深度强化学习的对抗鲁棒性评测

Evaluating the Adversarial Robustness of Deep Reinforcement Learning

（申请清华大学工学硕士学位论文）

培养单位：计算机科学与技术系
学　　科：计算机科学与技术
研 究 生：刘大为
指导教师：朱　军　　教　授

二〇二一年五月

Evaluating the Adversarial Robustness of Deep Reinforcement Learning

Thesis Submitted to
Tsinghua University
in partial fulfillment of the requirement
for the degree of
Master of Science
in
Computer Science and Technology

by

Davide Liu

Thesis Supervisor: Professor Zhu Jun

May, 2021

THESIS REVIEWERS AND DEFENSE COMMITTEE

公开评阅人名单

黄民列	副教授	清华大学
胡晓林	副教授	清华大学

答辩委员会名单

主席	周立柱	教授	清华大学
委员	朱军	教授	清华大学
	邓志东	教授	清华大学
	汪东升	教授	清华大学
	黄民列	副教授	清华大学
秘书	戴音	讲师	清华大学

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（1）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（2）为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容；（3）按照上级教育主管部门督导、抽查等要求，报送相应的学位论文。

本人保证遵守上述规定。

（保密的论文在解密后遵守此规定）

作者签名：______________ 导师签名：______________
日　　期：______________ 日　　期：______________

摘　要

现有文献已经广泛研究了针对深度学习模型的对抗攻击，并提出了几种防御方法来保护神经网络免受恶意对抗样本的攻击。然而，大多数工作主要集中在处理图像数据（例如，图像分类任务）的卷积神经网络上。近年来，由于卷积神经网络具有从原始像素提取重要图像特征的能力，这些模型还广泛应用于深度强化学习算法中。这些算法只需向模型提供像素观察值并据此预测特定的动作，即可从头开始学习策略。尽管如此，该领域的大多数研究都针对提升智能体性能，即最大程度地提高它们获得的收益。现有针对智能体鲁棒性的研究中，较少关注一类将其引向危险状态的对抗性攻击，这类攻击将减少智能体所获得的收益。本文首先研究了一些最常见的深度强化学习算法的鲁棒性。其次，分析了上述算法面对设计恶意策略的攻击的脆弱性。第三，研究了它们在策略和算法之间的可转移性。最后，研究了在防御机制存在的情况下，基于图像的攻击制定有效的对抗性观察的能力。本文提出，除针对策略攻击外，还应该针对图像攻击方法来评估策略的鲁棒性，因为某些图像攻击与特定的策略攻击结合可以更好地发挥作用。本文提出采用与受害方策略相同算法训练的替代策略进行的对抗性观察具有更好的可传递性，而且使用与替代策略算法相似的算法来训练替代策略时，可传递性更为有效。

关键词：强化学习；对抗决策；对抗攻击；鲁棒性

ABSTRACT

Adversarial attacks against deep learning models have been widely studied in literature, and several defense methods have then been proposed to protect neural networks against malicious adversarial examples. However, most of the work is mainly focused on convolutional neural networks dealing with image data such as for image classification tasks. In recent years, thanks to their ability to extract important image features from raw pixels, those models have also found wide application in deep reinforcement learning algorithms which are able to learn a policy from scratch simply by feeding them with pixel observations and predict specific actions accordingly. Still, most of the research in this field is addressed on improving agents' performance in terms of maximizing their earned reward, while relatively little has been done to study their robustness against adversarial attacks that are able to mislead those policies toward dangerous states, thus reducing the obtained reward. In this thesis work, we first investigated the robustness of some of the most common deep reinforcement learning algorithms. Second, we showed their vulnerability against attacks that design malicious policies. Third, we studied their transferability propriety across policies and algorithms. Finally, we investigated the ability of image-based attacks to craft effective adversarial observations, even in presence of defense mechanisms. We hypothesize that policy robustness should be evaluated not only against policy attacks but also against image attack methods since some image attacks work better in combination with particular policy attacks. Moreover, we showed that adversarial observations crafted on surrogate policies trained with the same algorithm of the victim policy benefit of a better transferability, and we also empirically demonstrated that transferability is more effective when the surrogate policy is trained with an algorithm similar to the algorithm used to train the victim policy.

Keywords: Reinforcement learning; Adversarial policies; Adversarial attacks; Robustness

TABLE OF CONTENTS

摘　要 I

ABSTRACT II

TABLE OF CONTENTS III

LIST OF FIGURES AND TABLES VI

LIST OF SYMBOLS AND ACRONYMS IX

CHAPTER 1　INTRODUCTION 1
- 1.1　Major breakthroughs of deep reinforcement learning 1
- 1.2　Importance of training robust policies 2
- 1.3　Thesis overview 2

CHAPTER 2　DEEP REINFORCEMENT LEARNING 5
- 2.1　Deep reinforcement learning 5
- 2.2　Deep Q-Network 5
- 2.3　Advantage actor-critic 6
- 2.4　Proximal Policy Optimization 7

CHAPTER 3　ADVERSARIAL ATTACKS ON IMAGES 9
- 3.1　Adversarial attacks on images 9
  - 3.1.1　Generating adversarial examples 11
  - 3.1.2　FGSM 11
  - 3.1.3　I-FGSM 11
  - 3.1.4　MI-FGSM 12
  - 3.1.5　PGD 12
  - 3.1.6　C&W 13
- 3.2　Defending against adversarial examples 13
  - 3.2.1　Adversarial training 13

CHAPTER 4　ADVERSARIAL ATTACKS ON POLICIES 15
- 4.1　Adversarial attacks on policies 15
  - 4.1.1　Single-agent adversarial attacks 16

4.1.2 Uniform attack 17
4.1.3 Strategically-timed attack 17
4.1.4 Enchanting attack 18
4.1.5 Critical point attack 18
4.1.6 Critical strategy attack 19
4.1.7 Adversarial policy 20
4.2 Defense methods against single-agent adversarial attacks 21
4.2.1 DQN adversarial training 21
4.2.2 A2C-PPO adversarial training 21
4.3 Multi-agent adversarial attacks 22
4.3.1 Multi-agent adversarial policy 23
4.3.2 Defense methods against multi-agent adversarial attacks 23
4.4 Applying FGSM to policies 24
CHAPTER 5 STUDYING POLICY ATTACKS TRANSFERABILITY 25
5.1 Studying policy attacks transferability 25
5.1.1 Environments 25
5.1.2 Experiments settings 26
5.2 Results 28
5.2.1 Attacking Pong-DQN 29
5.2.2 Attacking Pong-A2C 30
5.2.3 Attacking Pong-PPO 31
5.3 Findings 32
5.3.1 Uniform attack vs strategically-timed attack 32
5.3.2 Critical point vs critical strategy attack 33
5.3.3 Adversarial policy and future search directions 34
5.3.4 Attack feasibility in common scenarios 34
5.3.5 Online vs offline attacks 36
CHAPTER 6 STUDYING POLICY ATTACKS TRANSFERABILITY AGAINST DEFENDED POLICIES 40
6.1 Studying policy attacks transferability against defended policies 40
6.1.1 Experiments settings 40

6.2 Results 42
6.2.1 Attacking adversarially trained Pong-DQN 42
6.2.2 Attacking adversarially trained Pong-A2C 43
6.2.3 Attacking adversarially trained Pong-PPO 43
6.3 Findings..44
6.3.1 Performance of similar policies 45
6.3.2 Performance of similar algorithms..45
CHAPTER 7 BENCHMARK OF EFFECTIVENESS OF IMAGE ATTACK METHODS AGAINST DEFENDED POLICIES...............................46
7.1 Benchmark of the effectiveness of image attack methods against defended policies 46
7.1.1 Experiments settings..47
7.1.2 Benchmark of untargeted white-box attacks....................................48
7.1.3 Benchmark of targeted white-box attacks.......................................49
7.1.4 Findings 49
7.2 Number of attacked policies 52
7.3 Implementation details 53
CHAPTER 8 Conclusions..59
8.1 Summary 59
8.2 Future research directions...59
REFERENCES 61
ACKNOWLEDGEMENTS..64
声 明...65
RESUME..66
COMMENTS FROM THESIS SUPERVISOR...67
RESOLUTION OF THESIS DEFENSE COMMITTEE...68

LIST OF FIGURES AND TABLES

Figure 5.1 Example of the behavior of the divergence function T for the game of Pong: the observation on the left (a) has a very low level of T since the probability that the agent fails is 0. The observation in the middle (b) presents a discrete T since considering the ball moving simply toward the right of the screen along the x-axis the agent can still make the ball bounce back. The third observation (c) has a very high value of T since there is no way the agent can save the ball. 29

Figure 5.2 The *reward vs. attack frequency* curves of 5 policy attacks with FGSM under the l_∞ norm attacking a Pong-DQN policy. 37

Figure 5.3 The *reward vs. attack frequency* curves of 5 policy attacks with FGSM under the l_∞ norm attacking a Pong-A2C policy. 38

Figure 5.4 The *reward vs. attack frequency* curves of 5 policy attacks with FGSM under the l_∞ norm attacking a Pong-PPO policy. 39

Figure 6.1 The *reward vs. attack frequency* curves of 2 policy attacks with FGSM under the l_∞ norm attacking a Pong-DQN policy defended with FGSM adversarial training with $epsilon = 0.01$. Attacks have been performed with $epsilon = 0.05$ to show the drop in reward as the attack frequency increases. 43

Figure 6.2 The *reward vs. attack frequency* curves of 2 policy attacks with FGSM under the l_∞ norm attacking a Pong-A2C policy defended with FGSM adversarial training with $epsilon = 0.01$. Attacks have been performed with $epsilon = 0.05$ to show the drop in reward as the attack frequency increases. 44

Figure 6.3 The *reward vs. attack frequency* curves of 2 policy attacks with FGSM under the l_∞ norm attacking a Pong-PPO policy defended with FGSM adversarial training with $epsilon = 0.01$. Attacks have been performed with $epsilon = 0.05$ to show the drop in reward as the attack frequency increases. 44

Figure 7.1 The *reward vs perturbation* curves of 5 DQN policies trained to play Pong and defended with 5 different defense methods (including no defense) under the l_∞ norm attacking with increased perturbations strength. Perturbations have been crafted in an untargeted way. 54

Figure 7.2 The *accuracy vs perturbation* curves of 5 DQN policies trained to play Pong and defended with 5 different defense methods (including no defense) under the l_∞ norm attacking with increased perturbations strength. Perturbations have been crafted in an untargeted way. 55

Figure 7.3 The *reward vs perturbation* curves of 5 DQN policies trained to play Pong and defended with 5 different defense methods (including no defense) under the l_∞ norm attacking with increased perturbations strength. Perturbations have been crafted in a targeted way...56

Figure 7.4 The *accuracy vs perturbation* curves of 5 DQN policies trained to play Pong and defended with 5 different defense methods (including no defense) under the l_∞ norm attacking with increased perturbations strength. Perturbations have been crafted in a targeted way...57

Figure 7.5 **First row**: Comparison of the effect of noise injections on Pong environment observations with different strength of perturbations. The most right one is a clear observation and from the right to the left have been injected noise with increasing perturbations $\epsilon = [0.1, 0.01, 0.001, 0.0001]$. Perturbations have been crafted with FGSM attacking a policy trained with PPO. **Second row**: Comparison of the effect of noise injections on Pong environment observations with different strength of perturbations. The most right one is a clear observation and from the right to the left have been injected noise with increasing perturbations $\epsilon = [0.1, 0.01, 0.001, 0.0001]$. Perturbations have been crafted with PGD attacking a policy trained with PPO.................58

Table 4.1 General overview of the attacks methods reviewed in this thesis and their corresponding categories..16

Table 5.1 Average score over 100 episodes for policies trained with DQN, A2C, and PPO. It can be clearly seen that PPO, being a more sophisticated algorithm than the other two also performs much better. Pong is easy enough that most algorithms can successfully train a policy to defeat the opponent............27

Table 5.2 Requirements needed for each evaluated policy attack method to work properly...35

Table 6.1 Average reward scored by three policies trained with DQN, A2C and PPO respectively and defended with FGSM Adversarial training with $epsilon = 0.01$ under l norm. 42

LIST OF SYMBOLS AND ACRONYMS

DRL	Deep Reinforcement Learning
DQN	Deep Q-Network
MDP	Markov Decision Process
A2C	Advantage-Actor-Critic
PPO	Proximal policy optimization
SGD	Stochastic Gradient Descent
θ	Weights of a Neural Network
π	Reinforcement Learning Policy
FGSM	Fast Gradient Sign Method
MI-FGSM	Momentum Iterative-Fast Gradient Sign Method
PGD	Projected Gradient Descent

CHAPTER 1 INTRODUCTION

This chapter gives a brief introduction about the major breakthroughs of the field of deep reinforcement learning starting from a simple DQN[1] to more advanced algorithms such as AlphaZero[2] and AlphaStar[3]. It gives an idea of the complexity of the environments and the challenges these AIs have to overcome and explains why training robust policies is so important for most applications in the real world.

1.1 Major breakthroughs of deep reinforcement learning

With the advent of deep learning, neural networks have quickly invaded the field of reinforcement learning giving birth to a new branch called deep reinforcement learning (DRL). First steps were taken back in 2013 when[1] used a Deep Q-Network (DQN) to learn to play a number of classic Atari games only by observing their environment from raw pixels using a simple convolutional network. Starting from zero knowledge, the agent explores strategical policies by itself, and after training for a sufficient number of frames, it hopefully learns to play on the target environment, sometimes even achieving super-human performance. Later on, the DRL field had another important breakthrough in 2016, when a famous AI developed by DeepMind known as AlphaGo[4] defeated for the first time in the history a 9-dan Go player, Lee Sedol, for 4 games to 1 playing against him without handicaps. The strategy game of GO was, at the time, considered very hard to be solved by AI algorithms due to its enormous number of possible states (about 10^{170}) an agent has to deal with. One year later, its evolution, AlphaZero[2], even managed to defeat Ke Jie who at the time of the match was ranked first among all human players worldwide. One year later, in 2019, a more general version of AlphaZero, MuZero[5], successfully mastered the game of Chess, Go, and Shogi without knowing the rules of the games and is even able to generalize to single-agent environments such as Atari games without changing its architecture. In 2018, a new agent developed by Uber, Go-Explore[6], for the first time solved the game Montezuma's Revenge with a score of over 43,000 points, which is almost 4 times the previous state of the art. Montezuma's Revenge is well known for being a very hard environment for RL agents since its rewards are very sparse, and many tasks have to be done before getting any score. One year later, in 2019, another agent developed by DeepMind, AlphaStar[3], managed to beat a top-10 player of Starcraft for 5 to 0 as well as defeating 86% of medium level players. Finally, also in 2019, Tencent MOBA-playing AI system[7] beat 99.81% of professional human opponents in full 1v1 games of the MOBA *Honor of Kings*. The actor-critic agent included several novel strategies such as control dependency decoupling, action mask, target attention, and dualclip PPO. Master the game of *Honor of Kings* is far more challenging than Go since in a normal match could be encountered about 10^{600} possible states.

1.2 Importance of training robust policies

However, besides performance, safety also plays a fundamental role when deploying DRL-based models in real-world applications. An agent that achieves super-human skills in a certain task but that fails to complete its goal when its inputs are subjected to minimal perturbations can't be considered as a reliable system. Even worse, this lack of robustness may even lead to dangerous situations. In a scenario where an agent driving an autonomous car is fed with adversarial examples, it may be deceived by the environment to choose to take sub-optimal actions, thus degrading its performance. In an even more dangerous scenario, an attacker may craft a sequence of adversarial samples so to lead the agent toward a dangerous target state, thus causing a possible collision or committing an infringement. Besides self-driving cars and videogames, DRL finds application in other critical systems such as training robots used in manufacturing, enhancing the performance of electric power systems, and evaluating trading strategies in the financial sector. Given the critical issues of the above-mentioned scenarios, it is reasonable to think that robustness plays a critical role and these systems should be prevented from being fooled by adversarial observations crafted by a malicious agent. Hence, the direction of this thesis project concerns studying the effects of adversarial attacks and defenses on deep reinforcement learning algorithms so to evaluate their robustness under different levels of adversary knowledge and their possible defenses, so to make these systems more reliable and usable in the real world.

1.3 Thesis overview

The work in this thesis thus investigates the robustness of some of the most common deep reinforcement learning algorithms and shows their vulnerability both against malicious policies with the goal to mislead the agent toward sub-optimal states and against adversarial attacks aimed at crafting more effective adversarial observations. Malicious or adversarial policies are generated employing policy adversarial attacks which will be introduced in chapter 4. Since the evaluated policies take images as input, policy adversarial attacks, in turn, perform image attacks on input observations so to change the victim policy's actions distribution toward an adversarial one. For this reason, we believe that, in order to do a comprehensive study on policies robustness, merely evaluating policy attacks is not enough since these attacks are conditioned by the performance of the image attacks they rely on. Image adversarial attacks will be introduced in chapter 3.

