ACADIA: Efficient and Robust Adversarial Attacks Against Deep Reinforcement Learning

Haider Ali

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Applications

Jin-Hee Cho, Chair
Ananthram Swami
Hoda Eldardiry

November 10, 2022
Falls Church, Virginia

Keywords: Deep Reinforcement Learning, Adversarial Learning, Adversarial Attacks

ACADIA: Efficient and Robust Adversarial Attacks Against Deep Reinforcement Learning

Haider Ali

(ABSTRACT)

Existing adversarial algorithms for Deep Reinforcement Learning (DRL) have largely focused on identifying an optimal time to attack a DRL agent. However, little work has been explored in injecting efficient adversarial perturbations in DRL environments. We propose a suite of novel DRL adversarial attacks, called ACADIA, representing AttaCks Against Deep reInforcement leArning. ACADIA provides a set of efficient and robust perturbation-based adversarial attacks to disturb the DRL agent's decision-making based on novel combinations of techniques utilizing momentum, ADAM optimizer (i.e., Root Mean Square Propagation, or RMSProp), and initial randomization. These kinds of DRL attacks with novel integration of such techniques have not been studied in the existing Deep Neural Networks (DNNs) and DRL research. We consider two well-known DRL algorithms, Deep-Q Learning Network (DQN) and Proximal Policy Optimization (PPO), under Atari games and MuJoCo where both targeted and non-targeted attacks are considered with or without the state-of-the-art defenses in DRL (i.e., RADIAL and ATLA). Our results demonstrate that the proposed ACADIA outperforms existing gradient-based counterparts under a wide range of experimental settings. ACADIA is nine times faster than the state-of-the-art Carlini & Wagner (CW) method with better performance under defenses of DRL.

ACADIA: Efficient and Robust Adversarial Attacks Against Deep Reinforcement Learning

Haider Ali

(GENERAL AUDIENCE ABSTRACT)

Artificial Intelligence (AI) techniques such as Deep Neural Networks (DNN) and Deep Reinforcement Learning (DRL) are prone to adversarial attacks. For example, a perturbed stop sign can force a self-driving car's AI algorithm to increase the speed rather than stop the vehicle. There has been little work developing attacks and defenses against DRL. In DRL, a DNN based policy decides to take an action based on the observation into the environment and gets the reward in feedback for its improvements. We perturb that observation to attack the DRL agent. There are two main aspects to developing an attack on DRL. One aspect is to identify the optimal time to attack (when-to-attack?). Second aspect is to identify an efficient method to attack (how-to-attack?). To answer the second aspect, we propose a suite of novel DRL adversarial attacks, called ACADIA, representing AttaCks Against Deep reInforcement leArning. We consider two well-known DRL algorithms, Deep-Q Learning Network (DQN) and Proximal Policy Optimization (PPO), under DRL environments of Atari games and MuJoCo where both targeted and non-targeted attacks are considered with or without the state-of-the-art defenses. Our results demonstrate that the proposed ACADIA outperforms state-of-the-art perturbation methods under a wide range of experimental settings. ACADIA is nine times faster than the state-of-the-art Carlini & Wagner (CW) method with better performance under defenses of DRL.

Acknowledgments

I would like to thank all the faculty and researchers involved in this dissertation and helping me achieve great milestones in my research. This dissertation would not have been possible without the help and encouragement of the following people. Dr. Jin-Hee Cho, my masters thesis advisor and mentor, always supported me when I got stuck or faced challenges. I started my research when I just had little knowledge about deep learning. Dr. Cho gave me plenty of time to learn and review literature of reinforcement learning. Dr. Cho helped me become an independent researcher by giving me a lot of room to bring up my own ideas and build upon them. She taught me reading, writing and presenting research papers. She made me collaborate with faculty and peers, where I got to know the collaboration ethics. She is the best advisor one can ever yearn for in terms of becoming an independent researcher. She always asked me about any personal or academic issues I have and she also focuses on the mental health of her students, which is a gem trait a professor can have. Even in my difficult times when my father was ill, she always gave me hopes and supported me every time. So, It has been a pleasure to work with Dr. Cho and I was extremely fortunate to get such a research advisor and mentor. I got the chance to collaborate with the great researchers of University of Arizona and Old Dominion University (ODU): Dr. Hongyi Wu, Dr. Chunsheng Xin, Dr. Rui Ning and Dr. Jiang Li. They helped me formalize the problem statement and writing the IEEE CNS paper [1]. Later, I collaborated with industry research leader and presidential award winner Dr. Ananthram Swami, who works at Army Research Laboratories (ARL). He taught me writing clear and concise problem statements in my IEEE CNS paper [1]. Later, he is collaborating with me on Federated Learning project as well, where he is always helpful in giving me great ideas and asking great questions to formalize

and solve the problem well. I would also like to thank Dr. Hoda Eldardiry who helped me mainly in my Federated Learning project as well as in understanding my own research well in terms of real world scenarios. I also want to thank Mohannad Al-Ameedi, who helped me writing the paper, brainstorming the idea and formalizing the problem. Mohannad has always been a great mentor for every question I had in terms of career or education. I want to thank Ahmad Faraz Khan, who has helped me in experiments of Federated Learning Project. This dissertation has been partly supported by Virginia's Commonwealth Cyber Initiative (CCI) and NSF Grant 2107450. This work was accepted and presented at 2022 IEEE Conference on Communications and Network Security (CNS) [1].

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Deep Reinforcement Learning (DRL) algorithms learn policies to guide a DRL agent to take optimal actions based on the state of the environment. These algorithms have successfully achieved high performance on various complex as well as critical tasks, such as robotics [3], autonomous vehicles [22], and cybersecurity [5]. A policy, a probabilistic distribution of actions by the DRL agent, is learned by Deep Neural Networks (DNN) to approximate the action-value function. The vulnerabilities of DNNs to adversarial attacks have been significantly studied [16, 38, 46] to mitigate the impact of the adversarial attacks when the DNNs are exploited by the adversaries. Common adversarial examples include adversarial perturbations imperceptible to humans but fooling DNNs easily in the testing or deployment stage [46]. A self driving can be fooled by a small perturbation in traffic signals to cause accidents as shown in Figure 1.1. For example, a perturbed stop sign can be perceived as 45 speed limit and perturbed 35 speed limit traffic signal can be perceived as 85 speed limit.

Researchers have explored various attacks and defenses for supervised DNN applications, such as image classification [16] or natural language processing [2]. However, adversarial attacks and defenses are largely unexplored in DRL environments. DRL also has numerous critical safety and security applications and accordingly drew our attention to the need for robust DRL. For robust DRL, there is a prerequisite of developing efficient, effective,

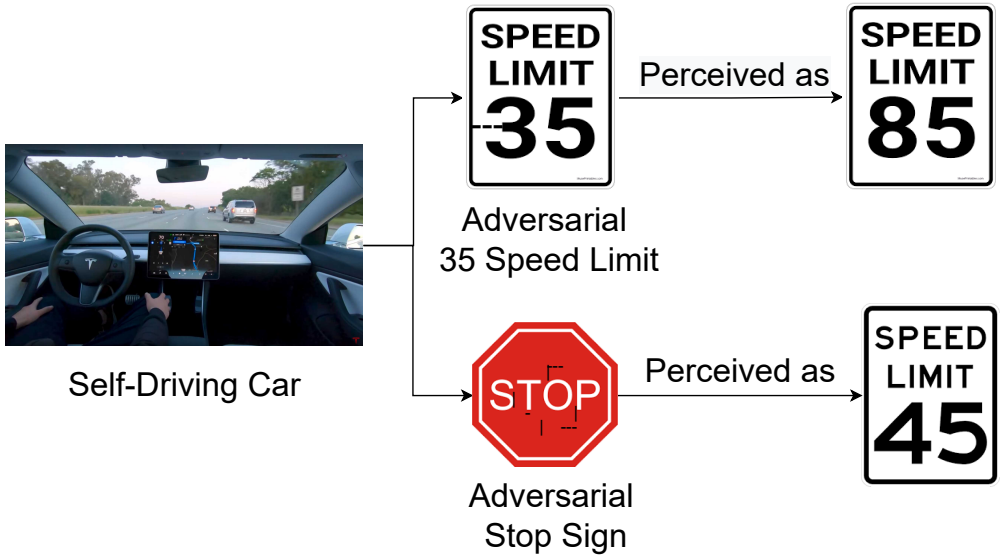


Figure 1.1: Motivation of this dissertation: Perturbations in traffic signals can turn them into hazardous traffic signals. For example, Stop Sign can be perceived as 45 Speed or 35 speed can be perceived as 85 speed limit using small perturbations.

and robust adversarial attacks which can be used to evaluate the robustness of defense mechanisms.

In the adversarial machine learning research community, researchers have developed adversarial attacks in DRL by answering the following two questions: (1) How to attack? and (2) When to attack? The first how-to-attack question is related to what perturbation method should be used for disrupting the state during an episode. The second when-to-attack question is associated with identifying an optimal time to attack during an episode. In this work, we aim to answer how-to-attack by proposing ACADIA, a set of novel adversarial AttaCks Against Deep reInforcement leArning. To be specific, the goal of this work is to develop robust and fast attacks by generating effective and efficient adversarial states in DRL settings.

Unlike DNN settings, there are non-trivial challenges in developing efficient and effective adversarial states under various DRL settings as shown in Figure 1.2: First, there is no

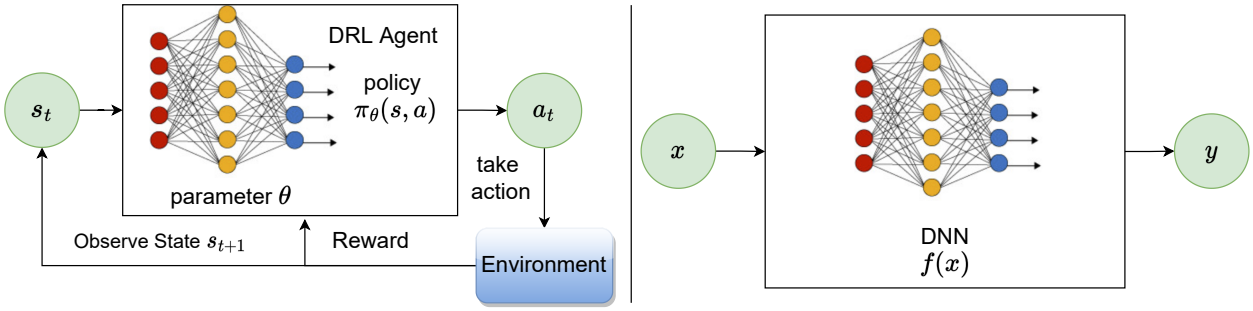


Figure 1.2: Differences between DRL (left) and DNN (right) processes. Unlike single classification in DNN, DRL is a continuous learning process driven by reward as a feedback given by the environment.

stationary dataset and correct action available in DRL settings. Instead, there is a dynamic series of steps where the DRL agent is continuously learning through a reward upon taking a series of actions. This means that the DRL agent is continuously tackling multiple situations in an episode. That is, attack success in one step does not guarantee attack success in future steps. Second, unlike DNN environments, there can be discrete as well as continuous action spaces in the DRL depending upon the environment. Third, defenses in DRL work on different principles as compared to the defenses in DNNs. Previous attack variants are not comprehensive enough to be used in various DRL settings.

1.2 Key Contributions

In this work, we propose ACADIA that integrates momentum, ADAM optimizer, random initialization and Fast Gradient Sign Method (FGSM) to effectively and efficiently solve the challenges faced in DRL settings. The existing state perturbation attacks are not comprehensive enough to be used in various DRL settings. Our proposed ACADIA is the first novel effective and efficient perturbation attacks under the DRL. We compare the performance of our proposed ACADIA with those of state-of-the-art adversarial attacks in DRL application

environments under various settings with or without defenses when attackers may perform targeted or non-targeted attacks. Via extensive comparative performance analyses, we validated the performance of ACADIA in terms of attack success rate metrics (ASR), average attack execution time per perturbation (AET), and average reward (AR) to the DRL agent.

