

[44] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Security and Privacy (SP), 2017 IEEE Symposium on*, pp. 3–18, IEEE, 2017.

[45] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. Tygar, "Antidote: understanding and defending against poisoning of anomaly detectors," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, pp. 1–14, ACM, 2009.

[46] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," in *Asian Conference on Machine Learning*, pp. 97–112, 2011.

[47] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," *arXiv preprint arXiv:1412.5068*, 2014.

[48] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," *arXiv preprint arXiv:1511.04508*, 2015.

[49] V. Behzadan and A. Munir, "The faults in our pi stars: Security issues and open challenges in deep reinforcement learning," *arXiv preprint arXiv:1810.10369*, 2018.

[50] T. Chen, J. Liu, Y. Xiang, W. Niu, E. Tong, and Z. Han, "Adversarial attack and defense in reinforcement learning-from ai security view," *Cybersecurity*, vol. 2, no. 1, p. 11, 2019.

[51] A. G. Barto and T. G. Dietterich, "Reinforcement learning and its relationship to supervised learning," *Handbook of learning and approximate dynamic programming. John Wiley and Sons, Inc*, 2004.

[52] C. P. Pfleeger and S. L. Pfleeger, *Analyzing computer security: a threat/vulnerability/countermeasure approach*. Prentice Hall Professional, 2012.

[53] W. Uther and M. Veloso, "Adversarial reinforcement learning," tech. rep., Technical report, Carnegie Mellon University, 1997. Unpublished, 1997.

[54] T. Everitt, V. Krakovna, L. Orseau, M. Hutter, and S. Legg, "Reinforcement learning with a corrupted reward channel," *arXiv preprint arXiv:1705.08417*, 2017.

[55] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, "Robust deep reinforcement learning with adversarial attacks," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2040–2042, International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[56] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, "Tactics of adversarial attack on deep reinforcement learning agents," *arXiv preprint arXiv:1703.06748*, 2017.

[57] Y. Han, B. I. Rubinstein, T. Abraham, T. Alpcan, O. De Vel, S. Erfani, D. Hubczenko, C. Leckie, and P. Montague, "Reinforcement learning for autonomous defence in software-defined networking," *arXiv preprint arXiv:1808.05770*, 2018.

[58] G. Clark, M. Doran, and W. Glisson, "A malicious attack on the machine learning policy of a robotic system," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pp. 516–521, IEEE, 2018.

[59] E. Tretschk, S. J. Oh, and M. Fritz, "Sequential attacks on agents for long-term adversarial goals," *arXiv preprint arXiv:1805.12487*, 2018.

[60] J. Kos and D. Song, "Delving into adversarial attacks on deep policies," *arXiv preprint arXiv:1705.06452*, 2017.

[61] A. Geramifard, T. J. Walsh, S. Tellex, G. Chowdhary, N. Roy, J. P. How, *et al.*, "A tutorial on linear function approximators for dynamic programming and reinforcement learning," *Foundations and Trends® in Machine Learning*, vol. 6, no. 4, pp. 375–451, 2013.

[62] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," *arXiv preprint arXiv:1608.04644*, 2016.

[63] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo method*, vol. 10. John Wiley & Sons, 2016.

[64] S. Baluja and I. Fischer, "Learning to attack: Adversarial transformation networks," *Proceedings of AAAI-2018. http://www. esprockets. com/papers/aaai2018. pdf*, 2018.

[65] V. Behzadan and A. Munir, "Whatever does not kill deep reinforcement learning, makes it stronger," *arXiv preprint arXiv:1712.09344*, 2017.

[66] V. Behzadan and A. Munir, "Mitigation of policy manipulation attacks on deep q-networks with parameter-space noise," *arXiv preprint arXiv:1806.02190*, 2018.

[67] Y.-C. Lin, M.-Y. Liu, M. Sun, and J.-B. Huang, "Detecting adversarial attacks on neural network policies with visual foresight," *arXiv preprint arXiv:1710.00814*, 2017.

[68] A. J. Havens, Z. Jiang, and S. Sarkar, "Online robust policy learning in the presence of unknown adversaries," *arXiv preprint arXiv:1807.06064*, 2018.

[69] O. Ogunmolu, N. Gans, and T. Summers, "Minimax iterative dynamic game: Application to nonlinear robot control tasks.," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, 2018.

[70] M. Bravo and P. Mertikopoulos, "On the robustness of learning in games with stochastically perturbed payoff observations," *Games and Economic Behavior*, vol. 103, pp. 41–66, 2017.