However, before going on with the main body of this thesis, it is important to have a clear concept of what is a *policy*, an *algorithm*, and what we mean for *transferability* in our context. A *policy* is a model that outputs a probability distribution over actions and it is trained using a reinforcement learning *algorithm*. However, two policies trained with the same algorithm may not necessarily be the same policy since a change in the random seed used to train the network implementing each policy would result in its weights having different parameters. Note that two policies trained or equipped with different defense mechanisms are also different policies since the defense method would influence their behavior. Furthermore, we have *policy transferability* when adversarial perturbations for the target policy are generated using another policy trained with the same algorithm and for the same task. Similarly, in the case of *algorithm transferability*, adversarial perturbations for the target policy are generated using another policy trained with a different algorithm for the same task. This is the same concept of attack transferability on image classification models but applied to the reinforcement learning domain. Moreover, when in the following chapters we are going to refer to a policy trained with algorithm A on environment E we are simply going to call it $E - A$. For instance, Pong-DQN corresponds to a policy trained with DQN on the environment of Pong.

Overall, the main contributions of this work are the following:

- **Study policy attacks transferability over policies and algorithms**: We computed the *reward vs attack frequency* curves of different policy attacks under a fixed threat model so to measure their effectiveness and their transferability against policies and algorithms[8]. We repeated the experiment for a total of 3 algorithms, 5 policy attacks, and 2 environments.
- **Study policy attacks transferability against defended policies**: We computed the *reward vs attack frequency* curves of different policy attacks under a fixed threat

model so to measure their effectiveness and their transferability against defended policies and defended algorithms. We repeated the experiment for a total of 3 algorithms, 2 policy attacks, and 2 environments.

- **Benchmark of the robustness of defended policies against white-box image attacks methods**: We compared the curves of victim policies' average reward and success rate of different white-box attack methods attacking input observations under a fixed threat model and over increasing values of perturbations' strength ϵ. Victim policies have also been protected with different defense methods so as to have a better idea of the effectiveness of each attack and the robustness of these defended policies. These curves, introduced in[9], are also known as robustness curves and are used as major evaluation metrics to measure image classification models' robustness.

The present thesis work is structured in the following way. Chapter 2 reviews the policies trained to conduct the experiments and the evaluated environments, chapter 3 introduces the concept of adversarial attacks and defenses on images and explains some common white-box attacks used as backbone to implement policies adversarial attacks and defenses. Chapter 4 gives a general overview of adversarial attacks on policies as well as possible defense methods. Chapters 5, 6 and 7 show, analyze and discuss the results obtained in each of the three experiments respectively. Finally, 8 gives a general discussion and conclusion about the work that has been carried out and points out some future research directions.

CHAPTER 2 DEEP REINFORCEMENT LEARNING

This chapter first gives a brief overview of the theory behind reinforcement learning and then introduces the three deep learning algorithms used during the experiments carried out in this thesis work, namely DQN[1], A2C[10] and PPO[11].

2.1 Deep reinforcement learning

Deep reinforcement learning (DRL) usually involves one or more agents interacting with an environment following a policy π, like in a Markov Decision Process (MDP), where their goal is to improve their policy such to maximize their earned reward. More specifically, during training, the agent is provided with an observation s_t of the environment for each timestep $t = 0, 1, 2, ...$, and it has to respond with an action a_t. Afterwards, the environment provides a reward $r(a_t, s_t)$, the next state s_{t+1}, and a discount factor γ_t. As regards to the selection of actions, they are selected according to a policy π, modeled by a neural network with weights θ that defines a probability distribution over the actions for each state. So, starting from state s_t encountered at time t, we can define the discounted cumulative reward R_t as

$$R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1}. \tag{2-1}$$

Thus, considering an episode starting at time $t = 0$ and terminating at time T, the expected discounted reward of a policy π is defined as

$$R_0 = \sum_{i=0}^{T-1} \mathbb{E}_{a_i \sim \pi(s_i)}[\gamma^i r(a_i, s_i)]. \tag{2-2}$$

Following, we are going to review in more details the three algorithms that have been employed to conduct this thesis work.

2.2 Deep Q-Network

Deep Q-network (DQN)[1] is a neural network used to implement the Q-learning algorithm. More specifically, it learns an estimate of the Q-value $Q(s, a, \theta)$ parameterized by the parameters θ of the network (online network). The Q-value, which measures the value of choosing a particular action when in a particular state, can be defined according to the following recursive formula:

$$Q_{\theta_t}(s_t, a_t) = r_t + \gamma Q_{\theta_t}((s_{t+1}, a_{t+1})|s_t, a_t, \theta_t). \tag{2-3}$$

Its architecture is composed of a first convolutional neural network (CNN) that takes as input a state s_t and learns to detect increasingly abstract features from it. Subsequently, a dense classifier maps the high-level extracted features to an output layer with one neuron per action in order to approximate the corresponding action value. The parameters of the network are trained by gradient descent to minimize the following loss function:

$$L_t(\theta_t) = (r_{t+1} + \gamma_{t+1} \max_{a'} Q_{\theta^-}(s_{t+1}, a') - Q_{\theta_t}(s_t, a_t))^2. \tag{2-4}$$

During training, a batch of experiences is randomly picked from the replay memory which is a sort of dataset storing the agent's past experiences defined as the tuple $(s_t, a_t, r(a_t, s_t), s_{t+1})$. The experience replay technique prevents the network from learning from strongly correlated experiences that break the i.i.d. assumption of many popular stochastic gradient-based algorithms, as well as avoiding the rapid forgetting of possibly rare experiences that would be useful later on. Because of the experience replay, DQN is an off-policy algorithm since some sampled experiences might be explored with an older version of the online policy. The gradient of the loss is then back-propagated only into the parameters θ of the online network which is used to select actions. The term θ^- represents the parameters of a target network, a periodic copy of the online network which is not directly optimized and is used to estimate the maximum expected reward in equation (2-4). During training, the parameters of the online network are periodically copied into the ones of the target network so to update its weights. The advantage introduced by having an independent target network is to make the target Q-value independent with respect to the predicted Q-value when computing the loss so as to increase training stability. Overall, the Q-value calculated by the target network has more accuracy since it has access to at least the first reward term to calculate it. In order to encourage exploration, in particular during the early stage of training, greedy actions are taken with probability 1-ϵ otherwise are performed random actions so to discover new policies that may lead to higher rewards.

2.3 Advantage actor-critic

Advantage Actor-Critic (A2C)[10] is formed by a neural network parameterized by θ with 2 output layers: the first one is a softmax layer with weights θ_π and outputs a policy $\pi(a_t|s_t; \theta_\pi)$ (actor), the second one consists of a linear output for the value function

$V(s_t;\theta_v)$ (critic) parameterized by θ_v. Both the policy and the value function are periodically updated by gradient descent computing the gradient as

$$\nabla_\theta = \log \pi(a_t|s_t;\theta_\pi)A(a_t,s_t;\theta_v), \tag{2-5}$$

$$A(a_t,s_t;\theta_v) = R_t - V(s_t;\theta_v), \tag{2-6}$$

where $A(a_t,s_t;\theta_v)$ is an estimate of the advantage function which measures the relative importance of each action respect to the state s_t. The formula to compute the gradient derives from the loss function which is simply the negative log likelihood of the predicted action multiplied by the advantage. It also exists an asynchronous version called A3C which relies on several asynchronous agents, each of them interacting with its own copy of the environment, that accumulate gradients and periodically send them to the global network to perform updates. Agents running in parallel, possibly using different exploration policies, are likely to explore different parts of the environment, thus increasing samples' diversity and decreasing their correlation. Other benefits brought by using asynchronous agents consist of a linear reduction of training time based on the number of parallel agents, and the absence of an experience replay. This method is on-policy since it sequentially learns over its observed states. Note that Actor-Critic doesn't have an explicit hyper-parameter to regulate its grade of exploration since it trains a stochastic policy where exploration is done by sampling actions according to the actions probabilities defined by the actor network. Over the course of training, the policy will learn to become progressively less random.

2.4 Proximal Policy Optimization

Proximal Policy Optimization (PPO)[11] is a model-free on-policy algorithm that aims to balance among ease of implementation, sample complexity, and ease of tuning. PPO tries to compute an update at each step that minimizes the cost function while ensuring that the deviation from the previous policy is relatively small:

$$L^{CLIP}(\theta) = \mathbb{E}_t[\min(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}A_t, clip(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon)A_t)], \tag{2-7}$$

where the clipping operator prevents the maximization of the objective to lead to an excessively large policy update by constraining it in the interval $[1-\epsilon, 1+\epsilon]$ and A_t is the advantage function computed as in equation (2-6). The architecture of the network involved in this algorithm is similar to A2C giving as output a distribution over actions and a value function used to compute the value of the advantage. Both functions are periodically updated after collecting a certain amount of trajectories. Moreover, PPO can run N parallel actors to collect the data, and then sample mini-batches of T timesteps to optimize the loss for K epochs using SGD.

CHAPTER 3 ADVERSARIAL ATTACKS ON IMAGES

Since the policy attacks included in this thesis work by perturbing the image observations given in input to the victim policies, this chapter reviews some of the most common image attacks, their threat model and introduces some common defense methods to counter them.

3.1 Adversarial attacks on images

Since the advent of deep convolutional networks used for image classification[12], the computer vision field has been subjected to many important breakthroughs with the aim to further improve the classification accuracy obtained on many images classification tasks[13][14]. However, despite their high accuracy, those models lack robustness since they can be easily deceived by adversarial examples[15]. More formally, an adversarial example is a *sample input data that has been modified very slightly in a way that is intended to cause a machine learning classifier to misclassify it*[16]. In fact, to be effective, an adversarial example should be misclassified by deep learning models, but not by the human brain. Therefore, only small changes can be made to the original input image x (legitimate example) to craft the adversarial example x_{adv}=x+δ, where δ is also known as adversarial noise. The distance introduced by the noise between x and x_{adv} is usually defined by the l_p norm of the difference between the original and the adversarial sample for some p=0,..,∞. Hence, an adversarial example x_{adv} has to satisfy the constraint $||x_{adv} - x||_p \leqslant \epsilon$ (adversarial constraint), where smaller ϵ values correspond to smaller input perturbations, thus leading to less perceptible changes under the condition that x_{adv} is misclassified. Moreover, images adversarial attacks can be designed to achieve two different goals:

- **Untargeted**: Untargeted attacks aim at making the model predict any class different from the correct one without targeting at any desired class. Formally speaking, given a classifier f and the true label y, an adversarial example would cause $f(x_{adv}) \neq y$. Hence, untargeted attacks' goal consists of maximizing the loss of the attacked model respect to the true label y, namely, max $L(x_{adv}, y)$, under the assumption $||x_{adv} - x||_p \leqslant \epsilon$. Given the relaxed constraints the adversarial examples are subjected to, these kinds of attacks are usually easier to perform.

- **Targeted**: Conversely, targeted attacks try to mislead the model's prediction toward a specific class y', that is, $f(x_{adv}) = y'$, where $y' \neq y$. The loss function can then be formulated as min $L(x_{adv}, y')$ always under adversarial constraints. In this way, given the input x_{adv}, it will be more likely that the attacked model would predict the target class y' rather than any other class. More sophisticated policy attacks usually require crafting adversarial observation in a targeted way so to force the victim agent to perform some required actions.

For example, if we want an agent simply to have its performance degraded, attacking under untargeted settings would be enough since preventing it to take the best action would naturally lower its total earned reward. However, if our goal is to control the agent by making it choose some predefined actions such as to lead it to a particular state, then adversarial attacks should be performed under targeted settings. Finally, adversarial attacks are also divided into two main categories depending on how much information is available regarding the model under attack:

- **White-box attacks**: Under this setting, the adversary has full access and knowledge of the model, that is, the architecture of the model, its parameters, gradients, and loss respect to the input as well as possible defense mechanisms are known to the attacker[15]. Thus, it is not particularly difficult to attack models under this condition, and common methods exploit model's output gradients to generate adversarial examples. Only attacks belonging to this category have been evaluated in this work.
- **Black-box attacks**: In this category of attacks, the adversary has zero or very little knowledge about the model. Thus existing methods often rely on training a similar model or an ensemble of them. These methods work because, generally, adversarial examples that fool one model are likely to fool another similar model. Furthermore, in practice, this is the most likely kind of attack, since, in normal circumstances, attackers don't have the possibility to access much of the models' knowledge. Other methods exploit knowledge about the accuracy of the prediction or only the label to craft adversarial examples[17].

Both white and black-box attacks are possible when attacking DRL algorithms and it depends on how much knowledge is known regarding the network defining the policy of the agents. Conversely, adversarial defenses aim to protect deep learning models from adversarial attacks, namely, making models more robust against adversarial examples. In this context, research on adversarial robustness resembles a minimax game where attackers constantly try to exploit more powerful techniques to fool deep learning models while, at the same time, defenders have to invent new defense methods to guard against these malicious attacks.

3.1.1 Generating adversarial examples

In the next sections, have been reported some of the most common white-box attacks on images that are often also used to attack DRL agents. Moreover, these methods can also be used to attack images under black-box settings by crafting adversarial examples attacking similar models and then exploiting the transferability propriety of the adversarial examples to fool the target model[18] or they can be directly applied to perform black-box attacks after estimating gradients by querying the target model[19].

3.1.2 FGSM

Fast Gradient Sign Method (FGSM)[15] is a basic one-step gradient-based approach that is able to find an adversarial example in a single step by maximizing the loss function $L(x_{adv}, y)$ with respect to the input x and then adding back the sign of the output gradients to x so to produce the adversarial example x_{adv}

$$x_{adv} = x + \epsilon \cdot sign(\nabla_x L(x, y)), \tag{3-1}$$

where $\nabla_x L(x, y)$ is the gradient of the loss respect to the input x, and the equation is expected to meet the l_∞ norm bound by design. This method works because adding a perturbation to the legitimate input such to maximize its loss respect to the correct label y decreases the likelihood that y could be predicted given the input x_{adv}. Mathematically, it moves the adversarial example in one direction toward the border between the true class and some other class[18]. This method is sometimes implemented without the *sign* operator (FGM) and it yields similar results to the version with sign[20].

3.1.3 I-FGSM

Basic iterative methods[16] iteratively apply FGSM with a small step size α. Thus, the iterative version of FGSM (I-FGSM) can be expressed as

$$x_{adv}^{t+1} = x_{adv}^{t} + \epsilon \cdot \alpha \cdot sign(\nabla_x L(x_{adv}^{t}, y)), \tag{3-2}$$

where $x_{adv}^0 = x$ is the legitimate example. There are several ways to make the adversarial example satisfy the norm bound. For example, x_{adv} could be clipped into the ϵ vicinity of x or set $\alpha = \epsilon/T$ with T being the number of iterations.[16] proved that iterative methods exploit much finer perturbations which do not destroy the image even with higher ϵ and at the same time confuse the classifier with a higher rate. Their drawback is that iterative methods are a little bit slower than their one-step counterparts.

3.1.4 MI-FGSM

Momentum iterative gradient-based methods[18] integrate momentum into iterative fast gradient method to generate adversarial examples satisfying the l_p norm bound. Traditionally, momentum is a technique for accelerating gradient descent algorithms by accumulating a velocity vector in the gradient direction of the loss function across iterations. However, this concept can also be applied to generate adversarial examples and obtain tremendous benefits. As it was for the learning rate update, the first step consists of updating the momentum g_t by accumulating the velocity vector in the gradient direction as

$$g_{t+1} = \mu \cdot g_t + \frac{\nabla_x L(f(x, y)}{||\nabla_x L(x, y)||_p}, \tag{3-3}$$

where μ is a decay factor. Next, the adversarial example x^t_{adv} is perturbed in the direction of the sign of g_t with a step size α as

$$x^{t+1}_{adv} = x^t_{adv} + \alpha \cdot sign(g_{t+1}). \tag{3-4}$$

In each iteration, the current gradient $\nabla_x L(x, y)$ is normalized by the l_p distance of itself because the authors noticed that the scale of the gradients in different iterations varies in magnitude.

3.1.5 PGD

Projected gradient descent (PGD)[21] is another iterative algorithm that exploits projected gradient descend to iteratively craft adversarial examples as:

$$x^{t+1}_{adv} = \pi_{x+S}(x^t_{adv} + \alpha \cdot sign(\nabla_x L(x^t_{adv}, y))), \tag{3-5}$$

where S is the set of all the allowed perturbations. Projected gradient descent performs one step of standard gradient descent, and then clips all the coordinates to be within the l_p ball. Moreover, in order to explore a large part of the loss landscape, the algorithm is restarted from many points within the l_p ball around data points taken from the evaluation set. Thanks to this large number of observations, the authors realized that all the local maxima found by PGD have similar loss values, both for normally trained networks and for adversarially trained networks, thus pointing out that robustness against the PGD adversary yields robustness against all first-order adversaries such as SGD based attacks.