We made the following key contributions in this work:

1. We develop a framework, called ACADIA, that provides a suite of efficient and effective adversarial attacks to disrupt DRL operations. The ACADIA framework provides novel attacks named Iterative ACADIA (iACADIA), ADAM-based Iterative ACADIA (aiACADIA), and Momentum-based, Iterative ACADIA (miACADIA), which generate fast and robust adversarial perturbations to compromise a DRL agent under either targeted or non-targeted perturbation attack(s).

2. We propose a novel metric to compute attack success rate (ASR) in continuous environments using Mean Absolute Error to show the per-step attack performance in DRL environments.

3. We conduct extensive experiments based on two well-known DRL algorithms of Deep-Q Learning Network (DQN) and Proximal Policy Optimization (PPO) on Atari games (i.e., Pong, BankHeist, and RoadRunner) and MuJoCo environments with/without state-of-the-art defenses in a given DRL algorithm.

4. Our results show that our proposed attacks outperform overall on ASR, AET, and AR. The proposed miACADIA performs the best among all. All of our variants of ACADIA significantly outperform the Carlini & Wagner method (CW) on AET while maintaining comparable ASR and AR values. In particular, our proposed ACADIA outperforms the state-of-the-art perturbation methods on ASR and AR metrics when the DRL uses defenses.

1.3 Structure of This Dissertation

The rest of this dissertation is structured as follows. Section 2 explains how the problem we aim to tackle is formulated in this work. Section 3 provides the overview of the related work in terms of adversarial attacks in DNNs and DRL environments. Section 4 describes brief background information that are mainly used to develop our ACADIA. Section 5 describes the details of the proposed ACADIA. Section 6 gives the details of the experiment setup including setting for attacks and defenses, metrics, application background, parameterization of key design variables, and comparing schemes considered for comparative performance analysis. Section 7 demonstrates the simulation results and the corresponding analyses of the observed results. Section 8 concludes the dissertation, suggests future research directions, and presents the publications accepted or submitted during my research.

Chapter 2

Problem Statement

A DRL agent interacts with the environment and learns policy π to choose action a given state s and obtains reward $r(s, a)$. This policy π can be a probabilistic model $\pi(\mathrm{s}, \mathrm{a}) \sim [0, 1]$, which gives the probability of taking action a given state s. The π can also be a deterministic model $s : a = \pi(\mathrm{s})$. The goal of the DRL agent is to maximize the cumulative reward R_o by learning an optimal policy π^*. The reward that the DRL agent aims to maximize is the expected discounted reward over next $T - 1$ time steps, represented by:

$$R_o = \sum_{t=0}^{T-1} E_{a_t \sim \pi(s_t)} \Big[\gamma^t r(s_t, a_t) \Big]. \tag{2.1}$$

where $\gamma \in [0, 1)$ is the discount factor. An attacker aims to reduce reward R_o by adding perturbation δ_t to the agent's observation s_t to mislead the agent to take non-optimal action a_t. The attacker has to generate perturbation δ as small as possible to be undetected. At each time step, the attacker may or may not choose to add perturbation δ to state s_t based on its strategy. In this way, the expected cumulative reward after the attack is given by:

$$R_{\mathrm{adv}} = \sum_{t=0}^{T-1} E_{a_t^{\mathrm{adv}} \sim \pi(s_t + u_t \delta_t)} \Big[\gamma^t r(s_t, a_t) \Big], \tag{2.2}$$

where u_t at given time t is 1 if the attacker injects perturbation; otherwise, we set $u_t = 0$.

In this work, we aim to solve the following problems:

- We aim to design the perturbations $\delta_0, \delta_1, \ldots, \delta_{T-1}$ to force the DRL agent to take the corresponding adversarial actions, denoted by $a_0^{\text{adv}}, a_2^{\text{adv}}, \ldots, a_{T-1}^{\text{adv}}$.

- These adversarial actions should reduce the reward of the DRL agent R_o, and the reward of the DRL agent under defense, R_{defense} in a non-targeted setting. The reduced reward is represented by R_{adv} in Eq. (2.2).

- These adversarial perturbations should generate the required targeted actions in the targeted attack setting.

- We aim to find each perturbation δ_t in a realistic time possible that could work under real settings.

Chapter 3

Review of Literature

In this section, we provide a brief overview of adversarial attacks developed in Deep Neural Networks (DNN) and Deep Reinforcement Learning (DRL) settings.

3.1 Adversarial Attacks in Deep Neural Networks (DNN)

Perturbation attacks seek to perturb the input to a DNN so as to cause targeted or non-targeted misclassification. A key aspect of generating these attacks involves the computation of gradients. Goodfellow and et al. [16] proposed the Fast Gradient Sign Method (FGSM), which is efficient but later, shown to be less robust against state-of-the-art defenses, and could not guarantee 100% ASR. The Carlini & Wagner (CW) method [7] is a well-known effective attack, guaranteeing 100% ASR; however, it is slow. Naïve FGSM provided the basis to build more sophisticated and better variants of FGSM in terms of ASR, including Randomized FGSM (RFGSM) [39], Diversity Iterative FGSM (DI-FGSM) [43], Momentum Iterative FGSM (MI-FGSM) [12], and ADAM Iterative FGM (AI-FGM) [42]. The Projected Gradient Descent (PGD) [29] was also proposed as a variant of Iterative FGSM (I-FGSM) to enhance its robustness. AutoAttack [11] is an extension of PGD attack which tries to cater to the suboptimal step size and problems of the objective function. AI-FGM was also proposed as black-box attacks in DNNs [45]. So far, MI-FGSM and PGD attacks are considered the most efficient and robust state-of-the-art adversarial examples in DNNs. In particular, AI-

FGM, RFGSM, DI-FGSM, AutoAttack and MI-FGSM were only designed and evaluated in the context of DNNs; however, to the best of our knowledge, have not been evaluated in DRL settings. These baselines still were not able to outperform ACADIA due to their inability to perform under all the varied settings of discrete/continuous DRL, targeted/non-targeted attacks and defense/no-defense.

In DNN settings, apart from perturbation based attacks, there exist other type of attacks called backdoor attacks [17, 18, 25, 26, 44] in which triggers are mostly implanted in training time in the datasets. We do not have a dataset in DRL, so, these attacks are not related to this dissertation.

3.2 Adversarial Attacks in Deep Reinforcement Learning (DRL)

In DRL, attackers exploit states, reward and policy to disturb the DRL process. State perturbation attacks are the most common ones in DRL. For example, Huang et al. [19] extended the adversarial attacks to DRL for the first time by crafting the existing Fast Gradient Sign Method (FGSM) perturbation method to algorithms like DQN to all time steps during an episode. This attack is often called uniform attack since it attacked on all time steps during an episode. They used Atari Games as application to target the policy by using adversarial states. Behzadan and Munir [6] presented policy induction attacks which used the similar notion of Uniform Attack by using FGSM as well as JSMA as its perturbation methods. This work proved that adversarial examples are transferable across DRL algorithms like DQN, which is the basis of policy induction attack.

Kos and Song [23] investigated the effectiveness of adversarial perturbations and random noise on DRL algorithms. They proved experimentally that FGSM based adversarial examples are more effective than random noise. They showed that we can achieve the same

performance with low attack rate using the value function. In other words, all moments during an episode might not reduce reward when attacked. They demonstrated this using different controlled attacks on A3C playing Atari Games.

Later, the state-of-the-art DRL attacks mainly identified an optimal time to attack the DRL agent [27]. [27] mainly focused on finding the optimal time to attack during an episode because attacking on all time steps during an episode would be easily detectable. For this purpose, they proposed strategically timed attack and enchanting attack. They used Carlini & Wagner (CW) perturbation method to perturb those optimal time steps. With only 25% attack rate, they were able to minimize the reward as well as achieve effective success rate. They demonstrated their attacks on A3C and DQN playing Atari Games.

Sun et al. [37] can be considered as a follow-up work of strategically timed attack in [27]. They also tried to perturb in minimum number of critical moments, in order to cause severe damage to agent while remain undetectable. First attack was called critical point attack where future states are predicted using a model. Strategies were devised and each strategy was assessed to select the optimal strategy. Second attack was antagonist attack where critical time steps of attacking were identified using a domain agnostic model. They were able to achieve fair reward reduction with attacking using Carlini & Wagner perturbation to only fewer than 5 critical steps during an episode in TORCS, Atari Games and MuJoCo environments.

Tretschk et al. [40] used Adversarial Transformer Network (ATN) to generate sequence of adversarial inputs to minimize the reward of DRL agent in a white-box setting. It was also focused on when-to-attack as in strategically timed attack of [27]. They showed the effectiveness of their attack with DQN playing Atari Pong game. Most of such strategy (when-to-attack) based attacks used the state-of-the-art CW perturbation method [7] to perturb those critical moments. CW was very slow and less successful under DRL with

defense(s) as demonstrated in our study.

Chen et al. [9] presented Common Dominant Adversarial Examples Generation Method (CDG) which attacks by generating high confidence adversarial examples using obstacles in the environment. They showed that this method was successful 99.9% of the time in a path finding problem solved using A3C in a white-box setting. This attack was considered successful if it was able to delay the DRL agent or stop it from reaching the destination. Similarly, [4] used weaknesses in DQN to craft attacks.

Clark et al. [10] presented an attack that tampered the sensory data to force a robot to follow a wrong path in a dynamic autonomous robot environment. They also proved that once they stopped tampering, robot returns to its actual path. This showed that a hidden attack can be crafted with no evidence left. They used DQN playing Autonomous Robot Emulation (JAV) in a white box setting where access to trained policy is required. Xiao et al. [41] proposed two online sequential attacks for attacking the environment in DRL using model querying instead of back-propagation as in FGSM. These two attacks were based on two methods of model querying: adaptive dimension sampling based finite difference method (SFD), and optimal frame selection method (OFSM). Due to model querying, they were even faster than FGSM-based attacks. They exploited the temporal consistency of the states while attacking. They also have given other attacks which targets observation and action instead of environment. TORCS is used with DDPG and DQN agents to show the attack's effectiveness in both white box and black box settings. None of these attacks seem generic enough to be widely used across varied settings of DRL since they show performance on limited settings/environments such as PathFinding.

Hussenot et al. [20] claimed that previous attacks were mostly not realistic or computationally expensive. They used a read-only environment setting where they can only read the observations through the environment. They proposed two attacks called per-observation

attack and universal mask attack. Per-observation attack added the perturbation against every observation observed in the environment by the agent. Universal mask attack added only one perturbation to all the observations, which is created at the start of the attack. In non-targeted attack, they showed that FGSM was efficient and effective. However, in targeted attacks, they showed that FGSM was not able to generate effective and imperceptible perturbations. They showed the effectiveness of their attacks on DQN and Rainbow playing Atari Games.

Fast perturbation methods, such as naïve FGSM [16], have been used in DRL [19]; however, naïve FGSM is easily detectable. Another variant of FGSM, called PGD, is one of the state-of-the-art DRL attacks; it is fast and has better ASR than CW under defenses. However, the state-of-the-art defenses in DRL [14, 32, 48] have recently challenged the robustness of PGD-based attacks [29], which were originally designed for DNNs. In addition, we observed in our experiments that PGD and its extensive variants (e.g., AutoAttack) did not perform well under targeted attacks.

Kiourti et al. [21] presented a backdoor or trojan attack with the access of training phase in DRL environment. By only changing 0.025% of the training data, trojan was induced in the data. Whenever this backdoor was triggered, the DRL agent performed poorly. This was one of the few attempts to leverage backdoor attack in DRL setting. They also showed that their attack worked greatly under current defenses of DRL.

Gleave et al. [15] formed a zero-sum game between an adversarial agent and the legitimate agent by introducing adversarial agent in the same environment. Natural adversarial observations can be created by using such an adversarial agent to make the actual DRL agent follow the adversarial policy. Frozen deployed models can help mitigate the negative efforts of such adversaries. Win rate was used in such games to show the effectiveness of their attacks instead of average reward.