[71] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings 1994*, pp. 157–163, Elsevier, 1994.

[72] A. Tamar, Y. Glassner, and S. Mannor, "Optimizing the cvar via sampling.," in *AAAI*, pp. 2993–2999, 2015.

[73] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine, "Epopt: Learning robust neural network policies using model ensembles," *arXiv preprint arXiv:1610.01283*, 2016.

[74] J. Morimoto and K. Doya, "Robust reinforcement learning," *Neural computation*, vol. 17, no. 2, pp. 335–359, 2005.

[75] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," *arXiv preprint arXiv:1703.02702*, 2017.

[76] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[77] W. Xu, D. Evans, and Y. Qi, "Feature squeezing mitigates and detects carlini/wagner adversarial examples," *arXiv preprint arXiv:1705.10686*, 2017.

[78] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 135–147, ACM, 2017.

[79] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *arXiv preprint arXiv:1703.00410*, 2017.

[80] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[81] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.

[82] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning*, pp. 1329–1338, 2016.

[83] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, A. Matyasko, V. Behzadan, K. Hambardzumyan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber, and R. Long, "Technical report on the cleverhans v2.1.0 adversarial examples library," *arXiv preprint arXiv:1610.00768*, 2018.

[84] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: a system for large-scale machine learning.," in *OSDI*, vol. 16, pp. 265–283, 2016.

[85] F. Chollet *et al.*, "Keras," 2015.

[86] A. Paszke, S. Gross, S. Chintala, and G. Chanan, "Pytorch," 2017.

[87] V. Behzadan, "Rlattack v2.0," Feb 2019.

[88] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Openai baselines," *GitHub, GitHub repository*, 2017.

[89] J. Leike, M. Martic, V. Krakovna, P. A. Ortega, T. Everitt, A. Lefrancq, L. Orseau, and S. Legg, "Ai safety gridworlds," *arXiv preprint arXiv:1711.09883*, 2017.

[90] A. Rajeswaran, K. Lowrey, E. V. Todorov, and S. M. Kakade, "Towards generalization and simplicity in continuous control," in *Advances in Neural Information Processing Systems*, pp. 6550–6561, 2017.

[91] A. Zhang, N. Ballas, and J. Pineau, "A dissection of overfitting and generalization in continuous reinforcement learning," *arXiv preprint arXiv:1806.07937*, 2018.

[92] U. Shaham, Y. Yamada, and S. Negahban, "Understanding adversarial training: Increasing local stability of supervised models through robust optimization," *Neurocomputing*, vol. 307, pp. 195–204, 2018.

[93] N. Cesa-Bianchi, C. Gentile, G. Lugosi, and G. Neu, "Boltzmann exploration done right," in *Advances in Neural Information Processing Systems*, pp. 6284–6293, 2017.

[94] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, p. 21, 2017.

[95] B. Piot, M. Geist, and O. Pietquin, "Boosted bellman residual minimization handling expert demonstrations," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 549–564, Springer, 2014.

[96] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, *et al.*, "Deep q-learning from demonstrations," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[97] J. Oh, S. Singh, H. Lee, and P. Kohli, "Zero-shot task generalization with multi-task deep reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2661–2670, JMLR. org, 2017.

[98] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *arXiv preprint arXiv:1611.02770*, 2016.

[99] F. Y. Shih, *Digital watermarking and steganography: fundamentals and techniques*. CRC press, 2017.

[100] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding watermarks into deep neural networks," in *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, pp. 269–277, ACM, 2017.

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

[102] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[103] V. Behzadan and A. Munir, "Adversarial exploitation of emergent dynamics in smart cities," in *2018 IEEE International Smart Cities Conference (ISC2)*, pp. 1–8, IEEE, 2018.

[104] V. Behzadan, A. Munir, and R. V. Yampolskiy, "A psychopathological approach to safety engineering in ai and agi," *arXiv preprint arXiv:1805.08915*, 2018.

[105] V. Behzadan, R. V. Yampolskiy, and A. Munir, "Emergence of addictive behaviors in reinforcement learning agents," *AAAI Workshop on Artificial Intelligence Safety (SafeAI)*, 2019.

[106] W. Kool, S. J. Gershman, and F. A. Cushman, "Cost-benefit arbitration between multiple reinforcement-learning systems," *Psychological science*, vol. 28, no. 9, pp. 1321–1333, 2017.