This conclusion also leads to the fact that as long as the adversary only uses gradients of the loss function with respect to the input, it will not find significantly better local maxima than PGD.

3.1.6 C&W

The method proposed by Carlini & Wagner[22] relies on the initial formulation of adversarial example and formally defines the problem of finding an adversarial instance for an image x as a minimization problem of a continuous function:

$$\min \ ||\delta||_p + c \cdot L(x + \delta, y), \tag{3-6}$$

under the condition that $(x + \delta) \in [0, 1]^n$. The function f is an objective function such that $L(x + \delta, y) \neq y$ if and only if $L(x + \delta, y) <= 0$. Finally, the term c is a suitable chosen hyper-parameter. Moreover, to ensure that the modification yields a valid image, the adversarial noise δ is constrained such that $0 \leqslant x_i + \delta_i \leqslant 1 \ \forall i$ (here assuming image pixels to be in the range [0,1]). One way to do it, is to replace $(x+\delta)$ with $(1+\tanh(w))/2$ so that the optimization problem in (3-6) becomes an unconstrained minimization problem in w.

3.2 Defending against adversarial examples

Extensive research in developing effective defense mechanisms in order to build robust models and safeguard them against adversarial attacks has also been conducted. One very promising technique is robust training which aims to make a classifier robust against small internal perturbations. Some possible strategies are based on adversarial training by adding generated adversarial examples to the training data[15], defensive distillation which consists of retraining a network using previously generated soft-labels[23], or another technique consists of training robust models with regularization such to train the defended model to ignore small perturbations[24]. Another category of defense methods that we are going to examine consists of input transformation. This method is not applied during training but only during inference by transforming the inputs right before feeding them to the classifier with the aim to make adversarial perturbations less effective.

3.2.1 Adversarial training

Adversarial training (AT) is a very simple and intuitive defense method to protect a model against adversarial examples. The first step consists of generating adversarial examples using different attack methods on the target model. In the second step, these adversarial examples are merged to the original training set so to form an augmented training set and finally the target model is retrained on the augmented training set. When adversarial examples are crafted with PGD we have PGD-adversarial training which can train very robust deep networks but it is much more expensive than traditional training due to the iterative design of PGD. In contrast, FGSM-adversarial training is typically faster but less effective[25].

3.2.1.1 JPEG compression

JPEG compression[26] is a simple input transformation method that converts each input image to JPEG before it is fed to the target network. It has been studied that compressing an image can partially remove possible adversarial perturbations and, at the same time, elude the human perception that no transformation has been applied. However, in practice, this method is not very effective since it degrades the performance of the model while not cleaning completely an image from all the adversarial perturbations. To remedy this problem, [27] proposes to vaccinate a policy by retraining it on JPEG compressed images multiple times on multiple compression qualities, and use an ensemble of these models to get the final classification label.

3.2.1.2 Feature squeezing

Feature squeezing[28] is another input transformation method that reduces the search space available to an adversary by compressing different features vectors in the original space into a single sample. Digital computers usually represent images as an array of pixels each of them representing a specific color (RGB images) or a shadow of grey (greyscale images). Thus, reducing bit depth can reduce the space that an adversarial has to craft perturbations possibly limiting the drop in accuracy that this compression may lead to. For example, an 8-bit greyscale image provides $2^8 = 256$ values for each pixel which if reduced to 5-bits we would have only $2^5 = 32$ values for each pixel that could be changed to create an adversarial example.

CHAPTER 4 ADVERSARIAL ATTACKS ON POLICIES

This chapter explains in detail the policy attacks adopted in the experiments and defines their threat model. It gives an overview of both single and multi-agent attacks as well as possible defense methods.

4.1 Adversarial attacks on policies

As for adversarial attacks in other domains such as image or graph data, we can always define the properties of threat models, that is, a set of assumptions about the adversary's goals, knowledge, and, in the reinforcement learning case, also the number of controllable agents interacting in the same environment. These conditions are specified by threat models so to determine under which conditions a defense is designed to be effective. Firstly, we can classify adversarial attacks on policies into two different categories based on the goal they are designed to achieve[29]:

- **Untargeted**: In the case of untargeted attacks, the goal is to decrease the performance of the victim agent, namely, causing it to take sub-optimal actions with respect to its learned policy so to reduce its total earned reward.
- **Targeted**: Targeted attacks aim to control the agent by making it take a sequence of some predefined actions so to lure it to a particular state. This kind of attack is more difficult to achieve since causing the agent to take a malicious specific action and leading it to a specific state are two tasks whose solutions are not immediate.

Secondly, depending on the knowledge the attacker has about the model defining the agent's policy and the algorithm used, we can individuate other three categories of attacks[30]:

- **White-box**: full knowledge about environment, training algorithm, and network.
- **Black-box I: Algorithm is known but not the policy**: the adversary has access to the training environment and knowledge of the training algorithm and hyperparameters. It also knows the neural network architecture of the target policy network, but not its random initialization.
- **Black-box II: Both algorithm and policy are not known**: under these settings, the adversary has no knowledge of the training algorithm or hyperparameters, but it only knows the environment. Hence, it may need to craft its attacks using a different

algorithm. In practice, it is often the case that an adversary does not have complete access to the neural network of the target policy[8].

Thirdly, we have another important distinction based on whether other controllable agents can interact in the same environment or not[31]:

- **Single-agent**: In the environment is present only one agent that is not directly controllable, thus it can be only deceived by adding adversarial noise to the input observations it receives.
- **Multi-agent**: In the same environment of the victim agent are also present other agents that can be controlled and trained to assume a particular behavior that could modify the environment such as to create observations that may deceive the victim.

In the next sections, we are first going to discuss the single-agent scenario and then we will briefly focus on the multi-agent one. All the reviewed attacks are summarized in table 4.1.

Table 4.1 General overview of the attacks methods reviewed in this thesis and their corresponding categories.

Attack method	Goal	Scenario
Uniform Attack	Untargeted	Single-agent
Strategically-Timed Attack	Untargeted	Single-agent
Enchanting Attack	Targeted	Single-agent
Critical Point Attack	Untargeted	Single-agent
Critical Strategy Attack	Untargeted	Single-agent
Adversarial Policy	Untargeted	Single-agent/Multi-agent

4.1.1 Single-agent adversarial attacks

In the case of single-agent DRL scenarios, the only way the attacker has to fool the victim agent consists of adding small perturbations to the observations it receives as input. In fact, one of the most interesting features that made DRL agents so popular is that they are only fed with some observations of the environment in order to learn or exploit their policy. Commonly, these observations only consist of raw or pre-processed images composed of pixels, thus vulnerable to adversarial attacks targeting images.

Following, we are going to introduce some of the most significant adversarial attacks under single-agent scenario. When it is required to make the victim agent choosing or avoiding a particular action, it can be used any method for adversarial attacks on images to craft adversarial observations, and sometimes, this procedure will be omitted without loss of generality.

4.1.2 Uniform attack

The first attack proposed to degrade the performance of DRL agents is the Uniform Attack[30], it falls into the family of untargeted attacks and consists of slightly perturbing all images the agent observes so to cause it choosing sub-optimal actions. Perturbations can be crafted with any existing method to attack image classification models under untargeted settings. This simple attack can be regarded as a direct extension of the adversarial attack on a CNN-based classification system since the adversarial example at each time step is computed independently of the adversarial examples at other time steps. However, it shows good results both in terms of transferability against algorithms and policies, in particular for large values of perturbations ϵ. However, despite being particularly effective against DQN, it doesn't scale well against more robust algorithms such as A2C and PPO.

4.1.3 Strategically-timed attack

Instead of attacking at every time step in an episode, Strategically-Timed attack[29] selects a subset of time steps to attack the agent. Given the difficulty of the problem, the authors decided to subdivide it into two separate tasks: *when to attack* and *how to attack*. To solve the first task, it can be defined a function c that computes the preference of the agent in taking the most preferred action over the least preferred action at the current state s_t. Hence, the c function can be defined as

$$c(s_t) = max_{a_t}\pi(s_t, a_t) - min_{a_t}\pi(s_t, a_t). \tag{4-1}$$

Then, the adversary attacks the victim agent at time step t when the relative action preference function c has a value greater than a threshold parameter β. The intuition behind this is that when an agent strongly prefers a specific action, it means that it is critical to perform that action, otherwise the accumulated reward will be reduced. To solve the second task, it can be used any image adversarial attack method to change the preferred action of the agent from the originally most preferred one to the originally least preferred one. Finally, the authors of this attack found that, on average, the strategically-timed attack can reach the same effect of the uniform attack[30] by attacking 25% of the time steps in an episode.

4.1.4 Enchanting attack

The goal of the Enchanting attack[29] is to lead the DRL agent from current state s_t to a specified target state s_g after H steps. Also this attack is divided into two tasks: the first one consists of planning a sequence of actions for reaching the target state s_g from the current state s_t. In the second one, it is crafted an adversarial example to lure the victim to take the first action of the planned action sequence using any image adversarial attack method. In particular, the first task is accomplished by training a video prediction model to predict a future state given a sequence of actions using a generative model. Since the goal of the enchanting attack is to reach the target state s_g, the success of the attack is based on the distance between s_g and s_{t+H} predicted by the model. Regarding the action selection, are computed a sequence of actions using a sampling-based cross-entropy method, and is selected the sequence leading to a final state s_{t+H} which is the closest to s_g. Based on evaluations on some Atari games, this method reaches an accuracy of about 70% with a drop of performance on environments with a high level of stochasticity.

4.1.5 Critical point attack

In Critical Point Attack[32] the adversary builds a domain-specific model to predict the states of the next few steps, as well as the damage consequences of all possible attack strategies. As for many other attacks, also this method is performed in many stages. Firstly, the adversary uses observations of the target environment to train a prediction model P such that it takes a state-action pair as input and outputs a deterministic prediction of the next state. Then, it defines different strategies as sequences of N actions starting at step t. In the third step, the adversary assesses the potential damage of all possible strategies from state t to $t+M$ with $M >= N$ and the last $M-N$ steps following the normal policy. Next, the adversary calculates the value of the last state $T(s_{t+M})$ using the function T which is a domain-specific divergence function defined as the least favorable state to reach. After computing $T(s'_{t+M})$ for each attack strategy and the normal policy, the Danger Awareness Metric (DAM) between the two states, one reached because of a malicious policy and the other one following the normal policy is defined as

$$DAM = |T(s'_{t+M}) - T(s_{t+M})|. \tag{4-2}$$

If DAM is larger than a threshold δ, the adversary will conduct the attacks in the next N steps following the chosen strategy. Otherwise, it will do nothing and repeat this assessment process from the next step $t+1$. Overall, this method requires about 1 to 4 steps to significantly degrade the performance of the victim policy, which is a significant improvement over[30] and[29].

4.1.6 Critical strategy attack

This attack doesn't formally exist in literature but it has been designed specifically to be included in this work. It borrows some concepts from Critical Point Attack with the difference that the divergence function T is not computed only over the final state s_{t+M} but over the whole sequence of states encountered from step t to step $t+M$. To make this attack more environment-independent, we can define the divergence function of a strategy to be equal to the sum of the corresponding reward of each state encountered along the trajectory defined by a strategy:

$$T(s'_{t+M}) = \sum_{i=0}^{M} R(s'_{t+i}, a'_{t+i}). \tag{4-3}$$

Hence, the second difference with[32] is that in this method lower values of T show the presence of more critical states. Here the word *Strategy* is due to the fact that this method keeps track of the whole sequence of states rather than focusing only on the last one. The final algorithm is shown in algorithm 4.1 and can be easily customized replacing T with other divergence functions. For both Critical Point Attack and Critical Strategy Attack, the number of planned strategies has a significant impact on the algorithm's speed. A simple way to generate them is to enumerate all possible combinations of $a'_t, ..., a'_{t+N-1}$ but it is easy to realize that this solution is unfeasible for large values of N. Hence, a possible solution consists of choosing a fraction of N such as N/S, list all combinations of N/S actions, and then repeat each action S times. The intuition behind this method is that a policy, at least for the Atari environments, usually doesn't need to execute many different actions to be adversarial, thus we can discard those strategies that change actions too frequently. For example, in the environments of Pong where the goal is to let the ball bounce against the bar, not moving the bar at all is already a discrete adversarial strategy.

Algorithm 4.1 Critical Strategy Attack performed at step t

```
Input: $M, N, P, T, s_t$
Output: optimal attack strategy $a'_t, ..., a'_{t+N-1}$
/* Calculate the baseline $T(s_{t+M})$ */
for $i = 0$ to $M - 1$ do
    $a_{t+i} \leftarrow AgentAct(s_{t+i})$;
    $s_{t+i+1} \leftarrow P(s_{t+i}, a_{t+i})$;
    $s_{t+i} \leftarrow s_{t+i+1}$;
end
Plan for all possible N-step attack strategies as a set A;
$s'_t \leftarrow s_t$;
for each attack strategy $a'_t, ..., a'_{t+N-1}$  $A$ do
    $T_{s'+M} \leftarrow 0$;
    for $i = 0$ to $M - 1$ do
        if $i \leqslant N$ then
            $a_{t+i} \leftarrow a'_{t+i}$;
        else
            $a_{t+i} \leftarrow AgentAct(s'_{t+i})$;
        end
        $s'_{t+i+1} \leftarrow P(s'_{t+i}, a_{t+i})$;
        $s'_{t+i} \leftarrow s'_{t+i+1}$;
        $T_{s't+M} \leftarrow R(s'_{t+i}, a_{t+i}) + T_{s't+M}$;
    end
    if $|T(s'_{t+M}) - T(s_{t+M})| > \Delta$ then
        return $a'_t, ..., a'_{t+N-1}$;
    end
end
return $None$;
```

4.1.7 Adversarial policy

Adversarial Policy attack consists of solving a two-players Markov game M, where the opponent α, controlled by the adversary, has to compete against the victim agent v in order to reduce its earned reward. Given that the victim policy π_v is held fixed, the game reduces to a single-player MDP $M = (S, A_\alpha, T_\alpha, R_\alpha)$ the attacker has to solve. Thus, the goal of the attacker is to find an effective adversarial policy π_α by maximizing the sum of its discounted rewards. To achieve this goal, the adversary can simply use any suitable algorithm to train a deep neural network where the adversary's reward R_α is set as the negative of the agent's reward R_v. The output is an adversarial policy that minimizes the reward on the target environment. Finally, it can be used any targeted attack on observations to let the target agent follow the adversarial policy and set a threshold parameter β like in[29] to control the attack frequency.

4.2 Defense methods against single-agent adversarial attacks

Given that attacks under this setting partially rely on attacks on images, consequently the same defenses used to protect image classification models bring the same benefits also when protecting DRL agents. A wide range of defence techniques is thus available such as defensive distillation[33], input transformation, certified defenses[34] [35] [36], or simply performing adversarial training[15]. As proved by[30], also the robustness of the algorithm used to train the policy can improve the general robustness against adversarial examples.

4.2.1 DQN adversarial training

As mentioned before, adversarial training consists of augmenting the dataset with adversarial examples and retrain the model on it so to improve its robustness. In the context of DQN, the dataset is nothing else than the replay buffer from which the algorithm will sample past experiences. Thus, to train DQN with adversarial training it is used any image attack method to attack each observation and store the corresponding generated adversarial observation in the buffer. As for the other values, it is stored the original action corresponding to the non-adversarial observation along with the non-adversarial next observation and the normal reward so to store the tuple $(s_t^{adv}, a_t, r(a_t, s_t), s_{t+1})$. Since the goal is simply to let the network learn the right Q-value given an adversarial observation s_t^{adv}, the corresponding observation s^{t+1} doesn't need to be adversarial. Consequently, the loss function will be formulated according to the following formula:

$$L_t(\theta_t) = (r_{t+1} + \gamma_{t+1} \max_{a'} Q_{\theta^-}(s_{t+1}, a') - Q_{\theta_t}(s_t^{adv}, a_t))^2. \tag{4-4}$$

In the implementation used in this work, adversarial training is applied on top of an already trained policy and perturbations are generated with untargeted attacks. The idea behind this mechanism is that the target network, which takes as input clear observations might supervise the online network so as to adjust its Q-values even in presence of adversarial observations.