Chen et al. [8] claimed that in DRL, we need new model extraction and imitation learning techniques instead of DNN techniques. They used a recurrent neural network (RNN) to infer the training algorithm of DRL agent used in a black-box setting based on the predicted actions. After knowing the model, they used imitation learning to get a copy of the victim model. Simply by extracting the model, more powerful adversarial examples can be generated in especially black-box setting. They used several algorithms like DQN, PPO, A2C to attack Atari Pong and Cart-Pole environments.

Figura et al. [13] proposed adversarial attacks against the network that uses consensus-based multi-agent RL. They showed that an adversarial agent can convince all participating agents in the consensus network to implement its desired adversarial policy. They showed the theoretical asymptotic convergence of their algorithm to a consensus result in favor of an adversary. In their experimental setting, decentralized Actor Critic is used by a network of DRL agents in a white-box setting. This attack was different than other mainstream research in adversarial DRL where either state, environment or reward was compromised using adversarial perturbations.

Liu and Lai [28] presented an Action Poisoning Attack (LCB-H). Previous works focused on either observation poisoning or environment poisoning. This was the first work to present action poisoning attack, where adversary tried to change the actual action by the agent. They proposed an attack scheme called LCB-H, which can force an efficient agent to choose an adversarial action frequently. This attack was applied on model-free DRL algorithms of UCB-H, UCB-B and UCBV1-CH in periodic 1-d grid world application with white-box as well as black-box settings. They also analyzed the computational complexity of their attack, which was either sub-linear or logarithmic.

Qiaoben et al. [34] presented a Tentative Frame Attack which was another effort to answer the question of when-to-attack in specifically DRL continuous environments. They claimed

that previous works lacked theoretical principles while finding the right time steps to attack. So, they gave a theoretical framework called Strategically-timed State-adversarial MDP (SS-MDP) to find the optimal frame to attack the agent. Frame attack strategy was trained using these optimal frames. They called their Tentative Frame Attack a version of strategically timed attacks. PPO with MuJoCo environments were used in a white-box setting to show the efficacy of their attacks as compared to other state-of-the-art strategically timed attacks.

Qu et al. [35] presented Frame-Correlation Economical Attacks. Previous attacks attacked a single frame at a time and ignored the relationship between neighboring states in a MDP. This resulted in high computational cost and this did not work in real-word scenarios due to limited time to attack. Transferability between these frames can be used to save time and build efficient attacks in no time, making realistic attacks possible. They introduced three types of frame-correlation transfers (FCTs). These FCTs include anterior case transfer, random projection-based transfer, and principal components-based transfer. These FCTs used genetic algorithm with different computational complexities in generating adversaries. They showed the effectiveness of their realistic real-time attacks using four state-of-the-art DRL algorithms with Atari Games in black-box setting.

Pattanaik et al. [33] proposed three types of attacks which differed on using either different type of perturbation method or using loss function in them. First one used random noise, second one was gradient-based (GB) with a loss function focusing on worst possible discrete action, and third one was an enhanced version of the second one with Stochastic Gradient Descent (SGD). They demonstrated that GB attacks were better than FGSM based attacks when DDPG and DDQN were playing Cart Pole, Mountain Car and MuJoCo environments. But, GB is also not robust due to lack of randomness. Therefore, there is a critical need to develop scalable, effective, and robust state perturbation attacks in DRL settings.

Chapter 4

Preliminaries

This section provides a brief overview of DRL and adversarial state generation methods considered in this work.

4.1 Deep Reinforcement Learning

Reinforcement learning (RL) algorithms optimize the expected cumulative reward by training a policy π. This policy can be a deterministic or probabilistic function that maps state s to action a, $\pi : S \rightarrow A$, where S and A are state and action spaces, respectively. In Deep RL (DRL), this policy function π is learned by a neural network. Deep Q-Networks (DQN) [30] and Proximal Policy Optimization (PPO) [36] are two well-known DRL algorithms, which are also evaluated in our work.

4.2 Gradient Perturbation Methods

We leveraged the following methods to develop our attacks.

4.2.1 Fast Gradient Sign Method (FGSM)

FGSM [16] focuses on attack efficiency to design adversarial examples, rather than optimal performance of adversarial examples. For an image x, a perturbed image x' according to FGSM is:

$$x'_{\text{FGSM}} = x + \epsilon \cdot \text{sign}(\nabla \text{loss}(h(x), y_{\text{true}})), \tag{4.1}$$

where ϵ is a parameter to make the perturbation small enough to be less noticeable. In DRL, the observation or state s is considered as an image x to compute FGSM based image or state. The loss function can be a cross-entropy function. FGSM uses the linear approximation of the model and solves the maximization problem in a closed form, which makes this method very fast. There are three kinds of norm constraints, L_0, L_1, and L_∞, which are used in the literature to constrain the perturbation to be undetectable. The norm constraint can be represented by equation:

$$||x - x'||_n < \epsilon. \tag{4.2}$$

where n is 0, 1 or ∞.

4.2.2 Iterative Fast Gradient Sign Method (I-FGSM)

I-FGSM [24] takes multiple small steps of size α in the direction of the gradient. Starting with $x'_0 = 0$, on every iteration i, it performs:

$$x'_i = x'_{i-1} - \text{clip}_\epsilon(\alpha \cdot \text{sign}(\nabla \text{loss}_{F,t}(x'_{i-1}))). \tag{4.3}$$

4.2.3 Randomized FGSM (RFGSM)

RFGSM [39] adds a small random step, α, to FGSM to escape the non-smooth vicinity of the data point before linearizing the model's loss. The perturbed state is computed as:

$$x^{\text{adv}}_{RFGSM} = x' + (\epsilon - \alpha) \cdot \text{sign}(\nabla_{x'} \text{loss}(x', y_{\text{true}})), \tag{4.4}$$

where

$$x' = x + \alpha \cdot \text{sign}(\mathcal{N}(0^d, I^d)). \tag{4.5}$$

4.2.4 Momentum Iterative FGSM (MI-FGSM)

MI-FGSM [12] is a variant of FGSM that integrates the momentum on each step of an I-FGSM to escape from poor local maxima and stabilize update directions. Starting with $x'_0 = 0$, at every iteration i, the perturbation update is:

$$g_i = \mu \cdot g_{i-1} + \frac{\nabla \text{loss}_{F,t}(x'_{i-1})}{||\nabla \text{loss}_{F,t}(x'_{i-1})||_1}, \tag{4.6}$$

$$x'_i = x'_{i-1} - \text{clip}_\epsilon(\alpha \cdot \text{sign}(g_i)), \tag{4.7}$$

where μ is the decay factor and g_i is the accumulated gradient at iteration i.

4.3 Definitions

We use the following terms to indicate type of attacks in terms of whether an attacker aims to achieve a target goal or not. In addition, we define what we mean by 'robustness' of an adversarial attack in DRL settings considered in this work.

- Non-Targeted Attack in DRL: In this attack, the attacker aims to reduce the reward of the DRL agent by crafting a perturbation that would lead to a sub-optimal action.
- Targeted Attack in DRL: In this attack, the attacker seeks to perturb the state so that the DRL agent takes a specific targeted action. For example, in Atari Pong, a targeted attack may aim to make the DRL agent take the targeted action 'up' at a given point.
- Robustness of DRL Attack: A DRL attack is said to be robust if it achieves high ASR and low AR under defenses.

We will next review the state-of-the-art perturbation methods before describing our proposed method.

Chapter 5

Proposed Approach: ACADIA

To develop robust, effective, and efficient perturbations, we propose the ACADIA framework, representing AttaCks Against Deep reInforcement leArning, that provides the following three novel attacks: iACADIA for Iterative ACADIA, aiACADIA for ADAM-based iACADIA, and miACADIA for Momentum-based iACADIA. ACADIA integrates RMSProp, momentum, random initialization and FGSM to maximize the effectiveness and efficiency of the proposed adversarial attacks in DRL settings. ACADIA performs state perturbation attacks on observation and reward of the DRL agent as shown in Figure 5.1.

All of our attacks differ in computing the gradient of the loss function and changing a step size through a number of steps. Now we discuss a generic version of our attacks where we will attack a single state during an episode. The details of the three attacks are described as follows:

5.1 Generic ACADIA

Let A be either iACADIA, miACADIA or aiACADIA which differs in computing a gradient g^A of the loss function. We consider the true label, a_t^{true}, the action produced by the DRL policy at a time step t during an episode, represented by:

$$a_t^{\text{true}} = \pi(s_t^{\text{true}}), \tag{5.1}$$

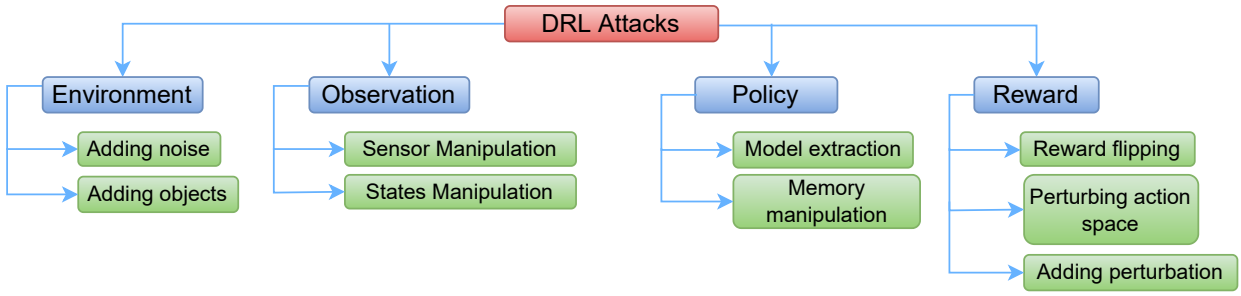


Figure 5.1: Classification of the existing DRL attacks. ACADIA comes under both States Manipulation to perturb the observation and adding perturbation to reduce the reward.

where s_t^{true} is the original non-adversarial state at time step t. At each step i of the attack until a maximum number of steps m, we calculate adversarial state s_i^{A} by taking a step of size α in the direction of the gradient g_i^A. Starting with adding the random step of size, α, to s_t^{true} in the random direction given by normal distribution $\mathcal{N}(0^d, I^d)$, called a random initialization:

$$s_0 = s_t^{\text{true}} + \alpha \cdot \text{sign}(\mathcal{N}(0^d, I^d)), \tag{5.2}$$

on every iteration i until m, the attacker computes:

$$s_i^A = s_{i-1}^A + \alpha \cdot \text{sign}(g_i^A), \tag{5.3}$$

$$a_t^{\text{adv}} = \pi(s_{m-1}^{\text{A}}). \tag{5.4}$$

We summarize the key procedures of generic iACADIA in Figure 5.2.