4.2.2 A2C-PPO adversarial training

A little different is the way adversarial training works for the two on-policy algorithms. Given that the two algorithms learn in an on-policy way, they collect a few experiences during the normal exploration phase and immediately make use of them to learn and improve their policy. However, those experiences will be forgotten right after. To perform adversarial training, we first let the neural network take as input the clean observation s_t and output its corresponding action a_t. Following, this action is used by any image adversarial attack to craft an adversarial observation s_t^{adv}. The network is then fed with the adversarial observation so to produce an adversarial action a_t^{adv} and value v_t^{adv} which are used to estimate the advantage. The following formula shows the way to compute the gradients under adversarial training in the case of A2C and it can be easily generalized to other on-policy actor-critic algorithms such as PPO:

$$\boldsymbol{\nabla}_\theta = \log \pi(a_t^{adv} | s_t^{adv}; \theta_\pi) A(a_t^{adv}, s_t^{adv}; \theta_v), \tag{4-5}$$

$$A(a_t^{adv}, s_t^{adv}; \theta_v) = R_t - V(s_t^{adv}; \theta_v). \tag{4-6}$$

However, in this way the network is being fed only with adversarial observations which may significantly degrade its performance on clean observations. To overcome this drawback, the network is trained with adversarial observations with probability p and clean ones with probability $1 - p$ with p for example equal to 0.5.

4.3 Multi-agent adversarial attacks

So far we have seen that an adversary can produce adversarial examples by perturbing some pixels on the input observations. However, in real scenarios, an attacker is not usually able to directly modify another agent's observations[31]. Nevertheless, under certain circumstances, an attacker can still control other agents interacting in the same environment of the victim agent so to create natural observation that results adversarial. To make it more clear, let's suppose that an agent is driving an autonomous car and its policy is based on observations of the surrounding environment. The way the autonomous system is designed, makes it impossible for an attacker to add adversarial noise directly to the frames the camera sends to the system. However, an agent could control another car or the behavior of a pedestrian so to create an adversarial observation and deceive the victim. To achieve this goal, an adversarial agent is usually trained with an adversarial policy with the aim to fool the victim agent in a targeted or untargeted way. Following, we are going to review in detail an example of a method attacking under multi-agents settings.

4.3.1 Multi-agent adversarial policy

We have already introduced adversarial policy attacks where their goal was to lower the rewards of the victim agent. However, since in this scenario we have two or more agents competing with each other, the goal of the attacker becomes to make the victim lose[31]. Hence, given an opponent α competing against the victim agent v with a fixed policy π_v. The goal of the attacker is to find an effective adversarial policy π_α by maximizing the sum of its discounted rewards:

$$\sum_{t=0}^{\infty} \gamma^t R_\alpha(s_t, a_{\alpha_t}, a_{v_t}, s_{t+1}), \tag{4-7}$$

where $s_{t+1} \sim T_\alpha(s_t, a_{\alpha_t}, a_{v_t})$, and $a_{\alpha_t} \sim \pi_\alpha(\cdot|s_t)$. The rewards are sparse and are given at the end of the episode depending on its outcome. More specifically, the adversary gets a positive reward when it wins the game and a negative reward when it loses or ties. Finally, the adversarial policy can be maximized with any DRL algorithm. Since the target policy is static, it can be viewed as part of the environment and the reward of the attacker to be maximized reduces to:

$$\sum_{t=0}^{\infty} \gamma^t R_\alpha(s_t, a_{\alpha_t}, s_{t+1}), \tag{4-8}$$

When evaluating this method on 1vs1 robotics simulator environments, the adversarial policies reliably win against the victims with an 86% rate, but in contrast, they generate seemingly random and uncoordinated behavior. Moreover, the authors of this method also realized that the greater the dimensionality of the component of the observation space under control of the adversary, the more vulnerable the victim is to adversarial attacks.

4.3.2 Defense methods against multi-agent adversarial attacks

One intuitive strategy to protect DRL agents against adversarial policies is to fine-tune the victim against the adversary. In practice, this method works, but only protects against that particular adversary. In fact, by repeating the same attack[31] against the re-trained agent, the adversary can learn a new policy to which the victim is vulnerable. However, repeating this procedure multiple times might provide protection against a wide range of adversaries. Another more interesting and effective defense mechanism consists of making the victim masked. Masking a victim agent means that the observation of the adversary's position is set to a static value corresponding to a typical initial position. During evaluation, masked victims perform slightly worse than a normal victim when playing against normal opponents, but they reveal almost invincible when playing against adversarial agents. This result implies that non-transitive relationships may exist between policies even in games that apparently seem to be transitive. In the context of policies, transitivity means that higher-ranked policies beat lower-ranked policies[31].

4.4 Applying FGSM to policies

FGSM is a white-box attack, hence it requires calculating the gradient of the cost function $\nabla_x L(x, y)$ with respect to the input observation x and the correct label y. The output of a reinforcement learning policy is a probability distribution over all possible actions, thus we assume that the action a with the maximum weight in y is the optimal action to choose. We are sure that a is optimal since we are assuming that the policy has been well trained on its task. Thus, $L(x, a)$ is the cross-entropy loss to be maximized to craft an adversarial observation under untargeted settings. It corresponds to the direction the gradient has to take so to decrease the value of the optimal action a. Similarly, under targeted settings, the goal of the adversary is to increase the value of a specific action a^{adv} defined by a certain malicious policy. It can be achieved by minimizing the loss $L(x, a^{adv})$ with respect to the desired action. The two algorithms A2C and PPO train stochastic policies, that is, it is immediate to get the probability of each action or their logits. However, DQN produces a deterministic policy since it always selects the action that maximizes the computed Q-value. This mechanism might cause some problems because it results in a gradient $\nabla_x L(x, y)$ of zero for almost all observations x[8]. Thus, when calculating $L(x, y)$ for policies trained with DQN, y is defined as the logits of computed Q-values to create adversarial observations.

CHAPTER 5 STUDYING POLICY ATTACKS TRANSFERABILITY

This section reports the results of the first of the three experiments included in this thesis. It evaluates different policy attacks focusing on their transferability and drawing some insights about their performance under different scenarios.

5.1 Studying policy attacks transferability

The first experiment included in this work consists of studying the transferability over different policies and algorithms of some policy attacks so as to have an idea of their robustness and gain a better understanding of the effectiveness and proprieties of the employed attacks. In particular, we are interested in understanding how vulnerable neural network policies are against adversarial policies and how their effectiveness is affected by the way those policies are trained and the amount of knowledge is available to the attacker with respect to the target policies. Their robustness is measured by plotting the *reward vs attack frequency curves* of different policy attacks under a fixed threat model for each curve with the aim to measure the reduction of the agent's reward as the attack frequency increases. The experiments have been conducted on three popular reinforcement learning algorithms: DQN, A2C, and PPO with policies trained on an environment belonging to the Atari games category, namely Pong. Besides, the five policy attack methods that have been compared include:

- Uniform attack;
- Strategically-timed attack;
- Critical point attack;
- Critical strategy attack;
- Adversarial policy.

5.1.1 Environments

All the environments where the policies have been trained consist of common Atari games included in the Gym library[37]. Hence, the observations are RGB images of the screen, which are arrays of shape (210, 160, 3). Moreover, each action is repeatedly performed for a duration of k frames, where k is uniformly sampled from $[2, 3, 4]$. To represent temporal information, 4 observations are stacked together, converted to grey-scale, and scaled down to have a shape of (84, 84, 1) pixels.

5.1.1.1 Pong

In this environment, both the agent and the opponent control a paddle which can be only moved vertically across the screen to hit a ball back and forth. Each player scores one point when the other player fails to return the ball to the other and their goal is to reach eleven points before the opponent. The player on the right is controlled by the agent and the one on the left is controlled by a simple hard-coded AI.

5.1.2 Experiments settings

5.1.2.1 Algorithms hyper-parameters

To implement and train the reinforcement learning policies with their corresponding algorithms, it has been used the framework Tianshou[38] based on Pytorch. Starting to describe algorithms specifications, the DQN variant adopted in this work takes advantage of a prioritized experience replay[39] of size 100k to sample more frequently the most critical observations, that is those observations where the discrepancy between the Q-values of the online and target network is more significant, so to lead to faster training. It has also been implemented Double Q-network to decouple the selection of the action from the evaluation and reduce overestimation, and multi-step learning with an estimation of 3 steps so to take advantage of the bootstrapped sequences of transitions to improve the target value accuracy. Training has been conducted for a total of 10M frames with ϵ linearly decaying from 1 to 0.05 during the first 1M frames to encourage exploration. The target network has been updated with the weights of the online network every 5000 frames. Regarding A2C and PPO algorithms, the policy has been trained for 10M frames and updated with trajectories of 128 steps and a batch size of 32. The advantage has been computed using GAE with $lambda$ equal to 0.95. The value loss and the entropy loss have been given a weight of 0.5 and 0.01 respectively. For PPO the clipping ratio is 0.2. Other common parameters used in all the three algorithms are a learning rate of 0.0001 and $gamma$ of 0.99 used to define the importance of long-term rewards. To speed up training, all policies have been trained on 16 parallel environments. In order to measure attack policy transferability, for each environment have been trained two policies using different seeds but with their performance remaining similar. The average reward of the surrogate policy of each environment doesn't differ more than 10% of the mean reward of the corresponding primary policy. Table 5.1 reports the average score over 100 episodes for the three different algorithms.

Table 5.1 Average score over 100 episodes for policies trained with DQN, A2C, and PPO. It can be clearly seen that PPO, being a more sophisticated algorithm than the other two also performs much better. Pong is easy enough that most algorithms can successfully train a policy to defeat the opponent.

Environment	DQN	A2C	PPO
Pong	20	20	21

5.1.2.2 Image adversarial attacks settings

All the evaluated policy attacks exploit FGSM to craft perturbations to generate adversarial observations. The reasons we have chosen to adopt a simple method like FGSM over more sophisticated white-box attacks are the following: first of all, FGSM is a one-shot method that can craft adversarial perturbation in just one iteration, thus it is many times faster than iterative ones such as PGD. Given the number of times the attack is required to be executed to hamper thousands of policy trajectories, evaluating an iterative method could be not worth the time in correspondence of a moderate gain in attacks success rate. Secondly, the experiments presented in section 7 empirically show that, when attacking Atari observations, FGSM doesn't perform worse than iterative methods under the same distance norm and perturbation strength. All perturbations have been computed under l_∞ norm with a distance of $\epsilon = 0.05$.

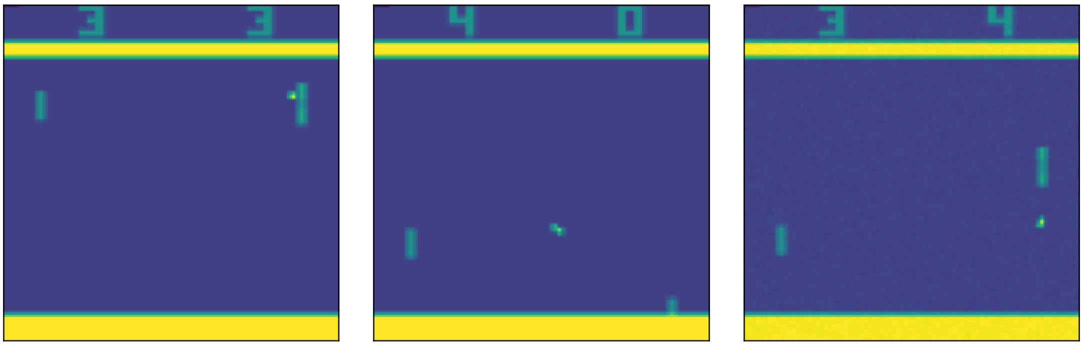
5.1.2.3 Policy adversarial attacks settings

Next, it is explained how attack-frequencies relative to the evaluated policy attacks have been defined in order to generate the corresponding curves. Among the included attacks, the simplest method is the uniform attack where it can be directly tuned a predefined frequency hyper-parameter, thus changing the probability to attack each observation. Strategically-timed attack and adversarial policy exploit the parameter $beta$ to fine-tune their frequency. Higher values of $beta$ mean that the difference between the most and the least likely actions should be higher in order to attack the corresponding observation, thus meaning it has been encountered a more critical state. When $beta = 0$ it results that every frame is attacked and a value of $beta$ close to 1 corresponds to very sparse but effective attacks. Under this criteria, critical states correspond to those states where choosing a particular action has a significant influence on the reward over the future states. Since the original method doesn't provide a way to define a certain attack frequency, the hyper-parameter $beta$ has also been added to adversarial policy attack so to tune its attack rate. Finally, the attack frequency of the critical point and critical strategy attack is determined by the two parameters n and m which affect the length of the adversarial strategies. When $n = m$, both methods perform attacks over all frames and their frequency reduces as the difference $m - n$ grows.

Besides critical point attack, all the other policy attacks can be generally used to attack different environments just by fine-tuning their hyper-parameters without requiring any extra configuration. However, critical point attack is based on a divergence function T to estimate how critical is any observation. In this context, critical observations are those observations where the likelihood to fail the episode is higher. Adapting this concept to the evaluated environments, the divergence function is correlated to the probability that the ball would pass over the paddle, giving one point to the opponent in the case of Pong. This probability is estimated as the distance between the center of the paddle and the ball along the y-axis for Pong. The concept behind this function is that the further the distance between the ball and the paddle along the corresponding axis, the more difficult is for the normal policy to catch the ball. A more sophisticated divergence function would have been designed by taking into consideration also the direction of the ball. Unfortunately, it would have been more complex to implement since it would involve a sequence of stacked frames and precise algorithms to estimate the direction of the ball by comparing its shape and position over different frames. In practice, a model could take as input sequences of stacked observations but in this work we consider only the last one to compute the value of the divergence function. Moreover, designing an optimal divergence function is out of the scope of this thesis project. An example of the behavior of the divergence function for the game of Pong is showed in figure 5.1.

5.2 Results

The next three sections show the *reward vs attack frequency curves* of all the evaluated policy attacks under a fixed threat model so to measure their effectiveness and their transferability against policies and algorithms. When analyzing the obtained results, sometimes we may refer to the curve of the policy attacked without transferability as base-



(a) Safe state   (b) Uncertain state   (c) Dangerous state

Figure 5.1   Example of the behavior of the divergence function T for the game of Pong: the observation on the left (a) has a very low level of T since the probability that the agent fails is 0. The observation in the middle (b) presents a discrete T since considering the ball moving simply toward the right of the screen along the x-axis the agent can still make the ball bounce back. The third observation (c) has a very high value of T since there is no way the agent can save the ball.

line policy. In addition, we may sometimes intend as *attacking with a surrogate policy* or *attacking with a surrogate algorithm* as crafting adversarial perturbations exploiting a surrogate policy/algorithm to attack the target policy. Let's also remind that the purpose of these experiments is not only to evaluate the efficacy of different attack methods but to study their transferability proprieties. Hence, the relative value of each curve with respect to the other ones is more important than their individual absolute value.

5.2.1 Attacking Pong-DQN

The curves relative to this experiment are depicted in figure 5.2. We observe that in the case of uniform attack, the average rewards decrease from 20 to -20 almost linearly as the attack frequency increases from 0 to 0.5. For higher frequencies we didn't observe a larger drop in reward, thus meaning that the victim policy already performs close to the worst possible case. On the evaluated environment, uniform attack also shows good transferability both in terms of policy and algorithm. In particular, crafting adversarial observations with a similar policy yields similar results to attacking the baseline policy, obtaining almost identical performance for lower attack frequencies. Perturbations crafted on the other two algorithms, namely A2C and PPO, still successfully fool the victim agent with their curves following the trend of the baseline but maintaining an average reward difference of about 5 points.

Analyzing the curves relative to the strategically-timed attack, we can observe that the attack frequency is several grades of magnitude lower than uniform attack and is the lowest among all the examined policy attacks. In the frequency interval taken into consideration, which varies from 0.002 to 0.012, the corresponding average reward drops from a little more than 10 to less than -15 for the baseline policy which follows an almost linear decay. However, considering that this time adversarial observations are crafted in a targeted way, transferability seems to be less effective than it was for uniform attack. While attacking with policy transferability still yields results relatively closer to the baseline, perturbations crafted on the two on-policy algorithms often fail to fool the agent, thus requiring more attacks to provide meaningful results.

Critical point attack has attack frequencies varying from 0.01 to 0.09, interval in which the reward falls from 20 to -15. As opposed to the previous two attacks, perturbations crafted under the surrogate policy don't work much better than the perturbations crafted on the policy trained with the other two on-policy attacks. Nevertheless, DQN policy transferability still seems to work a little better than A2C and PPO algorithm transferability as the attack frequency increases. The curves of critical point and critical strategy attack are very similar both in terms of rewards and frequencies. However, both methods can't reduce the agent's reward to less than -15 in the given frequency interval.

Finally, adversarial policy attack seems to perform the worst. The curves representing the two algorithms' transferability are very far from the other two curves and struggle to lower the reward to less than 10 points. The curve corresponding to policy transferability is also much weaker than the baseline but they follow the same trend and it is able to lower the average reward to less than 0 points. Overall, we can conclude saying that DQN is not a very robust algorithm since perturbations crafted with surrogate policies still fool the victim model with performance similar to directly attacking the victim policy. In fact, the closer are the curves to the baseline, the less robust is the target algorithm.

5.2.2 Attacking Pong-A2C

The curves relative to the A2C policy are shown in figure 5.3. Beginning analyzing uniform attack, we can see that the curve of the baseline is similar to its corresponding DQN version but the performance of the surrogate policies drop considerably. Attacks with the surrogate similar policy still follow the trend of the baseline with a rapid decrease of the agent's reward for frequencies increasing from 0 to 0.25 and a more amortized decline until a frequency of 0.5. Attacks exploiting a surrogate algorithm never inflict too much damage in the considered interval with PPO producing the least effective perturbations. Adversarial observations crafted with DQN struggle to lower the average reward of the target policy to below 0 in the given frequency interval while PPO slowly reduces the rewards from 20 to 10. Strategically-timed attack once again benefits of a very low attack frequency. However, the robustness of A2C deeply impacts the effectiveness of the attacks with the surrogate policies. The baseline reduces the rewards from 20 to -15 attacking less than 2% of the observed frames. Same as in uniform attack, adversarial examples generated attacking PPO need a higher attack rate to reduce the rewards by 10 points and DQN performs even worse. Also the baselines of the curves of critical point and critical strategy attack are similar to the baseline of their corresponding curves obtained in the previous section when attacking the DQN agent. As it was for the other methods, the surrogate policies perform much worse, with the attacks based on algorithm transferability suffering the most. Also in the case of adversarial policy, the adversary doesn't perform very well and its baseline resembles the one of uniform attack. In fact, it needs to increase the attack frequency to over 0.3 to achieve successful results. Attacks with DQN and PPO present very similar curves, both decreasing steadily to reduce the average reward to -5. We can then draw out the conclusion that A2C algorithm is more robust than DQN, in particular, it seems to resist very well against attacks with surrogate policies trained with different algorithms. In fact, all attacks show poor algorithm transferability and discrete policy transferability.

5.2.3 Attacking Pong-PPO

The last set of charts depict the curves relative to the policy trained with PPO and it is shown in figure 5.4. Starting from the uniform attack, nothing new appears here with respect to the charts of the uniform attack relative to the other two algorithms. All curves follow the trend of the baseline but the adversarial examples generated with transferability are this time less effective. Attacks crafted with DQN are the ones performing the worst. Strategically-timed attack seems not to be able to attack as easily as it did against DQN. In fact, it requires a relatively higher attack frequency to achieve similar performance. Moreover, same as in the case of A2C, attacks with surrogate models trained with different algorithms, have to put more effort to decrease the average reward of the victim agent. Conversely, policy transferability still works discretely well. Critical point and critical strategy attack demonstrate good transferability and similar results, thus meaning that in the environment of Pong a hand-crafted divergence function is unnecessary to elaborate good attack strategies. Surprisingly, adversarial policy attack seems to work better than it did when attacking A2C. It can almost linearly lower the agent's average reward from 20 to 0 attacking no more than 20% of the frames. and it also owns a better transferability property than it had against A2C and DQN. Overall, PPO seems to be a little more robust than DQN but less robust than A2C. In terms of transferability all attacks, except for strategically-timed attack, are able to craft effective adversarial observations using their respective surrogate policies. Strategically-timed attack still works very well under policy transferability but fails when attacking with different algorithms.

5.3 Findings

This section analyses the policy attack methods that have been evaluated in this experiment and draws some conclusions about their usage, effectiveness, and drawbacks. It also points out some future research directions to improve those attacks and makes insightful comparisons among similar methods.

5.3.1 Uniform attack vs strategically-timed attack

Not surprisingly, uniform attack is the most simple but also the weakest attack that has been examined in the present evaluation. However, it is very fast and is compatible with untargeted image attacks, thus allowing the exploration of other image attacks such as Deepfool that only work under untargeted settings[40]. Interestingly, strategically-timed attack achieves the best performance under the lowest frequencies. The reason behind its excellent performance is that this method has specifically been designed to work under low attack frequencies attacking only in correspondence with the most critical observations. Being its attacks very sparse, it takes more frames to make the victim agent fail. Hence, resulting in a very low attack frequency. The intelligent way strategically-timed attack decides whether to attack an observation or not could be also borrowed to other methods such as uniform attack to lower their frequency while limiting the drop in performance. In this work, this technique has been applied also to the adversarial policy attack to vary its frequency. One possible drawback of this technique is that it may miss some observations that apparently don't seem critical but that attacking them may lead to worse states in the future steps, a drawback that has more important consequences in the case of the adversarial policy attack.

5.3.2 Critical point vs critical strategy attack

Critical point attack and critical strategy attack are constrained by the hyper-parameter n which, forcing the two methods to attack every observation over sequences of n frames, might result in some unnecessary attacks. Overall, the attack frequency is largely influenced by the m parameter which defines the number of steps taken without attacking the policy of the agent. Higher values of m comport a decrease of the attack frequency but also lead to a potential risk of missing some critical states which it would have been worth attacking. In fact, the last $m - n$ observations of an observed trajectory are completely ignored by both attacks. Different frequencies are thus generated varying n and m. Going more deeply into understanding their behaviors, critical point attack relies on a divergence function to estimate how critical is the last state of a trajectory defined by a strategy. However, both the strategy that defines some sequences of malicious actions and the divergence function of critical point attack are hand-crafted, which means that they require some domain knowledge to be designed by a human. While a strategy that works for a certain environment may still work quite well in similar environments, a specific divergence function will not work in other environments, even for similar ones. For instance, a strategy that only repeats the same action is considered a malicious strategy for almost all the Atari environments, but clearly, this transferability property may not be true for some other environments. In contrast, critical strategy attack depends on the cumulative reward over a trajectory to determine the maliciousness of a strategy: the lower the reward the more malicious a strategy is. However, what makes reinforcement learning challenging is exactly the sparsity of the rewards. This means that the value of most trajectories will be zero. On one hand, this is a positive property since adversarial strategies will always aim to prevent the agent to increase its score, eventually leading it to fail the episode or lose one life. On the other hand, with many strategies preventing the agent to get any reward along, it's not easy to determine which one may continue to lead the agent to more malicious states in the future steps. For both attacks, the special case where $m = n$ means attacking every frame and both attacks are more effective as n increases and the difference $|m - n|$ reduces. Furthermore, the hyper-parameter n has influence on the number of adversarial strategies to be tested. Supposing a to be the number of actions of an environment and the strategies to be all possible combinations of n actions, $a^n + a * (m - n)$ strategies will be evaluated with n adding an exponential computational cost since more strategies will be evaluated. Thus, the way to generate adversarial strategies should be carefully crafted.

Overall, critical point attack performs slightly better than critical strategy but it depends on a divergent function to be computed on the input observations which sometimes could be a challenging task.

5.3.3 Adversarial policy and future search directions

Given the results presented in this section, apparently adversarial policy attack is the one performing the worst. The reason behind this is that this method doesn't have a real mechanism to control and vary its frequency. Hence, it borrows the technique used by strategically-timed attack. The difference between the two is that while the strategically-timed attack executed the least likely action, adversarial policy attack crafts perturbations such that the victim agent executes a specific action predicted by the network implementing the adversarial policy. In fact, in adversarial policy attack, all actions are equally important since the goal here is to lead the agent toward a certain state where failure is inevitable. However, missing one or a few steps because wrongly considered as not critical may lead to other less malicious policies. Generally, this method as well as critical point and critical strategy, focuses on making the agent fail faster rather than slowly sabotage it as uniform attack and strategically-timed attack do. However, blindly attacking each frame is still not an elegant solution to make these attacks effective. Attacking more observations not only requires more computational power but also increases the possibility of the attacker being detected. If hand-crafting a mechanism to determine when to attack is not enough for such more complex methods, then machine learning is once again the answer. Future research directions might then focus also on *when to attack* rather than *how to attack*, thus learning to predict critical observations along the corresponding malicious action and even learning to limit the number of attacks so to control the frequency.[32] pioneers this work with antagonist attack.

5.3.4 Attack feasibility in common scenarios

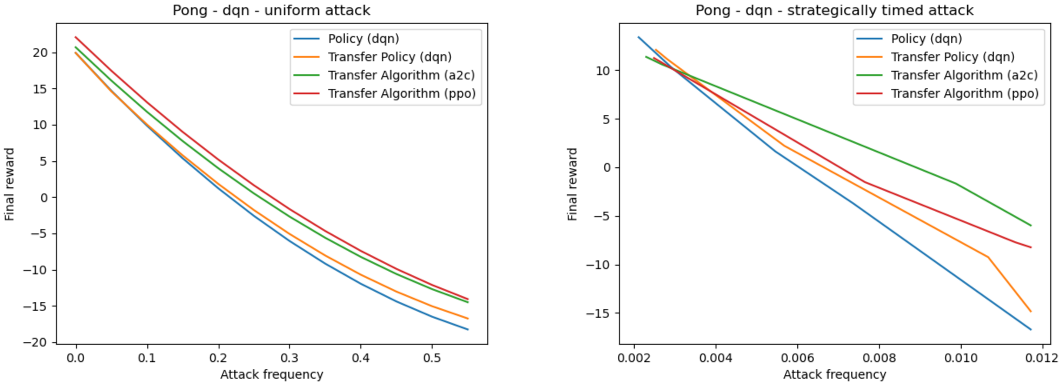
Another point that is important to mention is that strategically-time attack takes advantage of the logits or probability of the predicted actions in order to attack the victim agent. However, this knowledge is not always available under common threat models. Thus, although the above-mentioned attack seems to work extremely well under experiment settings, in practice it may result ineffective in real-world scenarios. In contrast, to use uniform attack the adversary only needs to know the action predicted by the model representing the policy so to cause it to take another random action. This knowledge is usually more commonly available than the actions probabilities but it is still a limiting constraint to make the attack work. However, this contrast could be relaxed by assuming the target agent to have taken a certain action, even a random action, and consequently use it to craft the corresponding perturbations. Critical point and critical strategy both need a simulator of the environment to assert the value of their strategies. In the best-case scenario, a copy of the state of the environment could be used so as to exactly know the outcome of each strategy. In fact, both methods could take actions in the environment according to the actions defined in a strategy, compute the value of the last observation or the value of the trajectory and then revert it to the saved state so to evaluate the next strategy. If retrieving the state of the environment is not possible, then it should be engineered other ways to make a simulation of the environment that is as close as possible to the original one. The authors of critical point attack proposed to train a neural network that given an observation and an action it predicts the next observation. An interesting solution is the one introduced in[41] which trains a variant of GAN to learn a simulator by simply watching an agent interact with the environment. In those circumstances where, for any reason, a simulation of the environment is not available, attack strategies can still be computed based on a simulation of similar environments or simply designing some strategies that intuitively may be malicious for the target environment regardless of its current state. The only requirement adversarial policy attack needs in order to attack its target is to have access to the input observations. Therefore, this method is more flexible since it can work under different threat models. However, it still needs to be trained in the same environment of the victim policy before it can be used to attack it. Let's remind that all the policy attacks evaluated in this thesis work only under the condition that input observations can be modified. Table 5.2 summarizes the requirements of each policy attack to work properly.

Table 5.2 Requirements needed for each evaluated policy attack method to work properly.

Attack method	Observations	Environment	Pred action	Actions probs
Uniform Attack	V	-	V	-
Strategically-Timed Attack	V	-	-	V
Critical Point Attack	V	V	-	-
Critical Strategy Attack	V	V	-	-
Adversarial Policy	V	Only for training	-	-

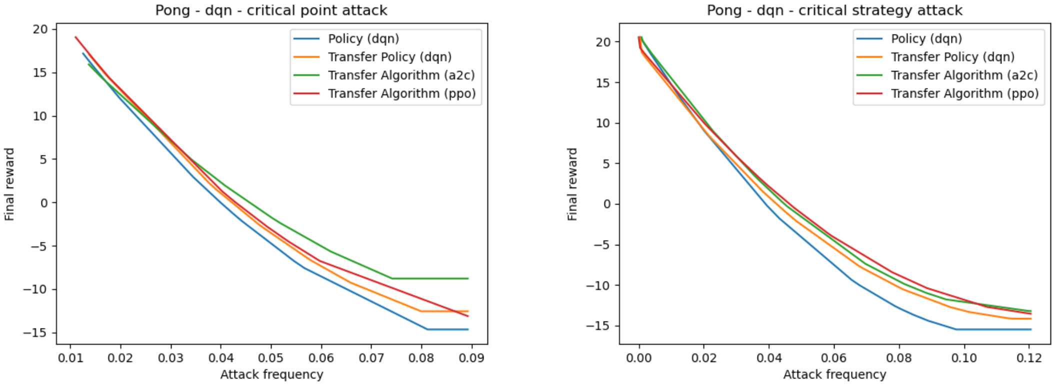
5.3.5 Online vs offline attacks

Uniform attack, strategically-timed attack, and adversarial policy attack are relatively fast attacks. In fact, excluding the time the image attack method takes to generate the adversarial perturbations, the first two methods only involve a few linear computations while the third one requires one forward pass of its network. Hence, those three attacks, combined with a fast image attack method, are suitable to attack in an online way following the stream of states observed by the target agent. Conversely, critical point and critical strategy attack need to explore and assert the damage of different strategies before start attacking and this operation should be repeated every $n+m$ frames. This bottleneck makes these two attacks too slow to be executed in an online way but more useful to attack offline when time is not a constraint. Nevertheless, in real life scenario is more common to have to attack online rather than offline.



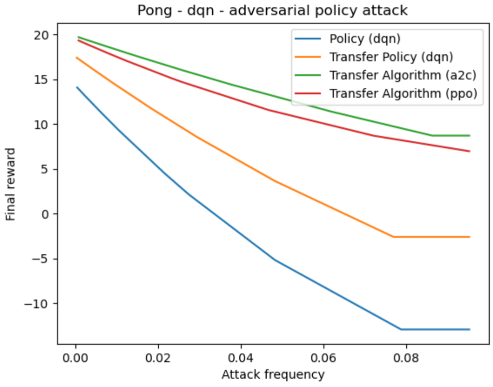
(a) Uniform attack

(b) Strategically-timed attack



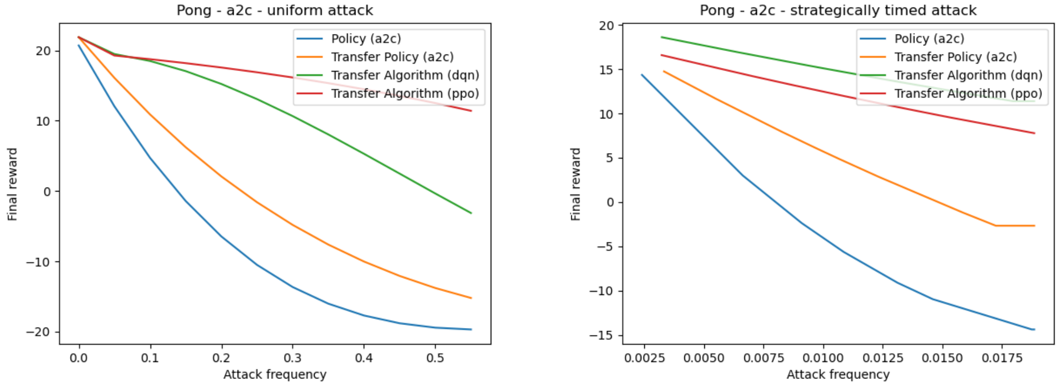
(c) Critical point attack

(d) Critical strategy attack



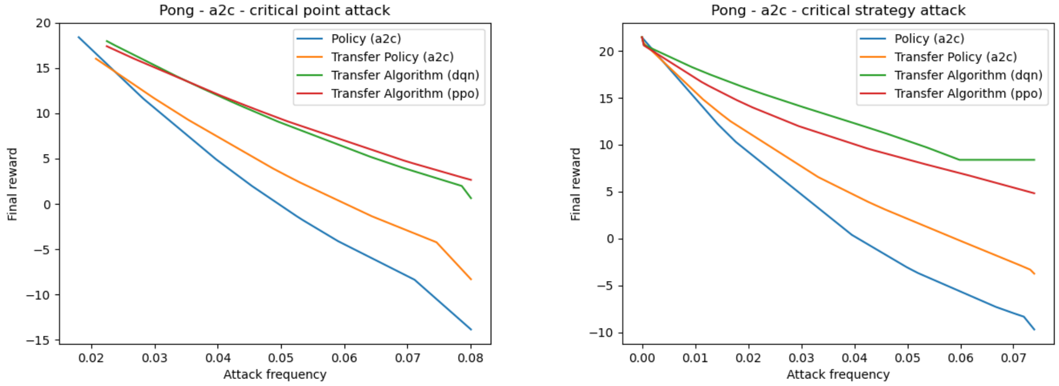
(e) Adversarial policy attack

Figure 5.2 The *reward vs. attack frequency* curves of 5 policy attacks with FGSM under the l_∞ norm attacking a Pong-DQN policy.