5.2 iACADIA

iACADIA is the basic version of our attacks where we simply compute a regular gradient

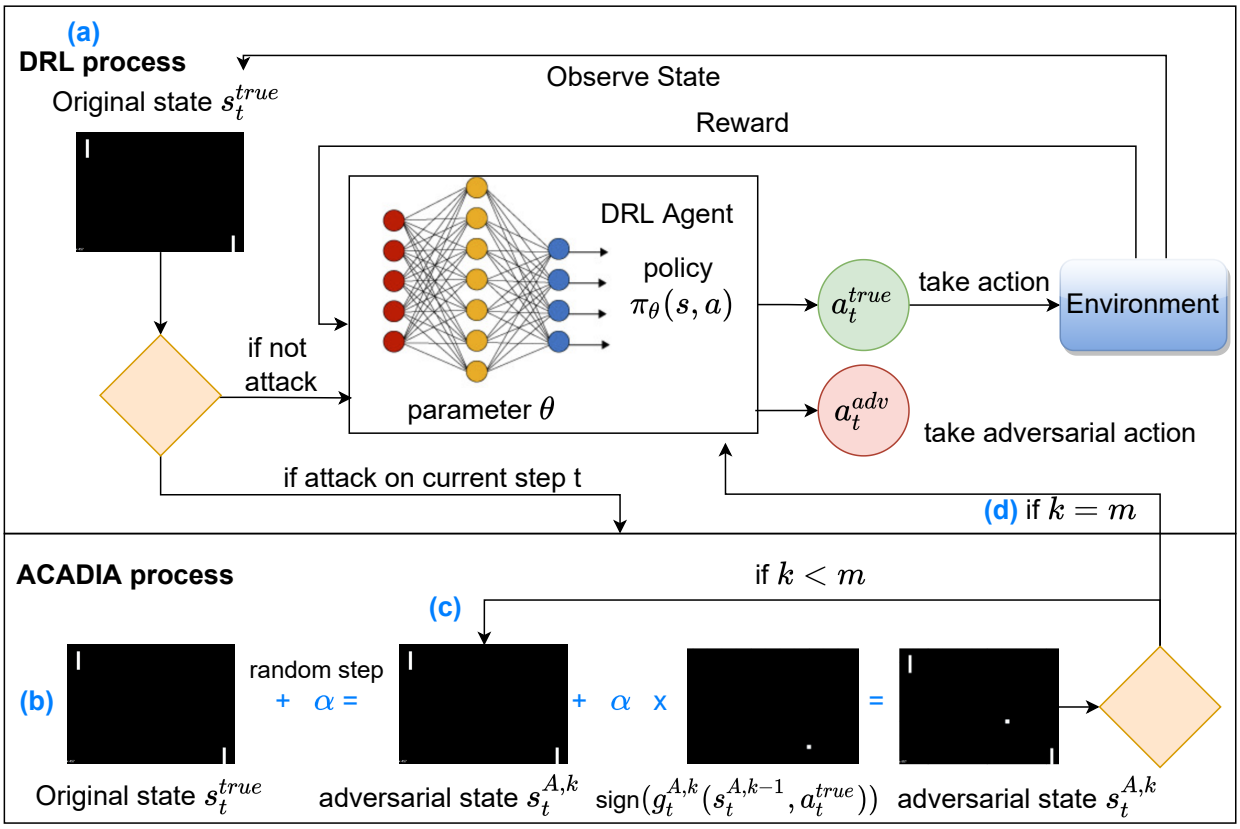


Figure 5.2: An overview of Generic ACADIA. It can be either iACADIA, miACADIA or aiACADIA depending upon gradient g^A in (c). It attacks a frame t during an episode to compromise a DRL agent following: (a) DRL agent observes a true non-adversarial state s_t^{true} and takes non-adversarial, true action, a_t^{true}; (b) Start by adding a random step of size α to s_t^{true}; (c) Until the number of steps, m, compute the adversarial state s_i^{A} by calibrating either basic gradient $g_i^{iACADIA}$, momentum-based accumulated gradient $g_i^{miACADIA}$ or ADAM based gradient $g_i^{aiACADIA}$, using s_{i-1}^{A} and a_t^{true}, and then clipping it; and (d) Compute adversarial action, a_t^{adv}, by giving a final adversarial state s_{m-1}^{A} to the DRL agent.

without incorporating any momentum or RMSProp term by:

$$g_i^{iACADIA} = \nabla_{s_{i-1}^A} \text{loss}(s_{i-1}^A, a_t^{\text{true}}), \tag{5.5}$$

where A is iACADIA here and loss function can be the cross entropy or Mean Squared Error (MSE) between the targeted/original action and the adversarial action. We use the cross entropy loss in our experiments.

5.3 miACADIA

This attack incorporates momentum in computing a gradient with $g_0^{miACADIA} = 0$ by:

$$g_i^{miACADIA} = \mu \cdot g_{i-1}^{miACADIA} + \frac{\nabla_{s_{i-1}^A} \text{loss}(s_{i-1}^A, a_t^{\text{true}})}{||\nabla_{s_{i-1}^A} \text{loss}(s_{i-1}^A, a_t^{\text{true}})||_1}, \tag{5.6}$$

where A is miACADIA here, μ is a decay factor and $g_i^{miACADIA}$ is an accumulated gradient incorporating momentum at iteration i.

5.4 aiACADIA

This attack uses the ADAM optimizer that incorporates momentum m and RMSProp v in computing a gradient with $m_0 = 0$ and $v_0 = 0$ and is performed by:

$$g_i = \frac{\nabla_{s_{i-1}^A} \text{loss}(s_{i-1}^A, a_t^{\text{true}})}{||\nabla_{s_{i-1}^A} \text{loss}(s_{i-1}^A, a_t^{\text{true}})||_1}, \tag{5.7}$$

$$m_i = \mu_1 \cdot m_{i-1} + (1 - \mu_1) \cdot g_i, \tag{5.8}$$

$$v_i = \mu_2 \cdot v_{i-1} + (1 - \mu_2) \cdot g_i^2, \tag{5.9}$$

$$g_i^{aiACADIA} = \frac{m_i}{\sqrt{v_i + \beta}}, \tag{5.10}$$

where A is aiACADIA here, μ_1 and μ_2 are decay factors and β is a very small number to make the denominator non-zero. This attack does not use the *sign* function in Eq. (5.3). Therefore, this equation can be re-written by:

$$s_i^{aiACADIA} = s_{i-1}^{aiACADIA} + \alpha \cdot (g_i^{aiACADIA}). \tag{5.11}$$

Removing the *sign* function here means adapting the step size α. This is exactly similar to adapting the learning rate to converge to global minima smoothly.

Eq. (5.3) means that we move policy π away from an optimal action by using the direction of the gradient. We make it iterative to take multiple gradient direction steps. Multiple gradient steps move the policy away from the optimal action, contributing to generating high ASR and low AR. Adding a random step α at the start contributes to high ASR under defense as it allows it to escape the non-smooth vicinity of the data point before linearizing the model's loss.

Eq. (5.6) shows the addition of a momentum term to the gradient at each step. This addresses the problem of the attack becoming stuck at a poor local maximum and gives the gradient a necessary boost, called momentum, to try to reach the global maximum. Momentum also stabilizes the gradient updates. We also observe in the experiments that momentum helps in gaining high ASR and low AR in the complex settings of DRL especially in targeted attacks and continuous environments of DRL. In summary, miACADIA uses the novel combination of random start, iterative steps of size α and momentum in a DRL setting.

miACADIA may not converge as it uses a constant step size of α. By incorporating RMSProp, we are changing the step size at every iteration for better and smoother convergence. In conclusion, aiACADIA uses the novel combination of random start, iterative steps using RMSProp, and momentum in a DRL setting.

Figure 5.3 highlights the key differences between FGSM variants and our proposed ACADIA. Random initialization helps in robustness and momentum together with RMSProp helps in smoother and faster convergence. FGSM-based perturbation methods are efficient and can work under realistic settings. Thus, we present the novel perturbation method using these features to make it robust, effective and efficient. PGD (i.e., the baseline most similar to

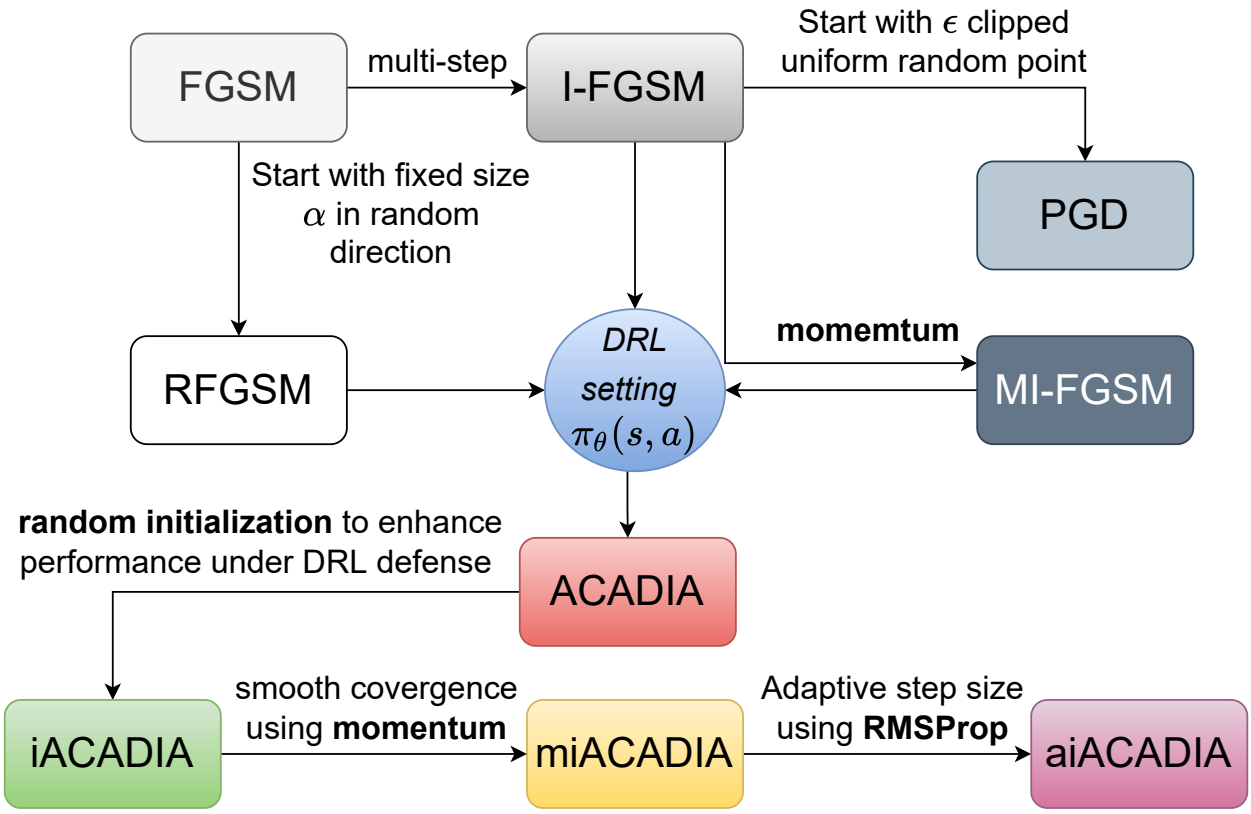


Figure 5.3: Difference between FGSM variants and our proposed aiACADIA, iACADIA and miACADIA.

ours) differs from ACADIA in that ACADIA takes a fixed size step α in a random direction instead of uniformly choosing a random point. PGD does not incorporate momentum and RMSProp, which is why it does not converge well especially in targeted attacks and complex settings of DRL. PGD does have some random initialization, increasing robustness. Other PGD-based attacks such as AuotAttack have similar problems of convergence. In addition, AutoAttack is time-consuming as it is an ensemble of different PGD-based attacks and BlackBox attacks. CW is not only time-consuming but also not robust because it takes too long, increasing detectability. MI-FGSM does not incorporate random initialization and RMSProp, which is why it may not converge well and is not robust. ACADIA provides a comprehensive set of features to efficiently and effectively enhance the performance under

targeted/non-targeted attack types, discrete/continuous DRL and defense/no-defense.

We do not employ any strategy to find an optimal time to attack during an episode as our major focus is to propose a comprehensive state perturbation attack in DRL (how-to-attack). Instead, we attack at every step during an episode. In addition, this approach allows a fair comparison with other state-of-the-art attacks because 'when-to-attack' can change the attack rate while we need to fix it to consider overall behavior across whole episode. Thus, the number of attacks during an episode equals the episode length T i.e. 100% attack rate.

Chapter 6

Experiment Setup

In this section, we provide details of the experimental setting used for our analysis in terms of metrics used to measure efficiency and effectiveness of attacks, the background of the DRL application environments (i.e., Atari and MuJoCo), parameterization of design variables in ACADIA and other existing state-of-the-art attacks, and comparing schemes for performance analysis.

6.1 Setting for Attacks and Defenses

6.1.1 Setting for Attacks

We consider a white-box attack where the attacker does not have access to the training setup. However, at run time the attacker has access to the trained DRL model and has access to the state which it can perturb. In our experiments, we utilize trained DQN, and PPO models using the implementation of [31] and [47] to play within several environments. DQN is used in Atari games since it is more suitable for discrete environments. For MuJoCo Walker, we use PPO, which is well-suited for continuous environments.

We assume that an attacker has the computational power equivalent to the commodity CPUs. To this end, we use the Google Colaboratory CPU (AMD EPYC 7B12, 2 CPUs @ 2.3 GHz, 13 GB RAM) for DQN experiments and personal computers for PPO due to the

inherent limitation of Google Colab to run MuJoCo. PPO experiments are run on Intel(R) Core(TM) i7-9750H CPU @ 2.60 GHz 32 GB RAM. We use the libraries of PyTorch and Open AI Gym for our experiments.

We craft non-targeted and targeted attacks with and without defenses in our evaluation experiments. For targeted attacks, the state-of-the-art attacks of DRL use the strategies to find the optimal time throughout the episode. Instead of using such strategies, we generate targeted actions randomly to attack all time steps during an episode. This helps us to control the experiment across all attacks by having the strategy fixed. Attacking on all the time steps enables testing the attacks on all situations during the episode. We describe a generic evaluation experiment to apply attacks in DRL in Algorithm 1.

6.1.2 Setting for Defenses

Very few defenses have been proposed for adversarial attacks on DRL. The conventional adversarial training defense has been widely used in DNNs [24] and DRL [23, 33]. However, recently more robust and efficient defense methods were proposed specifically for DRL, such as Robust ADversarIAl Loss (RADIAL) [32], and State Adversarial (SA) [48], and Alternating Training of Learned Adversaries (ATLA) [47]. This research mainly considers RADIAL and ATLA which outperform SA and other major defenses of DRL. In particular, we used RADIAL-DQN, RADIAL-PPO and ATLA-PPO as defenses. However, we could not implement ATLA-DQN [47] due to the lack of resources given by authors.

6.2 Metrics

We use the following metrics for our analyses:

- Average Attack Execution Time Per Perturbation (AET) is the average time required to generate a perturbed state and is measured by:

$$\text{AET}_T = \frac{\mathcal{T}(N_A)}{N_S}, \tag{6.1}$$

where N_S is the total number of adversarial states computed, $\mathcal{T}(N_A)$ is the total time elapsed to generate all targeted or non-targeted adversarial states.

- Average Reward (AR) measures the average reward across all episodes. Given N_e the number of episodes and r_i the reward accumulated during an episode i, AR is measured by:

$$\text{AR} = \frac{\sum_{i=0}^{N_e} r_i}{N_e}. \tag{6.2}$$

- Attack Success Rate (ASR) measures the total number of attack successes over the total number of attack attempts. Considering either targeted or non-targeted attacks, ASR is measured by:

$$\text{ASR}_{NT} = \frac{N^{AS}}{N_{NT}}, \tag{6.3}$$

where N^{AS} is the total number of attack successes by targeted or non-targeted attacks and N_{NT} is the total number of attempts by targeted or non-targeted attacks. Under non-targeted attacks, a failure indicates that the attack is entirely unable to succeed or the DRL agent takes an action maximizing its reward. An attack attempt is defined as a one-time attack per time step. On the other hand, targeted attack success means generating a perturbation to lead the DRL agent to take targeted action. Thus, reward reduction is not considered an attack success in targeted attacks.

- ASR in continuous environments (ASR-C) is an ASR in a continuous environment, such as MuJoCo. We estimate a Mean Absolute Error (MAE) of two actions and if

the MAE of two actions is less than the threshold λ, we treat both actions as equal. In a non-targeted setting, an attempt is considered success if $MAE(a^{\text{true}}, a^{\text{adv}}) > \lambda$. However, in a targeted setting, success is defined when $MAE(a^{\text{targeted}}, a^{\text{adv}}) < \lambda$. We set $\lambda = 0.1$ in MuJoCo Walker in the results with ASR-C in the main results. We set $\lambda = 0.1$ in non-targeted setting because $\lambda \geq 0.1$ can create sufficient deviation from optimal actions. We also show the results of ASR-C when varying the value of λ.

6.3 Parameterization

6.3.1 Key Parameters

We use the following parameters for optimizing the performance of the variants of ACADIA (i.e., iACADIA, miACADIA, aiACADIA) and existing counterparts. We summarize the key parameters for different attacks in Table 6.1.

- Number of steps of perturbation (m): We need to make the steps as low as possible to reduce the attack execution time while achieving high ASR. We found the optimal at $m = 20$ for gradient-based attacks.
- Size of a perturbation (ϵ): It should not be very small or very large. As small ϵ leads to weak attack in terms of Attack Success Rate (ASR) while large ϵ leads to weak attack in terms of detectability. We found $\epsilon = 8/255$ to be optimal.
- Size of the step (α): It defines the size of a step taken in iterative perturbation methods with $\alpha < \epsilon$. Smaller α's can increase attack granularity at high step sizes but larger α's can decrease the number of steps to converge. We found $\alpha = 2/255$ to be optimal and used it in our work.

Table 6.1: Optimal Values of the Parameters Identified Under Each Perturbation Method for DQN and PPO

Perturbation Method	Steps (m)	ϵ	α
CW	1000	NA	NA
PGD	20	8/255	2/255
DI-FGSM	20	8/255	2/255
MI-FGSM	20	8/255	2/255
FGSM	1	8/255	NA
iACADIA	20	8/255	2/255
miACADIA	20	8/255	2/255
aiACADIA	20	8/255	2/255

- Threshold for ASR-C (λ): If the Mean Absolute Error (MAE) of two continuous actions is less than threshold λ, we treat both actions as equal. We set $\lambda = 0.1$. In a setting with non-targeted attacks, a higher λ implies a higher chance to generate a different action than the true action. Thus, we changed γ from 0.1 to 1 to observe the trends on ASR-C. However, in a setting with targeted attacks, lower λ means a higher chance to generate the targeted action.

- Decay values (μ): For miACADIA and MI-FGSM, we set $\mu = 0.99$. For aiACADIA, we set $\mu_1 = 0.99$ and $\mu_2 = 0.999$.

6.3.2 Loss Functions

We use a loss function between original/targeted action and the adversarial action based on the cross-entropy loss of PyTorch for gradient-based attacks. We employ L_∞ norm in all FGSM-based baselines, including our attacks while using L_2 norm for the CW. We run every experiment 100 times.

6.4 Baseline Schemes for Comparison

We use the following perturbation attacks for comparison against our ACADIA variants. As discussed earlier, we do not consider strategy attacks [27, 37] as comparing schemes because they focus on answering the second question, when-to-attack, rather than how-to-attack which is mainly considered in this work. We also did not include AutoAttack [11] as our baseline since it is based on PGD and has equivalent performance to PGD. PGD-based attacks including AutoAttack were not performing well on targeted attacks. So, it was mostly similar to PGD when used in DRL settings. Our proposed ACADIA is compared against the following baseline and state-of-the-art counterpart schemes:

- Fast Gradient Sign Method (FGSM) [16]: We choose this as our baseline scheme because it has been extensively used in both DNN settings and DRL settings, such as uniform attack [19].

- Carlini & Wagner Method (CW) [7]: This method is one of the well-known state-of-the-art methods in DRL settings guaranteeing 100% ASR. The state-of-the-art end goal-based DRL attacks [27, 37] use the CW method to generate targeted perturbations. Hence, outperforming the CW indicates clear advances of the state-of-the-art as an attack in DL.

- Projected Gradient Method (PGD) [29]: PGD is considered to be a robust, fast, and successful state-of-the-art attack in DRL settings. Due to its robustness, PGD is usually employed to test defenses as in [14, 32, 48]. Therefore, outperforming PGD shows the high performance of miACADIA.

- Momentum Iterative Fast Gradient Sign Method (MI-FGSM) [12]: We extend MI-FGSM from DNN settings to DRL settings and compare it with our method. MI-FGSM

is the state-of-the-art adversarial example in DNN settings. This effective method has not been enhanced to be applied in DRL settings in the literature.

- Diversity Iterative Fast Gradient Sign Method (DI-FGSM) [43]: We also extend DI-FGSM from DNNs to DRL and compare it with our method.

We report the average and standard deviation of ARs, AETs and ASRs. We conduct extensive experiments to evaluate the efficiency, effectiveness, and robustness of our attacks compared to other benchmark attacks. We consider both targeted and non-targeted attacks, and DRL with and without defense (i.e., RADIAL and ATLA). For non-targeted (NT) attacks, we avoid the desired action identified by the DRL model. For targeted (T) attacks, we take a particular perturbation to mislead a DRL agent to the attacker's desired action. The targeted actions were chosen randomly to evaluate the ability of perturbation methods to generate a targeted action. We do not include targeted attacks on MuJoCo Walker environments since most of the attacks, including CW, cannot precisely generate random targeted actions. Therefore, we leave studying efficient and precise targeted attacks in continuous DRL environments for our future work.

The number of steps (m) is fixed at 20, which is optimal for FGSM-based perturbation attacks based on our experiment. However, we use $m = 1,000$ for the CW method at which it performed best. For a fair comparison, we also add results for CW with $m = 20$. We considered naïve FGSM, using $m = 1$ by definition, as a baseline. AET cannot be fixed as it fluctuates with high variance. However, in order to give a fair comparison, we considered CW with $m = 100$ and $m = 120$ as they were taking AET comparable to ACADIA. By fixing $m = 20$, baseline perturbation methods are also taking time (AET) comparable to ACADIA.

We fix $\epsilon = 8/255$ and $\alpha = 2/255$ for our experiments under DQN and PPO, which are

Algorithm 1 Perform Attacks in DRL

Input:
$A \leftarrow$ an actions set, $s_0 \leftarrow$ an initial non-adversarial state
Parameters:
targeted $\leftarrow$ return 1 if attack is targeted; 0 otherwise
defense $\leftarrow$ return 1 if defense is applied; 0 otherwise
procedure PerturbationMethod((A, s_0)) ▷ a perturbation method used
    for each episode do
        for each step t during an episode do
            if targeted is true then
                $a_t^{\text{adv}*} = \text{RandomStrategy}(A)$
                $s_t^{\text{adv}} = \text{PerturbationMethod}(s_t, a_t^{adv*})$
            else
                $s_t^{\text{adv}} = \text{PerturbationMethod}(s_t)$
            end if
            if defense is true then
                $a_t^{\text{adv}} = \text{DRL}_{\text{defense}}(s_t^{\text{adv}})$
            else
                $a_t^{\text{adv}} = \text{DRL}(s_t^{\text{adv}})$
            end if
            $r_t^{\text{adv}}, s_{t+1}, \text{done} = \text{Perform}(a_t^{\text{adv}})$
            if done is true then
                break
            end if
        end for
    end for
end procedure

identified as optimal in terms of detectability, AR, and ASR after rigorous sensitivity analysis on all considered perturbation methods.

Chapter 7

Results & Analyses

In this section, we compare the performance of our proposed ACADIA with the baseline and existing counterparts in terms of the metrics in Section 6.2 under multiple DRL applications. We also vary the types of attacks by being targeted or non-targeted. Further, we examine the effectiveness and efficiency of the attacks when a defense exists in the DRL.

7.1 Comparative Performance Analysis based on Average Attack Execution Time per Perturbation (AET)

Tables 7.1 and 7.2 show the comparison of AET for Atari Pong, Atari RoadRunner and Atari BankHeist played by DQN and MuJoCo Walker played by PPO under targeted and non-targeted attacks. Overall, FGSM variants perform comparably. Non-targeted iACADIA, the best performing, is 12 ms faster than the non-targeted PGD attack for Pong played by DQN. For targeted attacks, PGD shows comparable performance to ACADIA. iACADIA outperforms again in AET for targeted attacks. Interestingly, CW with 20 steps is faster than most FGSM variants with 20 steps, while CW performs poorly with 20 steps. CW with 100-120 steps takes comparable AET to ACADIA. miACADIA (20 steps) is six to nine times faster than the state-of-the-art CW (1,000 steps) with better robustness. Similar results are observed for other Atari games and MuJoCo environments. As our attacks take a little more

Table 7.1: Comparative Performance Analysis of ACADIA and Other Existing Schemes in Terms of Attack Execution Time (AET) for Pong and BankHeist using DQN. Table shows mean and standard deviation of AET.