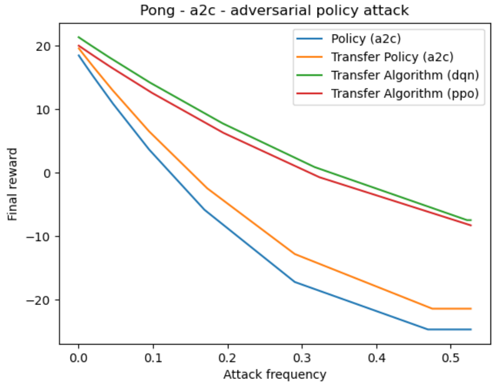
(a) Uniform attack

(b) Strategically-timed attack



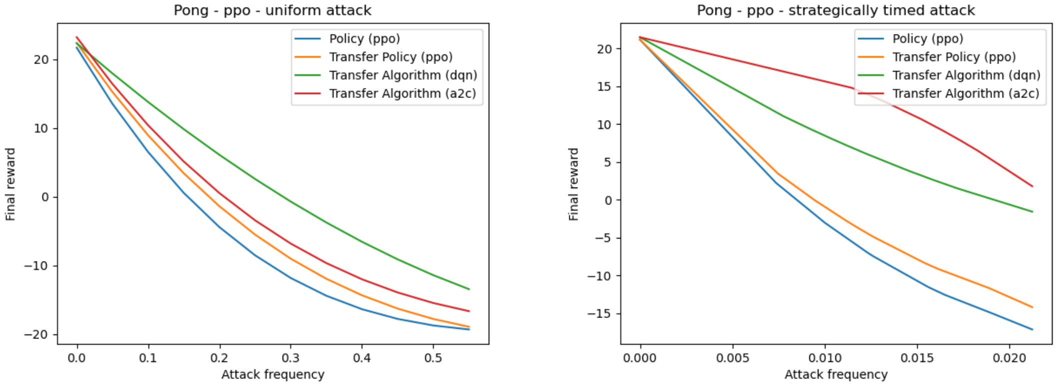
(c) Critical point attack

(d) Critical strategy attack

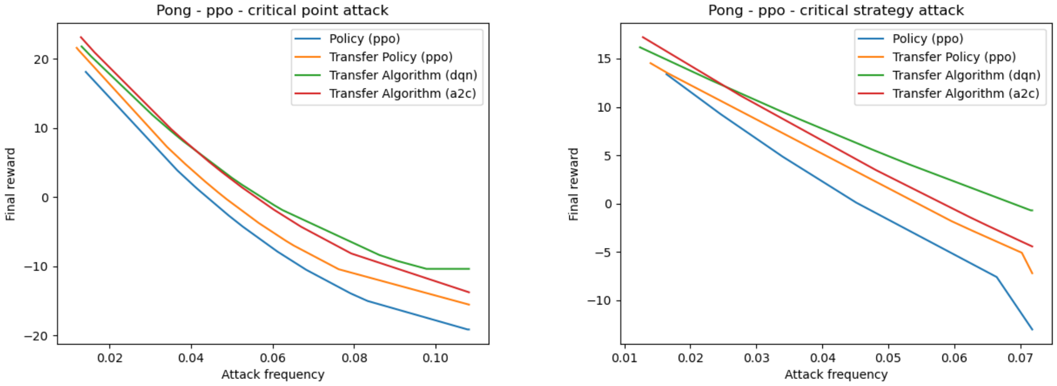


(e) Adversarial policy attack

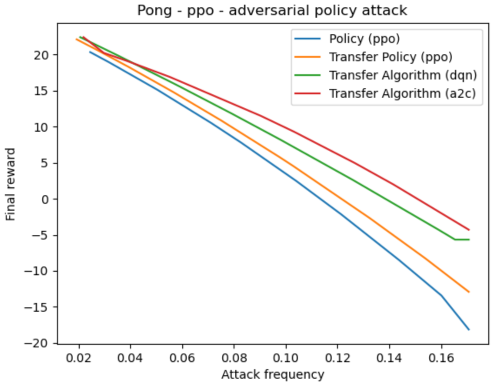
Figure 5.3 The *reward vs. attack frequency* curves of 5 policy attacks with FGSM under the l_∞ norm attacking a Pong-A2C policy.



(a) Uniform attack (b) Strategically-timed attack



(c) Critical point attack (d) Critical strategy attack



(e) Adversarial policy attack

Figure 5.4 The *reward vs. attack frequency* curves of 5 policy attacks with FGSM under the l_∞ norm attacking a Pong-PPO policy.

CHAPTER 6 STUDYING POLICY ATTACKS TRANSFERABILITY AGAINST DEFENDED POLICIES

This section reports the results of the second of the three experiments included in this thesis. It evaluates the same policy attacks but this time attacking defended policies. It still focuses on attacks' transferability and shows some interesting findings

6.1 Studying policy attacks transferability against defended policies

This experiment can be viewed as the direct extension of the previous one showed in section 5. This time, all the policies under attack have been defended with FGSM adversarial training and the goal of this work is to study the transferability of some policy attack methods against defended policies. To draw the *reward vs accuracy* curve of the evaluated methods, adversarial training has been conducted with a value of $epsilon$ slightly inferior to the $epsilon$ that defines the distance of the adversarial perturbations. In this way, it's more clear the effect of the attacks against the defended policies. The experiments have been conducted on the same three reinforcement learning algorithms adopted in section 5: DQN, A2C, and PPO with policies trained only on the environment of Pong. In addition, the policy attack methods that have been compared include:

- Uniform attack;
- Strategically-timed attack.

The reason why we have chosen only uniform attack and strategically-timed attack is to test one method based on untargeted image attacks and the second one based on targeted image attacks. Given the previous results, these two methods are enough to have a general idea of the robustness of an algorithm. Thus, the scope of the experiment has then been reduced to 2 policy attacks.

6.1.1 Experiments settings

6.1.1.1 Defenses hyper-parameters

All policies have been protected with FGSM adversarial training applied over the same policies trained normally. This means that adversarial training has not been applied on models training from scratch but it has been applied on models that had already learned a working policy. Once again, FGSM has been chosen because of its speed over iterative methods. Many works such as[42] cite that during FGSM adversarial training, the robust accuracy of the model would suddenly drop to almost 0% after a certain point. This phenomenon was commonly referred to as *catastrophic overfitting*. The reason for this phenomenon is that, since FGSM is a simple attack, the model learns to fool the FGSM attacks by inducing obfuscated gradient, thus making the gradient no longer a useful direction for constructing adversarial examples. However, the way adversarial training is applied to reinforcement learning algorithms is a little different from the standard adversarial training applied to image classification models and not much is known about catastrophic overfitting under reinforcement learning adversarial training. The second important point to remark is that during adversarial training perturbations have been crafted with $epsilon = 0.01$ and, when attacking, policies' noise $epsilon$ has been raised to 0.05. All distances are computed under l_∞ norm. This discrepancy is to make sure that the adversarial perturbations can break the defended policies so as to draw more understandable *reward vs attack frequency* curves. As we can see from table 6.1, adversarial training works well enough that adversarial observations crafted with the same perturbation distance used during training are not capable to fool the defended model. Moreover, the configuration of the algorithms during adversarial training is the same used in the previous experiments. The only small difference regards DQN, since the base policy has already been trained, the algorithm doesn't need much exploration during adversarial training. Consequently, the exploration noise has been held fixed to 0.05 for the whole process of training. The training process has been conducted over 2.5M frames for DQN and 5M frames for A2C and PPO. During adversarial training, the network is given in input adversarial observations with probability 0.5 and clean observations with the same probability so as to not forget how to deal with clear observations. In another experiment where adversarial training was conducted only using adversarial observation, the trained policy worked very well in presence of perturbations but seemed to predict random actions in correspondence to clear observations.

6.1.1.2 Image adversarial attacks settings

The configuration of the white-box attacks is identical to the one used in the previous experiment in section 5. In this way, we provide a fair comparison with the curves obtained when attacking undefended policies.

Table 6.1 Average reward scored by three policies trained with DQN, A2C and PPO respectively and defended with FGSM Adversarial training with $epsilon = 0.01$ under l norm.

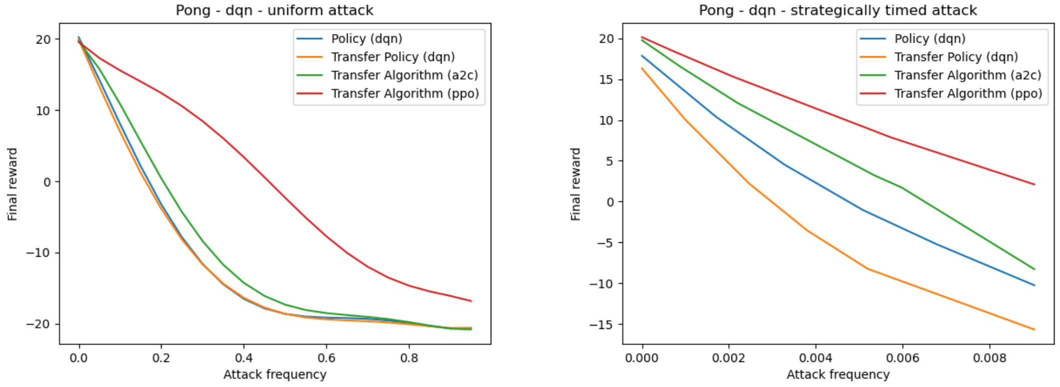
Observations	DQN (FGSM-AT)	A2C (FGSM-AT)	PPO (FGSM-AT)
Clear	19.6	18.8	19.7
Adversarial	19.4	17.9	18.7

6.2 Results

The next two sections show the *reward vs. attack frequency* curves of uniform attack and strategically-timed attack under a fixed threat model so to measure their effectiveness and their transferability against policies and algorithms protected with adversarial training. When analyzing the obtained results, sometimes we may refer to the curve of the policy attacked without transferability as baseline policy. In addition, we may sometimes intend as attacking with a surrogate policy or attacking with a surrogate algorithm as crafting adversarial perturbations exploiting a surrogate policy/algorithm to attack the target policy. Since we want to focus on studying attack transferability and that the adversary is cheating by crafting perturbations that are stronger than the ones crafted by adversarial training, we will ignore the absolute performance of the curves but we will focus on their relative distance between each other and the baseline. In fact, table 6.1 shows that, even if all observations were adversarial, that is when the attack frequency is equal to 1, crafting adversarial observations with the same perturbation strength used during training the average reward wouldn't be lowered too much respect to not performing any attack at all.

6.2.1 Attacking adversarially trained Pong-DQN

The curves obtained attacking Pong-DQN with uniform and strategically-timed attack are shown in figure 6.1. The first thing we can observe is that the curves of the baseline and of the surrogate policy are practically identical, this indicates that uniform attack benefits an excellent policy transferability. Similarly, also the curve for the algorithm transferability of A2C stays close to the baseline, thus showing a good transferability. This phenomenon is more clear in correspondence to higher attack frequencies. Only attacks crafted with PPO seem to be less capable to fool the target policy. In fact, its corresponding curve keeps a distance of about 10 points with respect to the baseline in the evaluated frequency interval. Analyzing strategically-timed attack, we can surprisingly note that attacking with the surrogate policy yields even better results than directly attack the target policy. Overall, PPO is the worst algorithm when crafting adversarial perturbations and it is always a safer choice to craft them on a policy trained with the same algorithm.



(a) Uniform attack

(b) Strategically-timed attack

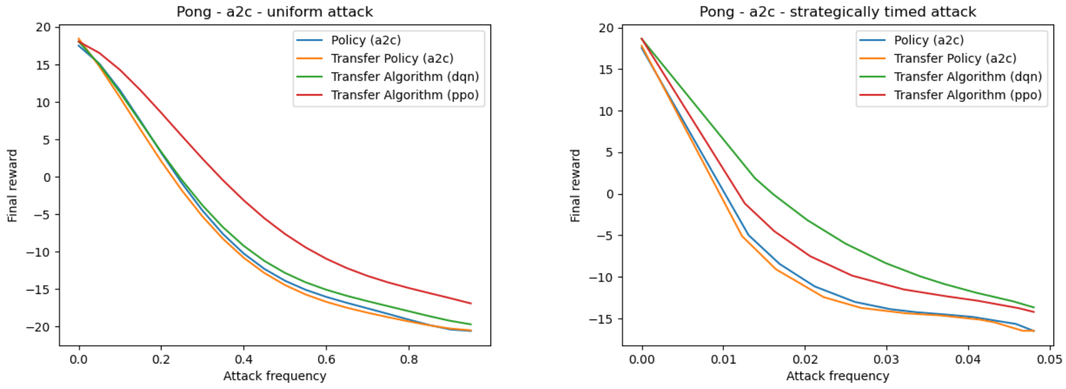
Figure 6.1 The *reward vs. attack frequency* curves of 2 policy attacks with FGSM under the l_∞ norm attacking a Pong-DQN policy defended with FGSM adversarial training with $epsilon = 0.01$. Attacks have been performed with $epsilon = 0.05$ to show the drop in reward as the attack frequency increases.

6.2.2 Attacking adversarially trained Pong-A2C

The curves resulting from attacking Pong-A2C are depicted in figure 6.2. The chart presents a similar pattern to the chart of the corresponding DQN version. In fact, the curves relative to policy transferability and DQN algorithm transferability remain close to the baseline. Attacks crafted on PPO result to be less effective with its curve remaining about 5 points over the other curves. Regarding strategically-timed attack, the curve of the surrogate A2C policy is still very close to the baseline of A2C. In this case, the perturbations crafted on DQN are the ones that perform the worst followed by A2C but still maintaining only a few points of difference with respect to the baseline.

6.2.3 Attacking adversarially trained Pong-PPO

The curves obtained from attacking Pong-PPO are represented in figure 6.3. Attacks crafted with any policy show a good transferability for the uniform attack such that choosing any surrogate model to generate adversarial observations would yield performance similar to attacking the victim model. PPO policy is a little more robust against strategically-timed attack which presents more spaced curves. More specifically, transferability over A2C works better than transferability over DQN, and transferability over

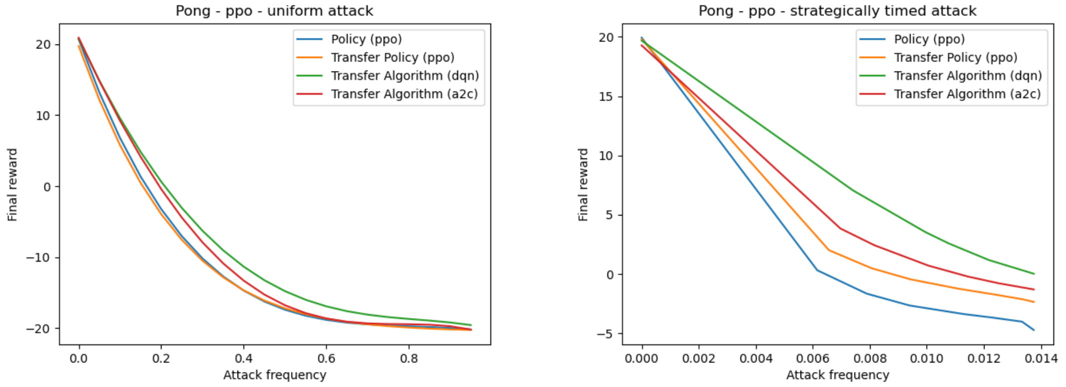


(a) Uniform attack

(b) Strategically-timed attack

Figure 6.2 The *reward vs. attack frequency* curves of 2 policy attacks with FGSM under the l_∞ norm attacking a Pong-A2C policy defended with FGSM adversarial training with $epsilon = 0.01$. Attacks have been performed with $epsilon = 0.05$ to show the drop in reward as the attack frequency increases.

PPO is the one that yields performance most similar to the baseline.



(a) Uniform attack

(b) Strategically-timed attack

Figure 6.3 The *reward vs. attack frequency* curves of 2 policy attacks with FGSM under the l_∞ norm attacking a Pong-PPO policy defended with FGSM adversarial training with $epsilon = 0.01$. Attacks have been performed with $epsilon = 0.05$ to show the drop in reward as the attack frequency increases.

6.3 Findings

This section provides further analyses about the policies that have been evaluated. In particular, it makes some conclusions about the performance obtained when attacking with similar policies and with similar algorithms.