Perturbation Method (steps)	Pong		BankHeist	
	Non-Targeted	Targeted	Non-Targeted	Targeted
CW (1000)	716 ± 32	963 ± 20	693 ± 55	715 ± 134
CW (120)	118 ± 9	119 ± 1	124 ± 9	135 ± 16
CW (100)	97 ± 5	99 ± 6	105 ± 6	109 ± 12
MIFGSM (20)	128 ± 8	138 ± 8	92 ± 6	91 ± 6
DIFGSM (20)	103 ± 6	135 ± 14	101 ± 5	100 ± 6
PGD (20)	94 ± 6	117 ± 13	87 ± 5	94 ± 7
CW (20)	21 ± 2	26 ± 2	21 ± 2	21 ± 2
FGSM (1)	6 ± 2	6.3 ± 0.7	5 ± 0.5	5 ± 0.6
miACADIA (20)	126 ± 7	125 ± 7	122 ± 5	122 ± 5
iACADIA (20)	82 ± 6	98 ± 6	87 ± 7	92 ± 11
aiACADIA (20)	138 ± 6	141 ± 7	122 ± 5	122 ± 5

time than other FGSM counterparts, so the question is whether it is worth spending more time. Minute increase of 20 milliseconds and 2-10 milliseconds in AET increased ASR of miACADIA and aiACADIA significantly up to 22% better than PGD and up to 24% better than MI-FGSM, which is quite reasonable. Overall, our attacks are effective under realistic situations as the time to craft a perturbation in our attacks is only a few milliseconds.

We have only shown AET under Vanilla DQN and Vanilla PPO. We did not show AET under defenses because it takes same time for the perturbation method to compute the perturbation under defense and no defense.

We observe that CW can be 7 to 22 times slower than our attacks. Although in the MuJoCo experiments, CW (1000 steps), best performing CW, takes less time than in Atari environments, it still cannot work under realistic situations where we have less time to craft the perturbation. Overall, we show that our attacks are effective under realistic situations as

Table 7.2: Comparative Performance Analysis of ACADIA and Other Existing Schemes in Terms of of Attack Execution Time (AET) for RoadRunner using DQN and MuJoCo Walker using PPO. Table shows mean and standard deviation of AET.

Perturbation Method (steps)	RoadRunner		MuJoCo Walker	
	Non-Targeted	Targeted	Non-Targeted	Targeted
CW (1000)	777 ± 90	790 ± 94	174 ± 131	317 ± 134
CW (120)	118 ± 4	123 ± 31	151 ± 156	102 ± 65
CW (100)	98 ± 7	107 ± 25	140 ± 119	87 ± 67
MIFGSM (20)	77 ± 5	81 ± 5	10 ± 2	15 ± 2
DIFGSM (20)	91 ± 8	93 ± 5	11 ± 2	18 ± 2
PGD (20)	79 ± 5	82 ± 5	10 ± 2	17 ± 3
CW (20)	22 ± 17	21 ± 9	16 ± 3	9 ± 2
FGSM (1)	5 ± 0.4	5 ± 0.5	5 ± 1	5 ± 1
miACADIA (20)	78 ± 5	82 ± 5	11 ± 2	14 ± 2
iACADIA (20)	81 ± 5	82 ± 7	10 ± 2	17 ± 3
aiACADIA (20)	78 ± 5	82 ± 5	11 ± 2	14 ± 3

the time to craft a perturbation in our attacks is only a few milliseconds.

7.2 Performance Analysis of Average Reward (AR), Attack Success Rate (ASR), and ASR in Continuous Environments (ASR-C)

Table 7.3 and Table 7.4 show a comprehensive comparison between ACADIA variants and the baselines on ASR when tested on DQN, RADIAL-DQN, PPO and RADIAL-PPO playing Atari and MuJoCo environments. High ASR means a better attack. Table 7.5 and Table 7.6 show the comparison in terms of AR. Low AR means a better attack. ACADIA variants outperform all baselines. When RADIAL is used as a defense in DRL, all alternatives are impacted; however, the ACADIA variants are impacted the least. On the other hand, CW completely fails in both non-targeted and targeted attacks, performing worse than even naïve

Table 7.3: Comparison of Attack Success Rate (ASR) for DQN under Targeted (T) and Non-Targeted (NT) attacks with and without RADIAL defense on Atari Games. (Note: V-DQN refers to Vanilla DQN, and R-DQN is RADIAL-DQN).

Perturbation Method (steps)	Pong				BankHeist			
	V-DQN		R-DQN		V-DQN		R-DQN	
	NT	T	NT	T	NT	T	NT	T
CW (1000)	100%	100%	3%	16%	100%	100%	3%	5%
CW (120)	100%	16%	3%	17%	94%	4%	0%	6%
CW (100)	100%	17%	3%	17%	91%	5%	0%	4%
CW (20)	100%	16%	3%	16%	100%	6%	3%	5%
PGD (20)	100%	100%	99%	72%	100%	100%	99%	78%
DIFGSM (20)	100%	99%	73%	44%	100%	99%	88%	68%
MIFGSM (20)	100%	99%	75%	65%	100%	100%	100%	88%
FGSM (1)	85%	35%	28%	16%	100%	66%	47%	47%
miACADIA (20)	100%	100%	99%	78%	100%	100%	100%	90%
iACADIA (20)	100%	100%	99%	73%	100%	100%	99%	79%
aiACADIA (20)	100%	100%	99%	75%	100%	100%	100%	92%

FGSM under RADIAL defense. PGD performs comparably to the worst ACADIA variant in ASR, but not to the best variant of ACADIA.

We showed 14 types of experiments in Table 7.5 and Table 7.6. CW(1000) shows higher AR than aiACADIA in only 4 of 14 types. These 4 cases are without the defenses, clearly showing CW(1000) is not robust and is therefore not a better perturbation method than our attacks. miACADIA and aiACADIA show lower AR than MI-FGSM in only 3 of the 14 types. In these few cases, aiACADIA and miACADIA are comparable to MI-FGSM and CW(1000). CW(100), CW(120) and CW(20) performed worse than ACADIA in all cases in terms of AR as well as ASR. Hence, overall our proposed attacks outperform in AR.

Table 7.7 shows the comparison of Perturbation Methods in ASR-C for MuJoCo Walker using multiple defenses of ATLA-PPO and RADIAL-PPO under non-targeted attacks. This metric actually shows whether an attack can generate a different action or not based on the

Table 7.4: Comparison of Attack Success Rate (ASR) for DQN and PPO under Targeted (T) and Non-Targeted (NT) attacks with and without RADIAL defense on Atari Games and MuJoCo environments (Note: V-DQN refers to Vanilla DQN, V-PPO is Vanilla PPO, and R-DQN is RADIAL-DQN).

Perturbation Method (steps)	RoadRunner				MuJoCo Walker	
	V-DQN		R-DQN		V-PPO	R-PPO
	NT	T	NT	T	NT	NT
CW (1000)	95%	97%	1%	0%	100%	98%
CW (120)	83%	7%	3%	4%	98%	100%
CW (100)	94%	7%	2%	5%	96%	100%
CW (20)	98%	6%	2%	0%	100%	98%
PGD (20)	99%	76%	99%	89%	100%	100%
DIFGSM (20)	99%	79%	89%	79%	100%	100%
MIFGSM (20)	99%	96%	100%	91%	100%	100%
FGSM (1)	79%	43%	91%	85%	100%	100%
miACADIA (20)	99%	98%	100%	95%	100%	100%
iACADIA (20)	99%	76%	100%	91%	100%	100%
aiACADIA (20)	99%	97%	99%	96%	100%	100%

λ value. Here we set $\lambda = 0.1$. Clearly, ACADIA are either better or comparable to baselines. CW is affected under RADIAL-PPO but with a minor impact. Hence, interestingly, CW works well under defenses in the MuJoCo PPO experiments while performing poorly on Atari games under RADIAL-DQN.

Table 7.8 shows the comparison of Perturbation Methods in AR for MuJoCo Walker using multiple defenses of ATLA-PPO and RADIAL-PPO under non-targeted attacks. ACADIA again show robustness against RADIAL-PPO since they achieve the minimum reward among all the baselines. ACADIA outperform in achieving minimum rewards under ATLA-PPO as compared to the gradient-based counterparts. However, CW can achieve a lower reward than our attacks under ATLA-PPO. The performance of our attacks can be evaluated through the reward reduction obtained. In MuJoCo Walker, the maximum reward achieved by ATLA-PPO under no attack is around 4,000 while our best attack gives 321. Therefore, the reward

reduction is around 3,679 by our attacks. Considering our attacks achieve 100% ASR, we can conclude that our attacks do not take the worst actions but can distract the DRL agent to take non-optimal actions.

7.3 Comparison of iACADIA, miACADIA and aiACADIA

In terms of AET, iACADIA is better than miACADIA and miACADIA is better than aiACADIA because miACADIA incorporates momentum and aiACADIA incorporates both momentum and RMSProp. But even our basic version, iACADIA, performs better than most of the baselines. On ASR and AR metrics, we observed that aiACADIA and miACADIA perform similarly in most cases. aiACADIA outperforms in a few instances, such as when RoadRunner is played by Vanilla DQN. Therefore, incorporating RMSProp helps in these cases. However, using momentum in the attack significantly helps achieve high ASR and low AR. As miACADIA takes less time than aiACADIA and shows comparable performance on AR and ASR, miACADIA can be considered the best variant.

7.4 Sensitivity Analyses

Based on sensitivity analyses by varying all design parameters discussed earlier, we share the two most interesting results as follows. We vary the number of steps (m) and the threshold for ASR-C (λ) to investigate their effects on attack performance. Parameter m is critical for introducing the direct impact on AET; thus, we need to choose the lowest m possible. Parameter λ is also important as our proposed novel metric ASR-C can be significantly influenced by varying λ.

7.4.1 Threshold for ASR-C (λ)

We vary λ from 0.1 to 1 to test the impact on ASR-C (i.e., $\lambda = [0.1, 0.25, 0.5, 0.75, 0.85, 0.9, 0.95, 1]$). Under a non-targeted attack on Vanilla PPO, miACADIA reduces ASR-C from 100% to 80% while CW reduces from 100% to 96%. However, both attacks maintain 100% until $\lambda = 0.75$, which means both attacks are quite successful in taking non-optimal actions. However, aiACADIA performance drops after $\lambda = 0.85$. This can be clearly observed in Figure 7.1 which shows our two best variants, the state-of-the-art FGSM-based attack (PGD) and CW. Under a non-targeted attack on RADIAL-PPO, we observe no change in ASR-C.

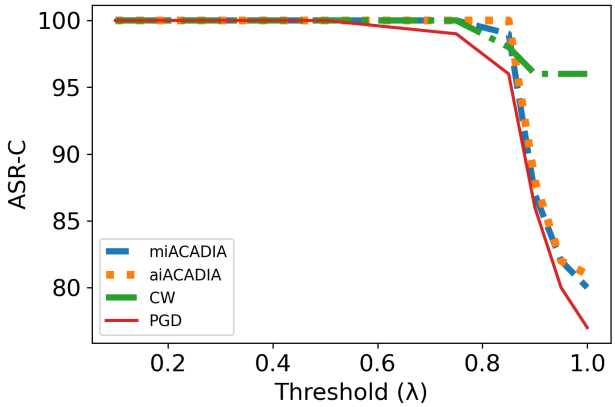
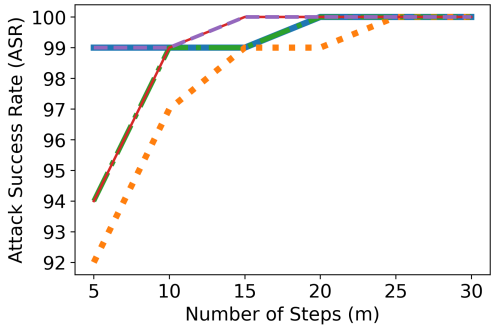


Figure 7.1: Attack Success Rate Continuous (ASR-C) parameterized against λ for four attacks: miACADIA, aiACADIA, CW and PGD.

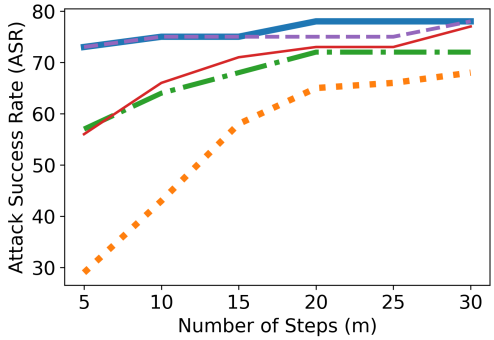
7.4.2 Number of Steps (m)

Here we compare the most promising five perturbation methods (i.e., FGSM-based attacks) observed in our experiments, including aiACADIA, miACADIA, iACADIA, MI-FGSM, and PGD to perform targeted attacks. We vary $m = [5, 10, 15, 20, 25, 30]$ and analyze their impact on ASR and AET. For non-targeted attacks under no defense on DQN playing Pong, we did not observe a considerable change in ASR and AR. For targeted attacks under no defense, Figure 7.2a shows the increase in ASR as we increase the number of steps for

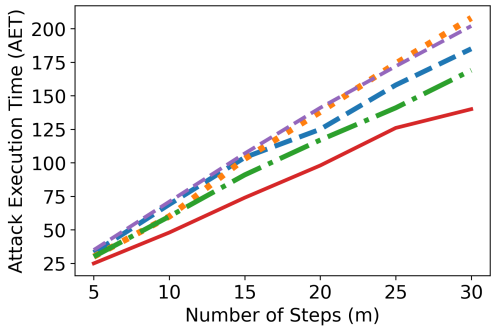
each perturbation method when Vanilla DQN is playing Pong. We can see that almost all perturbation attacks give maximum ASR at $m = 20$. However, our attacks were able to reach the maximum ASR of 100% at $m = 15$, which shows their effectiveness on low AET as fewer steps take less time. This behavior can be seen in Figure 7.2c, where increasing the number of steps increases the AET. This figure also shows that all FGSM variants have comparable performance but iACADIA is the fastest. Figure 7.2b shows the ASR of targeted attack for RADIAL DQN playing Pong. Again, ASR increases as the number of steps increases. aiACADIA and miACADIA clearly outperform under RADIAL defense. We only include analyses for Atari Pong as we observed similar trends in other environments.



(a) ASR of targeted attack for Vanilla DQN playing Pong



(b) ASR of targeted attack for RADIAL DQN playing Pong



(c) AET of targeted attack for Vanilla DQN playing Pong

Figure 7.2: Sensitivity analysis of aiACADIA, miACADIA, iACADIA, MI-FGSM, and PGD under varying the number of steps (m) in Attack Success Rate (ASR) and Attack Execution Time (AET).

Table 7.5: Comparison of Average Reward (AR) for DQN under Targeted (T) and Non-Targeted (NT) attacks with and without RADIAL defense on Atari Pong and Atari BankHeist. Low AR means better attack.

Perturbation Method (steps)	Pong			
	Vanilla DQN		RADIAL-DQN	
	Non-Targeted	Targeted	Non-Targeted	Targeted
CW (1000)	-21 ± 0	-21 ± 0	$+20.85 \pm 0.3$	$+20.5 \pm 0.5$
CW (120)	-21 ± 0	$+20.5 \pm 0.5$	$+21 \pm 0$	$+20.5 \pm 0.5$
CW (100)	-19 ± 0	$+21 \pm 0$	$+20.5 \pm 0.5$	$+20.5 \pm 0.5$
CW (20)	-21 ± 0	$+20.7 \pm 0.4$	$+20.7 \pm 0.4$	$+20.8 \pm 0.4$
PGD (20)	-21 ± 0	-20.6 ± 0.4	-20.9 ± 0.2	-20.4 ± 0.8
DIFGSM (20)	-21 ± 0	-20.3 ± 0.6	-19.8 ± 1.3	-16.7 ± 2.6
MIFGSM (20)	-21 ± 0	-20.8 ± 0.4	-20.5 ± 0.7	-20.4 ± 0.7
FGSM (1)	-21 ± 0	-20.7 ± 0.9	$+20.7 \pm 0.4$	$+16.8 \pm 7.8$
miACADIA (20)	-21 ± 0	-20.7 ± 0.4	-20.9 ± 0.2	-20.3 ± 0.9
iACADIA (20)	-21 ± 0	-20.3 ± 0.6	-20.9 ± 0.3	-20.2 ± 0.9
aiACADIA (20)	-21 ± 0	-20.5 ± 0.5	-21 ± 0	-21 ± 0
Perturbation Method (steps)	BankHeist			
	Vanilla DQN		RADIAL-DQN	
	Non-Targeted	Targeted	Non-Targeted	Targeted
CW (1000)	0 ± 0	0 ± 0	252 ± 34	302 ± 39
CW (120)	630 ± 0	780 ± 0	390 ± 390	390 ± 170
CW (100)	450 ± 0	770 ± 10	390 ± 170	390 ± 170
CW (20)	550 ± 50	710 ± 10	309 ± 36	321 ± 46
PGD (20)	6 ± 9	10 ± 0	2.3 ± 5.5	4 ± 6
DIFGSM (20)	3 ± 4	20 ± 8	1 ± 4	4 ± 6
MIFGSM (20)	0 ± 0	20 ± 8	0.6 ± 2.4	3.6 ± 5.4
FGSM (1)	0 ± 0	23 ± 20	0 ± 0	2.3 ± 4.2
miACADIA (20)	0 ± 0	16 ± 17	1.6 ± 4.5	3 ± 5.2
iACADIA (20)	6 ± 4	6 ± 4	1 ± 3	3 ± 4.5
aiACADIA (20)	0 ± 0	20 ± 8	1 ± 2	3 ± 0

Table 7.6: Comparison of Average Reward (AR) for DQN and PPO under Targeted (T) and Non-Targeted (NT) attacks with and without RADIAL defense on Atari RoadRunner and MuJoCo Walker environments. Low AR means better attack.

Perturbation Method (steps)	RoadRunner			
	Vanilla DQN		RADIAL-DQN	
	Non-Targeted	Targeted	Non-Targeted	Targeted
CW (1000)	0 ± 0	0 ± 0	14000 ± 1000	18400 ± 15400
CW (120)	11700 ± 0	0 ± 0	5750 ± 4750	9050 ± 5550
CW (100)	100 ± 0	0 ± 0	22200 ± 15000	9050 ± 5550
CW (20)	8600 ± 0	0 ± 0	14000 ± 1000	18400 ± 15400
PGD (20)	200 ± 141	0 ± 0	17 ± 45	23 ± 76
DIFGSM (20)	1100 ± 816	0 ± 0	87 ± 106	280 ± 399
MIFGSM (20)	500 ± 0	0 ± 0	3 ± 18	3 ± 18
FGSM (1)	100 ± 0	0 ± 0	247 ± 394	220 ± 338
miACADIA (20)	33± 47	0± 0	0± 0	73± 254
iACADIA (20)	100± 141	0± 0	13± 34	33± 79
aiACADIA (20)	23± 22	0 ± 0	0± 0	0± 0

Perturbation Method (steps)	MuJoCo Walker	
	Vanilla PPO	RADIAL-PPO
	Non-Targeted	Non-Targeted
CW (1000)	-7 ± 3	-14 ± 1
CW (120)	-7 ± 2	-11 ± 0.3
CW (100)	-6 ± 2	-12 ± 0.3
CW (20)	-5 ± 8	-18 ± 3
PGD (20)	71 ± 3	-49 ± 1
DIFGSM (20)	68 ± 8	-49 ± 1
MIFGSM (20)	76 ± 4	-49 ± 1
FGSM (1)	52 ± 3	-49 ± 1
miACADIA (20)	52 ± 3	-50 ± 2
iACADIA (20)	67 ± 3	-50 ± 1
aiACADIA (20)	52 ± 3	-51 ± 3

Table 7.7: Comparison of Perturbation Methods in ASR-C for MuJoCo Walker using multiple defenses of ATLA-PPO and RADIAL-PPO (Note: V-PPO for Vanilla PPO; A-PPO for ATLA-PPO; and R-PPO for RADIAL-PPO).

Perturbation Method (steps)	V-PPO	A-PPO	R-PPO
CW (1000)	100%	100%	98%
CW (20)	100%	82%	98%
CW (120)	98%	81%	97%
CW (100)	96%	80%	97%
PGD (20)	100%	98%	100%
MI-FGSM (20)	100%	98%	100%
FGSM (1)	100%	96%	100%
miACADIA (20)	100%	100%	100%
iACADIA (20)	100%	98%	100%
aiACADIA (20)	100%	100%	100%

Table 7.8: Comparison of Perturbation Methods in AR for MuJoCo Walker using multiple defenses of ATLA-PPO and RADIAL-PPO (Note: V-PPO for Vanilla PPO; A-PPO for ATLA-PPO; and R-PPO for RADIAL-PPO).

Perturbation Method (steps)	V-PPO	A-PPO	R-PPO
CW (1000)	-7 ± 3	-14 ± 1	-14 ± 1
CW (20)	-5 ± 8	-7 ± 4	-18 ± 3
CW (120)	-7 ± 2	-10 ± 2	-11 ± 0.3
CW (100)	-6 ± 2	-10 ± 1	-12 ± 0.3
PGD (20)	71 ± 3	330 ± 42	-49 ± 1
MI-FGSM (20)	76 ± 4	331 ± 48	-49 ± 1
FGSM (1)	52 ± 3	351 ± 27	-49 ± 1
miACADIA (20)	52 ± 3	322 ± 80	-50 ± 2
iACADIA (20)	67 ± 3	325 ± 43	-50 ± 1
aiACADIA (20)	52 ± 3	321 ± 43	-51 ± 3

Chapter 8

Conclusions & Future Work

8.1 Summary of Key Findings

We proposed an efficient and robust perturbation attack applicable in deep reinforcement learning (DRL) environments, including aiACADIA, miACADIA and iACADIA. This is the first of its kind research addressing the need to develop efficient and robust perturbations specifically for DRL. Following are the key findings obtained from this study:

- ACADIA is deployable to time-sensitive real-life applications as it can generate the state perturbations in less than 140 ms, which is a realistic time. ACADIA is nine times faster than CW and comparable to their FGSM counterparts.

- ACADIA outperforms baselines in Attack Success Rate (ASR) and Average Reward (AR) overall baselines, especially under the defense. ACADIA is either better or comparable to baselines under no defense.

- PGD could not perform well on targeted attack settings and CW could not perform well under defenses.

- miACADIA can be considered as our best variant in terms of efficiency and effectiveness.

8.2 Future Work

For future work, we will focus on:

- Conducting extensive experiments to evaluate and improve targeted ACADIA and baselines in continuous control environments of DRL.
- Leveraging other components of the DRL process such as reward and policy functions to enhance the performance of ACADIA.
- Developing novel when-to-attack strategies and concatenating existing when-to-attack strategies with ACADIA.

8.3 Publications

The following paper is published based on the current dissertation research:

- Haider Ali, Mohannad Al Ameedi, Ananthram Swami, Rui Ning, Jiang Li, Hongyi Wu, and Jin-Hee Cho. 2022. ACADIA: Efficient and Robust Adversarial Attacks Against Deep Reinforcement Learning. 2022 IEEE Conference on Communications and Network Security (CNS) (2022).

The following paper is submitted and currently under review:

- Mohannad Al Ameedi, Haider Ali, Ananthram Swami, Rui Ning, Jiang Li, Chunsheng Xin, Hongyi Wu, and Jin-Hee Cho. ERBANA: Efficient, Robust Backdoor Attack Detection Using Principal Component Analysis. 2023 IEEE International Conference on Communications (ICC).