6.3.1 Performance of similar policies

In most of the charts showed in this section as well as in section 5, the attacks crafted on the surrogate policies trained with the same algorithm of the victim policy own a better transferability than the attacks crafted on the policies trained with different algorithms. We can then conclude that attacking similar policies is more effective than attacking different algorithms. This is the same property observed when attacking common image classificators: attacking a surrogate model with the same architecture provides better results than attacking models with different architectures.

6.3.2 Performance of similar algorithms

Among the three evaluated algorithms, DQN is off-policy, which means that it learns in a different way than A2C and PPO which are on-policy. Moreover, A2C and PPO are substantially similar with the introduction of the clipping hyper-parameter being a major difference. Hence, as we can see from the charts of the experiments in both this chapter and in the previous one, a policy trained with A2C can be more easily attacked with PPO rather than with a policy trained with DQN. On the other way around, a policy trained with PPO can be more easily attacked with A2C. Thus algorithm transferability is more effective when the surrogate policy is trained with an algorithm similar to the algorithm used to train the victim policy.

CHAPTER 7 BENCHMARK OF EFFECTIVENESS OF IMAGE ATTACK METHODS AGAINST DEFENDED POLICIES

This section reports the results of the last of the three experiments included in this thesis. It benchmarks the effectiveness of different image attacks used by two different policy attacks to decrease victim's rewards. The victim models have been protected with different defense mechanisms.

7.1 Benchmark of the effectiveness of image attack methods against defended policies

The last experiment included in this work consists of a benchmark to evaluate the robustness of defended policies trained with DQN on the environment of Pong. All policies have been attacked with the white-box image attack FGSM under a fixed threat model. This work compares the *reward vs perturbation* and *accuracy vs perturbation* curves of the defended victim policies by attacking them over increasing values of perturbations' strength. Policies have been defended with the following defense methods:

- FGSM adversarial training;
- PGD adversarial training;
- JPEG conversion;
- Feature squeezing (Bit squeezing);
- No defense.

Note that in the benchmark is also included an undefended version of the victim policy so as to have a comparison of the defended policies with the undefended one. Policies have been attacked with three different white-box image attacks methods including:

- FGSM;
- PGD;
- MI (Momentum Iterative);
- No attack.

The *no attack* method which corresponds to evaluating the target policy without attacking it at all has been introduced so to compare the effect of each attack method with respect to not attacking at all. The curves have been generated attacking with both untargeted and targeted attacks and their corresponding results are reported in two different sections.

7.1.1 Experiments settings

7.1.1.1 Defenses hyper-parameters

Two of the four defenses that have been included in the benchmark consist of adversarial training with FGSM and PGD respectively. The first policy is the same policy trained with DQN used in the experiment in section 6. The policy protected with PGD adversarial training has been trained crafting adversarial perturbations with 10 iterations of PGD. Besides the type of attack and the number of iterations, there is no difference between the way adversarial training has been applied. Perturbations have been crafted under l_∞ norm and ϵ fixed to 0.01.

The third defense that has been implemented is a simple method that converts each observation to JPEG format before feeding it to the network. JPEG images retain 20% of the quality of the original observations.

The fourth method is known as feature squeezing and it consists of reducing the number of features of each pixel for each observed image. Since we are dealing with gray-scale images, what we are going to reduce is the number of bits to represent each pixel, thus we may sometimes refer to this method as bit squeezing. More specifically, it reduces the number of bits from 8 to 5, thus lowering the number of features from 2^8 to 2^5. This method shrinks the number of features by 80% with respect to the original observations.

In this benchmark we have evaluated a powerful defense method such as adversarial training and two simple defenses such as JPEG conversion and feature squeezing so to test defenses with different levels of effectiveness.

7.1.1.2 Image adversarial attacks settings

This section reports the value of the hyper-parameters adopted in the image attacks included in the benchmark. Note that this time the perturbation strength $epsilon$ is variable. The first attack, FGSM, has already been employed in the previous experiments and its hyper-parameters are maintained the same for this work. The two iterative methods PGD and MI execute 10 iterations each attack and each iteration refines the adversarial perturbations with $epsilon_i$ (attack step size) of $\epsilon/5$.

The main reason why we have chosen the above-mentioned attacks is that they all work under targeted and untargeted settings and they support the three most common distance norms, namely l_∞, l_1, l_2. They also explicitly define the hyper-parameter ϵ to limit the size of the adversarial perturbations. For example, adversarial examples crafted with C&W method are not easily comparable in this benchmark since the method doesn't have a hyper-parameter ϵ that can be fine-tuned. Moreover, those attacks are well-studied and commonly used in research papers, thus proving to work well in practice and worth being evaluated.

7.1.1.3 Threat model

All methods included in this benchmark are evaluated under both untargeted and targeted settings. Adversarial examples are crafted using the l_∞ distance norm. Their perturbations' strength ϵ ranges from 0 to 0.05 included with a step size of 0.005. Curves are generated by computing the rewards of the policy under attack and the success rate of the malicious attacks according to the value of ϵ defined in correspondence of each point.

7.1.2 Benchmark of untargeted white-box attacks

The charts relative to the *reward vs perturbation* and *accuracy vs perturbation* curves obtained under untargeted settings are shown in figures 7.1 and 7.2 respectively. When attacking the policy protected with FGSM adversarial training, MI is the attack that results most effective followed by PGD and FGSM. It is interesting to note that although adversarial training has been conducted with ϵ=0.01, adversarial training can still protect quite well against adversarial observations crafted with higher values of perturbation strength. The defense is completely broken for noise generated with ϵ greater than 0.03. Conversely, when attacking PGD adversarial training, FGSM and PGD perform the best but the difference between the three attack methods is never too significant. PGD struggles to reduce the reward of the agent against JPEG filter and bit squeezing with respect to the other two attack methods which perform very similar. FGSM is also the only attack that can't reduce the average reward to -20 when attacking the undefended policy. When comparing the success rate of the three methods, PGD is the attack the overall performs the best and FGSM is the worst. In particular, PGD adversarial training seems to be more robust against FGSM, and bit squeezing is more vulnerable against PGD attack. When attacking FGSM adversarial training and JPEG filer, all the attacks show a similar success rate.

It is interesting to note that when attacking JPEG filter and bit squeezing, all attacks seem to break the defense right after increasing the strength of the perturbations to be greater than 0.01. In fact, after this threshold, their curves drop to -20 points and the attack accuracy skyrockets to be 100%. The two defenses based on adversarial training are better at improving the robustness of their policies. As we can see from the charts, the average rewards still begin decreasing when ϵ is greater than 0.02 or 0.03 but both curves approach -20 with a more gentle descent. Conversely, the attack accuracy against the same defenses steadily increases over the whole perturbations range but the rise is never too steep.

7.1.3 Benchmark of targeted white-box attacks

The charts relative to the *reward vs perturbation* and *accuracy vs perturbation* curves obtained under targeted settings are shown in figures 7.3 and 7.4 respectively. For each observation, the targeted action that has been selected is the action with the lowest probability. Hence, it is equal to adopt strategically-timed attack attacking all the possible observations. In terms of reward, FGSM is the attack that performs the worst against the two adversarial training-based defenses while the other two attacks obtain similar results. As for JPEG filter defense, the chart doesn't present any substantial difference among the curves of the evaluated attack methods. PGD is also the best attack to break bit squeezing defense. All curves begin to drop down for perturbations greater than 0.01 and the descent is steeper for JPEG filter and bit squeezing. As it was for the untargeted case, the value of the curves of the adversarial training-based defenses decreases more gradually and eventually reaches the bottom when ϵ gets close to 0.03 or 0.04. The difference in terms of performance between adversarial training and the other two defenses is more significant when comparing their success rate curves. In fact, in the perturbations' strength interval taken into examination, targeted attacks against adversarial training protected policies work at most 20% of the times while attacks against the other two defenses have a success rate of 100% for ϵ greater than 0.01. Under targeted settings, FGSM is the only attack whose accuracy never reaches 100% against the undefended policy. It also shows an inferior performance against all the other defenses.

7.1.4 Findings

This section shows the findings relative to the benchmark evaluated in this work. In particular, it makes some conclusions about the importance of this experiment, the differences between attacking under targeted and untargeted settings and it provides more insights about the evaluated defenses and their robustness.

7.1.4.1 Importance of selecting the most appropriate image attack method

Policy attacks based on image observations rely on image attacks to craft effective perturbations to fool their target agent. Hence, selecting the most effective image attack that performs better against a certain policy trained with a certain algorithm on a specific environment is a fundamental step to perform effective policy attacks. In addition, different image attack methods might work better or worse against specific defense methods that protect the target policy. Thus, there are many variables that can condition the effectiveness of a policy attack. Also choosing the most appropriate threat model contributes to enhance the performance of the attacks and craft the most effective or imperceptible adversarial observations. For instance, in order to craft imperceptible adversarial examples, it is useful to know which minimal amount of ϵ is enough to attack policy P defended with defense D on environment E or, given a fixed ϵ, it might be useful to find which attack shows the highest attack success rate or that reduces the agent's reward for the highest amount. Thus, reinforcement learning policies that take as input environment observations should have their robustness evaluated not only against policy attacks but also against image attack methods.

7.1.4.2 Targeted and untargeted image attacks

In this section, we are going to analyze the main differences that emerge when attacking under targeted and untargeted scenarios. Targeted attacks seem to be better at reducing the agent's reward but the probability that their attacks are executed correctly is significantly lower than their untargeted counterpart. However, they are still effective since most of the failed attacks, although they fail to fool the agent to select the selected action, they still corrupt the observations enough to fool the agent to take a random sub-optimal action. However, given a similar attack accuracy, targeted attacks can easily outperform untargeted attacks. For example, given the charts relative to the undefended policy, FGSM has 100% accuracy under untargeted attack scenarios and a little more than 70% accuracy under targeted settings. Nevertheless, its average reward for the untargeted case is a few points higher than its corresponding average reward obtained in the targeted scenario. This phenomenon is probably due to the fact that targeted attacks are more lethal and when they succeed they provoke greater damage to the victim policy. It is also worth noting that, in terms of accuracy, FGSM seems to suffer a loss in performance when attacking under targeted settings with respect to its targeted version. Overall, MI is the most effective attack under both settings.

7.1.4.3 Robustness of the evaluated defenses

Among the evaluated policies, the most robust one seems to be the one defended with the two adversarial training-based methods, in fact, they offer great protection under both attack settings. PGD adversarial training is perhaps a little better than FGSM when comparing the success rate of malicious attacks. JPEG filter and feature squeezing reduce the quality of the input observations with the goal to smooth eventual malicious perturbations. The more the quality reduction, the more perturbations would be filtered out. However, an excessive quality reduction would also negatively affect the agent's performance. Hence, when those methods are used in practice, some attention should be put into finding the right tradeoff between performance and robustness. As we can see from table 6.1 adversarial training doesn't impact the agent's performance in a significant way.

7.1.4.4 Defenses applied during and after training

One difference between adversarial training and the other two defenses that have been evaluated is that the first is applied during training while JPEG compression and feature squeezing have been applied only after training, that is during evaluation. Generally speaking, adversarial training should work better since the model learns to play following its policy in presence of perturbations. However, training the model for a longer time is costly and usually constrained to learn against perturbations crafted with just one type of attack method. Fortunately, adversarial training presents a sort of transferability, meaning that models defended with FGSM can also protect against PGD attacks and vice versa. This property is clearly shown in the charts in figure 7.1 and 7.3. JPEG compression and feature squeezing can still work against a large variety of attacks but the protection that they offer is often limited.

7.1.4.5 Observations noise imperceptibility

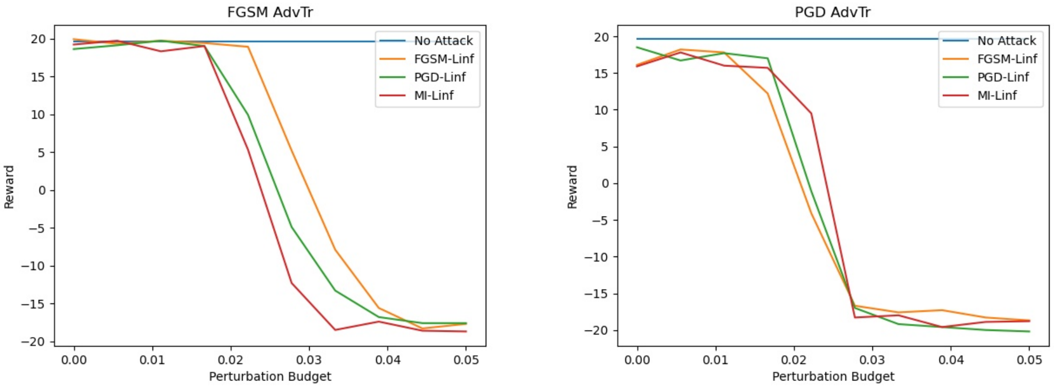
Given a fixed perturbations' strength ϵ, different image attacks can inject more or less perceptible noise to the target observations. Lower values of ϵ usually lead to less visible perturbations, while higher values can make the target images too noisy and easily perceived by the human brain. Hence, to give an idea of the effect caused by the injection of noise to clear observations, we crafted four adversarial observations with FGSM and increasing values of ϵ in the range $[0.1, 0.01, 0.001, 0.0001]$ and compared them with clear observations. Then, we crafted other four adversarial observations with PGD and with the same values of ϵ. The noisy observations and the clear ones are shown in figure 7.5. As we can immediately note in the figure, for similar ϵ PGD crafts less perceptible adversarial observations than the one generated with FGSM. The injection of noise is visible in all the malicious observations generated with FGSM, while for adversarial examples generated with PGD, only the one crafted with ϵ equal to 1 can be clearly detected. All perturbations have been crafted attacking a Pong-PPO model with l_∞ distance.

7.2 Number of attacked policies

One of the major difficulties encountered while carrying out this thesis work is the computation time taken to run all the experiments. For the first experiment have been evaluated 5 policy attacks against 3 reinforcement learning algorithms trained on 2 environments. Each policy has been attacked 4 times corresponding to the four curves shown on each chart: attacking without transferability, attacking a surrogate policy, and attacking two different surrogate algorithms. Each curve has been drawn out of 10 different attack frequencies corresponding to ten points along the x-axis. Uniform attack, since it is the fastest attack, has its corresponding curves containing 20 points. Moreover, each point has been computed as average of 10 runs. Thus, in total have been attacked at least $5 * 3 * 2 * 4 * 10 * 10 = 12000$ policies. The second experiment instead evaluates 2 policy attacks against 3 reinforcement learning algorithms trained in a single environment. Similarly, each policy has been attacked 4 times corresponding to the four curves shown on each chart as in the previous experiment. Given that each curve is composed of 10 points averaged over 10 runs, in total have been attacked at least $2 * 3 * 1 * 4 * 10 * 10 = 2400$ policies. Finally in the third experiment have been evaluated 6 policies trained with DQN where 5 of them were defended with a defense mechanism and trained on the Pong environment. Each policy has been attacked with 3 different white-box attack methods and each curve is composed of 20 points taken over an average of 10 runs. The experiment has been conducted on both targeted and untargeted threat models. Hence, in total have been attacked $6 * 3 * 2 * 20 * 10 = 7200$ policies. Hence, the whole work includes a total of 12000+2400+7200=21600 policy attacks. It took about two months using between 4 and 8 GeForce GTX TITAN X GPUs. In practice, more experiments have been conducted while tuning the attacks' hyper-parameters and verifying the correctness of all the implemented attacks.