The following papers are currently in preparation:

- Haider Ali, Ahmad Faraz Khan, Mohannad Al Ameedi, Ananthram Swami, Rui Ning, Jiang Li, Hongyi Wu, and Jin-Hee Cho. 2023. PETER: Privacy-prEserving verTical fEderated leaRning Against Feature Inference Attacks. 2023 ACM Asia Conference on Computer and Communications Security (AsiaCCS)

- Haider Ali, Dian Chen, Mathew Harrington, Nathaniel Salazaar, Mohannad Al Ameedi and Jin-Hee Cho. 2023. A Survey on Attacks and Their Countermeasures in Deep Learning: Applications in Deep Neural Networks, Federated, Transfer, and Deep Reinforcement Learning (ACM Computing Surveys 2023)

Bibliography

[1] Haider Ali, Mohannad Al Ameedi, Ananthram Swami, Rui Ning, Jiang Li, Hongyi Wu, and Jin-Hee Cho. 2022. ACADIA: Efficient and Robust Adversarial Attacks Against Deep Reinforcement Learning. In 2022 IEEE Conference on Communications and Network Security (CNS). IEEE, 1–9.

[2] Moustafa Alzantot, Bharathan Balaji, and Mani Srivastava. 2018. Did You Hear That? Adversarial Examples Against Automatic Speech Recognition. arXiv preprint arXiv:1801.00554 (2018).

[3] Smruti Amarjyoti. 2017. Deep Reinforcement Learning for Robotic Manipulation-The State Of The Art. arXiv preprint arXiv:1701.08878 (2017).

[4] Tao Bai, Jinqi Luo, and Jun Zhao. 2020. Recent Advances in Understanding Adversarial Robustness of Deep Neural Networks. arXiv:2011.01539 [cs] (Nov. 2020).

[5] Ahmad Hoirul Basori and Sharaf Jameel Malebary. 2020. Deep Reinforcement Learning For Adaptive Cyber Defense And Attacker's Pattern identification. In Advances in Cyber Security Analytics and Decision Systems. Springer, 15–25.

[6] Vahid Behzadan and Arslan Munir. 2017. Vulnerability of Deep Reinforcement Learning to Policy Induction Attacks. In International Conference on Machine Learning and Data Mining in Pattern Recognition. Springer, 262–275.

[7] Nicholas Carlini and David Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In 2017 IEEE Symposium on Security and Privacy (SP). IEEE, 39–57.

[8] Kangjie Chen, Shangwei Guo, Tianwei Zhang, Xiaofei Xie, and Yang Liu. 2021. Stealing Deep Reinforcement Learning Models for Fun and Profit. In Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security. 307–319.

[9] Tong Chen, Wenjia Niu, Yingxiao Xiang, Xiaoxuan Bai, Jiqiang Liu, Zhen Han, and Gang Li. 2018. Gradient Band-based Adversarial Training for Generalized Attack Immunity of A3C Path Finding. arXiv preprint arXiv:1807.06752 (2018).

[10] George Clark, Michael Doran, and William Glisson. 2018. 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). IEEE, 516–521.

[11] Francesco Croce and Matthias Hein. 2020. Reliable Evaluation of Adversarial Robustness with an Ensemble of Diverse Parameter-Free Attacks. In International Conference on Machine Learning. PMLR, 2206–2216.

[12] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. 2018. Boosting Adversarial Attacks with Momentum. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 9185–9193.

[13] Martin Figura, Krishna Chaitanya Kosaraju, and Vijay Gupta. 2021. Adversarial Attacks in Consensus-based Multi-Agent Reinforcement Learning. In 2021 American Control Conference (ACC). IEEE, 3050–3055.

[14] Marc Fischer, Matthew Mirman, Steven Stalder, and Martin Vechev. 2019. Online Robustness Training for Deep Reinforcement Learning. arXiv preprint arXiv:1911.00887 (2019).

[15] Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. 2019. Adversarial Policies: Attacking Deep Reinforcement Learning. arXiv preprint arXiv:1905.10615 (2019).

[16] Ian Goodfellow and et al. 2014. Explaining and Harnessing Adversarial Examples. ICLR (2014).

[17] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. Badnets: Evaluating backdooring attacks on deep neural networks. IEEE Access 7 (2019), 47230–47244.

[18] Wei Guo, Benedetta Tondi, and Mauro Barni. 2022. An overview of backdoor attacks against deep neural networks and possible defences. IEEE Open Journal of Signal Processing (2022).

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

[20] Léonard Hussenot, Matthieu Geist, and Olivier Pietquin. 2019. CopyCAT: Taking Control of Neural Policies with Constant Attacks. arXiv preprint arXiv:1905.12282 (2019).

[21] Panagiota Kiourti, Kacper Wardega, Susmit Jha, and Wenchao Li. 2019. Trojdrl: Trojan Attacks on Deep Reinforcement Learning Agents. arXiv preprint arXiv:1903.06638 (2019).

[22] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. 2021. Deep Reinforcement Learning for Autonomous Driving: A Survey. IEEE Transactions on Intelligent Transportation Systems (2021).

[23] Jernej Kos and Dawn Song. 2017. Delving into Adversarial Attacks on Deep Policies. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings. OpenReview.net. https://openreview.net/forum?id=BJcib5mFe

[24] Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. 2016. Adversarial Examples in the Physical World. arxiv.org/pdf/1607.02533 (2016).

[25] Shaofeng Li, Benjamin Zi Hao Zhao, Jiahao Yu, Minhui Xue, Dali Kaafar, and Haojin Zhu. 2019. Invisible backdoor attacks against deep neural networks. arXiv preprint arXiv:1909.02742 (2019).

[26] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2022. Backdoor learning: A survey. IEEE Transactions on Neural Networks and Learning Systems (2022).

[27] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. 2017. Tactics of Adversarial Attack on Deep Reinforcement Learning Agents. In Proceedings of the 26th International Joint Conference on Artificial Intelligence (Melbourne, Australia) (IJCAI'17). AAAI Press, 3756–3762.

[28] Guanlin Liu and Lifeng Lai. 2021. Provably Efficient Black-Box Action Poisoning Attacks Against Reinforcement Learning. Advances in Neural Information Processing Systems 34 (2021).

[29] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards Deep Learning Models Resistant to Adversarial Attacks. arXiv preprint arXiv:1706.06083 (2017).

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou,

Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013).

[31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-Level Control Through Deep Reinforcement Learning. Nature 518, 7540 (2015), 529–533.

[32] Tuomas Oikarinen, Wang Zhang, Alexandre Megretski, Luca Daniel, and Tsui-Wei Weng. 2021. Robust Deep Reinforcement Learning through Adversarial Loss. In Advances in Neural Information Processing Systems, A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (Eds.). https://openreview.net/forum?id=eaAM_bdW0Q

[33] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. 2017. Robust Deep Reinforcement Learning with Adversarial Attacks. arXiv preprint arXiv:1712.03632 (2017).

[34] You Qiaoben, Xinning Zhou, Chengyang Ying, and Jun Zhu. 2021. Strategically-Timed State-Observation Attacks on Deep Reinforcement Learning Agents. In ICML 2021 Workshop on Adversarial Machine Learning.

[35] Xinghua Qu, Yew-Soon Ong, and Abhishek Gupta. 2021. Frame-Correlation Transfers Trigger Economical Attacks on Deep Reinforcement Learning Policies. IEEE Transactions on Cybernetics (2021).

[36] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347 (2017).

[37] Jianwen Sun, Tianwei Zhang, Xiaofei Xie, Lei Ma, Yan Zheng, Kangjie Chen, and Yang

Liu. 2020. Stealthy and Efficient Adversarial Attacks Against Deep Reinforcement Learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34. 5883–5891.

[38] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing Properties of Neural Networks. arXiv preprint arXiv:1312.6199 (2013).

[39] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2017. Ensemble Adversarial Training: Attacks and Defenses. arXiv preprint arXiv:1705.07204 (2017).

[40] Edgar Tretschk, Seong Joon Oh, and Mario Fritz. 2018. Sequential Attacks on Agents for Long-term Adversarial Goals. arXiv preprint arXiv:1805.12487 (2018).

[41] Chaowei Xiao, Xinlei Pan, Warren He, Jian Peng, Mingjie Sun, Jinfeng Yi, Mingyan Liu, Bo Li, and Dawn Song. 2019. Characterizing Attacks on Deep Reinforcement Learning. arXiv preprint arXiv:1907.09470 (2019).

[42] Yatie Xiao, Chi-Man Pun, and Bo Liu. 2020. Adversarial Example Generation with Adaptive Gradient Search for Single and Ensemble Deep Neural Network. Information Sciences 528 (2020), 147–167.

[43] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille. 2019. Improving Transferability of Adversarial Examples with Input Diversity. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2730–2739.

[44] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. 2019. Latent backdoor

attacks on deep neural networks. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. 2041–2055.

[45] Heng Yin, Hengwei Zhang, Jindong Wang, and Ruiyu Dou. 2020. Improving the Transferability of Adversarial Examples with the Adam Optimizer. arXiv preprint arXiv:2012.00567 (2020).

[46] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. 2019. Adversarial Examples: Attacks and Defenses for Deep Learning. IEEE Transactions on Neural Networks and Learning Systems 30, 9 (2019), 2805–2824. https://doi.org/10.1109/TNNLS.2018.2886017

[47] Huan Zhang, Hongge Chen, Duane Boning, and Cho-Jui Hsieh. 2021. Robust Reinforcement Learning on State Observations with Learned Optimal Adversary. arXiv preprint arXiv:2101.08452 (2021).

[48] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. 2020. Robust Deep Reinforcement Learning against Adversarial Perturbations on State Observations. arXiv preprint arXiv:2003.08938 (2020).

Appendices

Appendix A

Background on Atari and MuJoCo

In the Atari environment, one can play classic Atari games, such as Pong, BankHeist, and RoadRunner. MuJoCo is a physics engine designed for fast and accurate robot simulation, such as Walker environment making a 2D robot walk. Each environment has an action space and a state space. Atari games have discrete action spaces. For example, Pong needs to choose an action from 6 possible discrete actions (up, down, no action, up, down, no action) while BankHeist and RoadRunner need to choose from 18 possible actions. The dimensions of state-space in Atari games are $(210, 160, 3)$, where 210 is a number of rows, 160 is a number of columns and 3 means RGB. But, we converted it to 84 rows and 84 columns $(84, 84)$ for our implementation similar to the implementation of [31]. MuJoCo environments have continuous action spaces. For instance, a Walker needs to choose 6 continuous actions from a continuous space of $[-3, 3]$ at a given point. The dimension of Walker's state space is $(17, 1)$, which shows positional values of different body parts of Walker including velocities. These environments are provided by Open AI Gym to support the standardization of environments. When a DRL agent plays in one of these environments, a reward is returned in each step. There are maximum total rewards per episode achieved by state-of-the-art DRL agents for these environments, which are 21, >1200, >4200 and >400 for Pong, BankHeist, RoadRunner and Walker, respectively.

Appendix B

Explanations of Acronyms/Abbreviations

Table B.1: Acronyms/Abbreviations used in the dissertation and their explanations

Acronym	Explanations
ACADIA	Novel framework of gradient-based AttaCks Against Deep reInforcement leArning
iACADIA	Iterative ACADIA: Multiple steps in ACADIA
miACADIA	Momentum iACADIA: Optimization using Momentum in iACADIA
aiACADIA	ADAM iACADIA: Optimization using ADAM in iACADIA
DRL	Deep Reinforcement Learning: DNN is used as policy in Reinforcement Learning
ADAM	Adaptive Moment Optimization is an optimization technique
DNN	Deep Neural Networks
PPO	Proximal Policy Optimization is a DRL algorithm
DQN	Deep Q-Learning Networks is a DRL algorithm
CW	Carlini & Wagner (baseline method)
FGSM	Fast Gradient Sign Method (baseline method)
AET	Attack Execution Time Per Perturbation (metric)
ASR	Attack Success Rate (metric)
AR	Average Reward (metric)
ASR-C	Attack Success Rate in Continuous Environments (metric)
MAE	Mean Absolute Error
PGD	Projected Gradient Descent (baseline method)