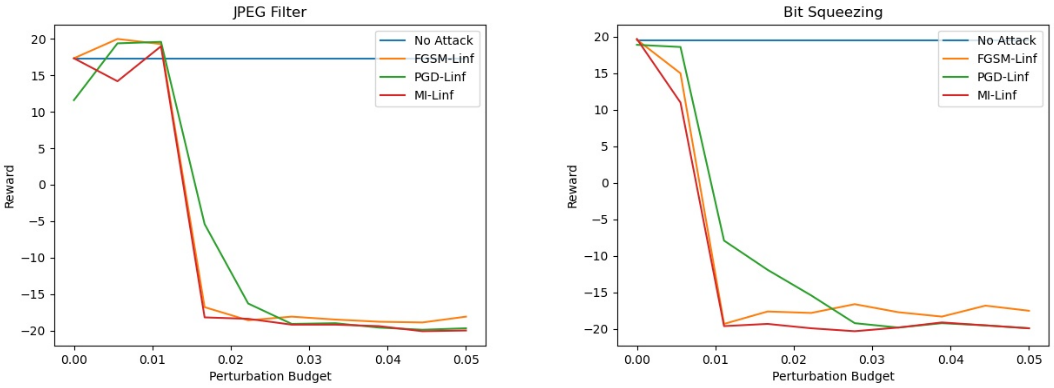
7.3 Implementation details

In this section, we are going to discuss the details relative to the implementation of the evaluated reinforcement learning algorithms, the policy attacks, and the image attacks. The Github repository containing the code to obtain the results presented in this thesis is available at the following address: https://github.com/davide97l/rl-policies-attacks-defenses. It has been developed with Pytorch but relies on other frameworks, each specialized in a different task. The reinforcement learning algorithms have been taken from the framework Tianshou[38] developed by TSAIL. It consists of a reinforcement learning platform based on pure PyTorch and with support to parallelized environments. Image adversarial attacks have been taken from Advertorch[43] developed by BorealisAI. Advertorch is a Python toolbox for adversarial robustness research also based on PyTorch. Given that the model interface required by Advertorch is different from the one required by the model trained with Tianshou, it has been created an adapter class to adapt Tianshou models to Advertorch models. The two simple image defenses, namely JPEG conversion and feature squeezing have been implemented from scratch. Policy adversarial training has been developed from scratch and integrated into Tianshou. It expands the Tianshou classes implementing the training procedure of the normal policies to defend them with adversarial training. Policy attacks have also been implemented from scratch and made compatible with Tianshou. Their modularized code allows to easily develop new attacks by expanding a base class or any of the existing policies. Hence, the result of this work is also a framework to test the robustness of reinforcement learning trained policies. Finally, the Atari environments have been taken from the Gym library developed by OpenAI. The same library also includes other well-studied environments made publicly available to the AI community.



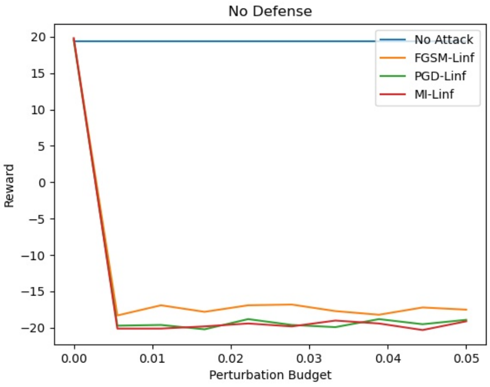
(a) FGSM adversarial training

(b) PGD adversarial training



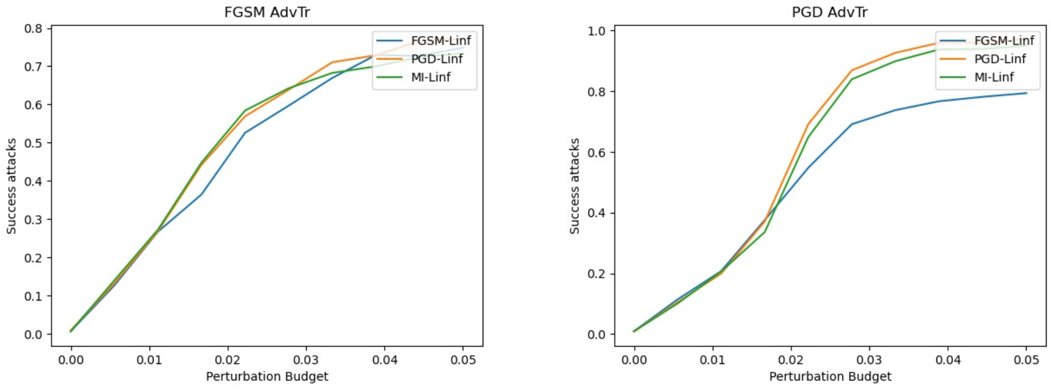
(c) JPEG filter

(d) Bit squeezing



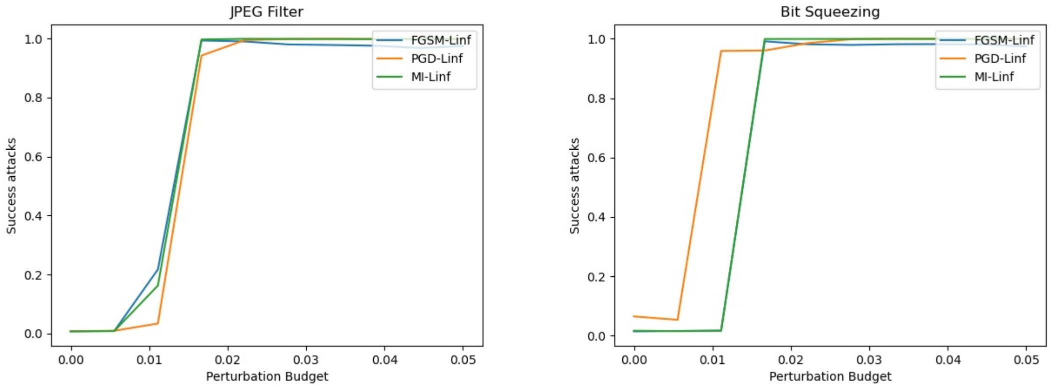
(e) No Defense

Figure 7.1 The *reward vs perturbation* curves of 5 DQN policies trained to play Pong and defended with 5 different defense methods (including no defense) under the l_∞ norm attacking with increased perturbations strength. Perturbations have been crafted in an untargeted way.



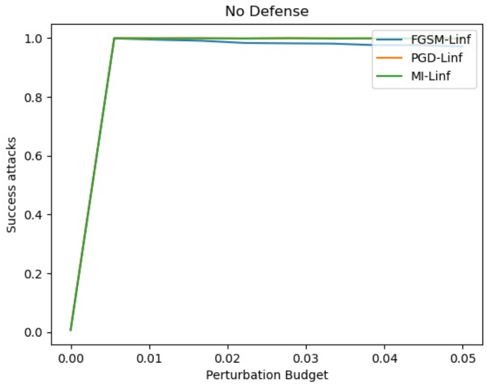
(a) FGSM adversarial training

(b) PPO adversarial training



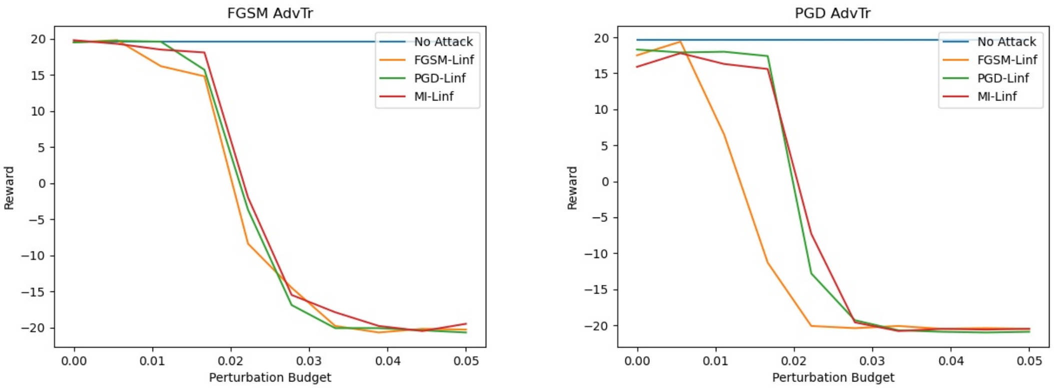
(c) JPEG filter

(d) Bit squeezing

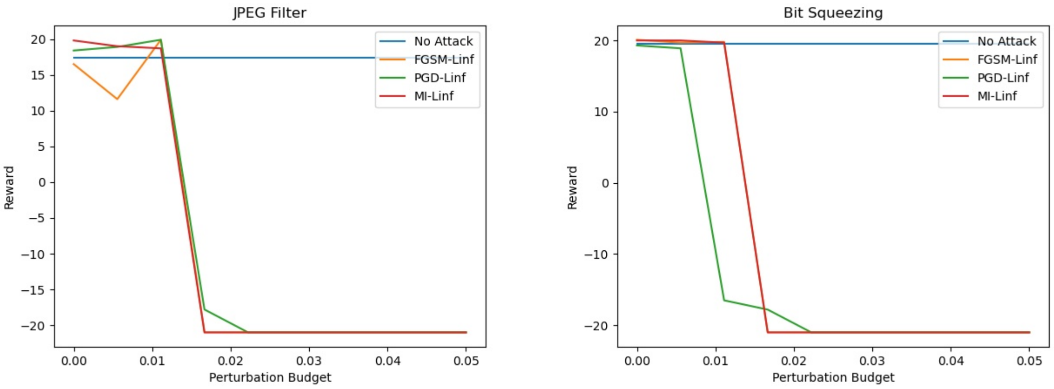


(e) No defense

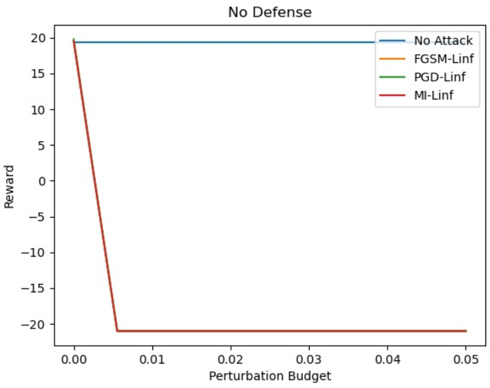
Figure 7.2 The *accuracy vs perturbation* curves of 5 DQN policies trained to play Pong and defended with 5 different defense methods (including no defense) under the l_∞ norm attacking with increased perturbations strength. Perturbations have been crafted in an untargeted way.



(a) FGSM adversarial training (b) PPO adversarial training

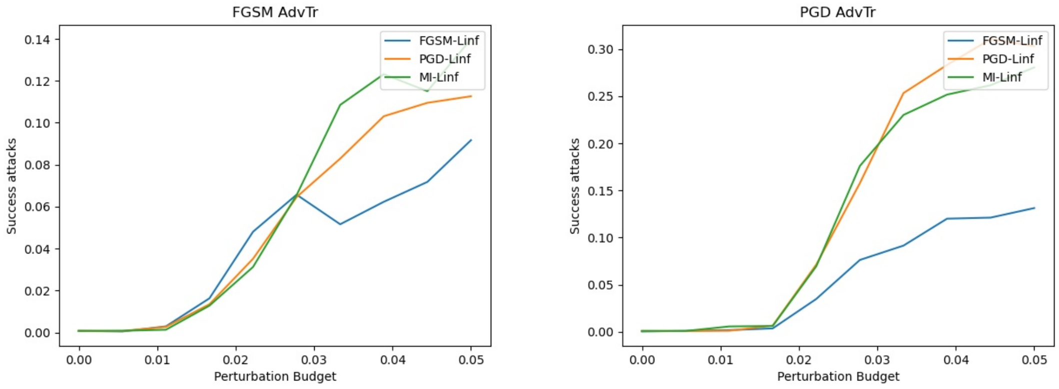


(c) JPEG filter (d) Bit squeezing

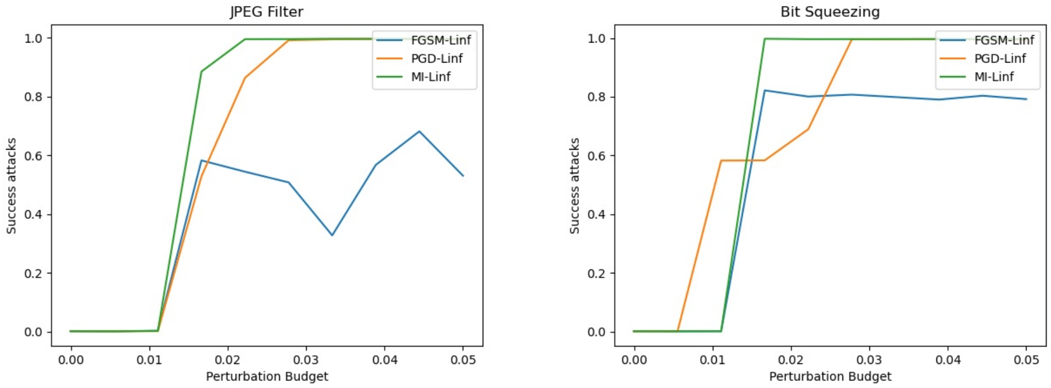


(e) No defense

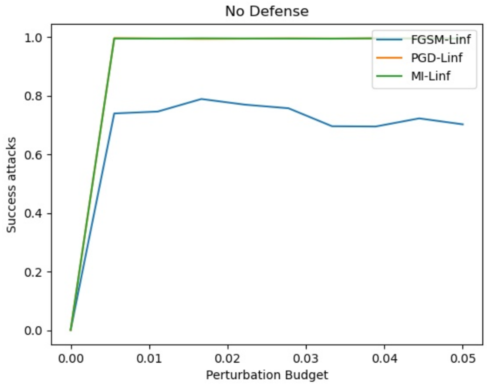
Figure 7.3 The *reward vs perturbation* curves of 5 DQN policies trained to play Pong and defended with 5 different defense methods (including no defense) under the l_∞ norm attacking with increased perturbations strength. Perturbations have been crafted in a targeted way.



(a) FGSM adversarial training (b) PPO adversarial training

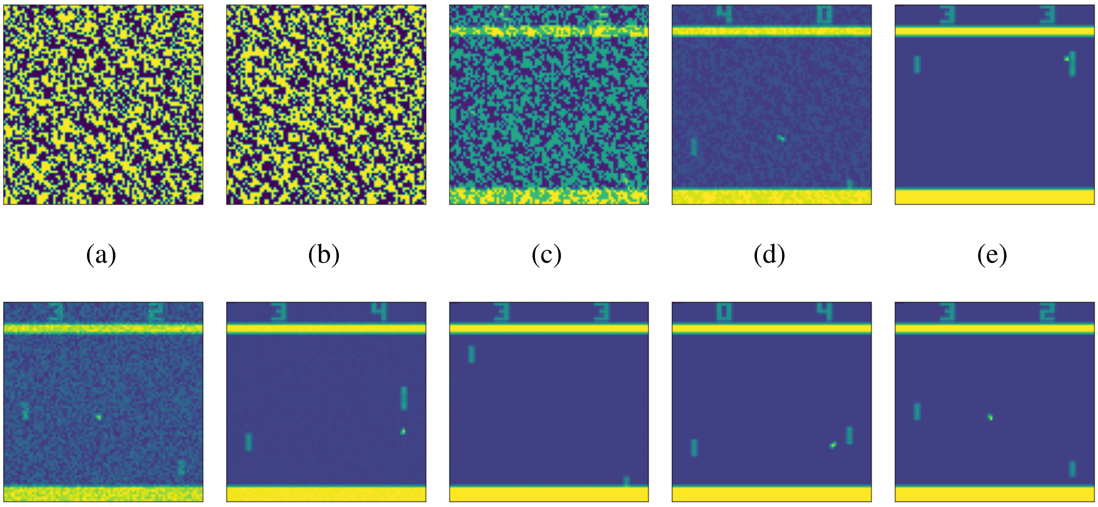


(c) JPEG filter (d) Bit squeezing



(e) No defense

Figure 7.4 The *accuracy vs perturbation* curves of 5 DQN policies trained to play Pong and defended with 5 different defense methods (including no defense) under the l_∞ norm attacking with increased perturbations strength. Perturbations have been crafted in a targeted way.



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figure 7.5 **First row**: Comparison of the effect of noise injections on Pong environment observations with different strength of perturbations. The most right one is a clear observation and from the right to the left have been injected noise with increasing perturbations $\epsilon = [0.1, 0.01, 0.001, 0.0001]$. Perturbations have been crafted with FGSM attacking a policy trained with PPO. **Second row**: Comparison of the effect of noise injections on Pong environment observations with different strength of perturbations. The most right one is a clear observation and from the right to the left have been injected noise with increasing perturbations $\epsilon = [0.1, 0.01, 0.001, 0.0001]$. Perturbations have been crafted with PGD attacking a policy trained with PPO.

CHAPTER 8 Conclusions

8.1 Summary

In this work, we conducted three experiments to study the robustness of reinforcement learning policies and their defenses. In the first part, we focused on studying policy attacks transferability over policies and algorithms. We did it by computing the *reward vs attack frequency* curves of different policy attacks under a fixed threat model so to measure their effectiveness and their transferability against policies and algorithms. We repeated the experiment for a total of 3 algorithms, 5 policy attacks, and 2 environments. In the second experiment, we studied policy attacks transferability against defended policies. We did it by computing the *reward vs attack frequency* curves of different policy attacks under a fixed threat model so to measure their effectiveness and their transferability against defended policies and defended algorithms. We repeated the experiment for a total of 3 algorithms, 2 policy attacks, and 2 environments. The third and last experiment consisted of a benchmark to measure the robustness of defended policies against white-box image attacks methods. We compared the curves of victim policies' average reward and the success rate of different white-box attack methods attacking input observations under a fixed threat model and over increasing values of perturbations' strength. Victim policies have been protected with different defense methods.

8.2 Future research directions

The present experiment, even if it took a lot of work, can still be expanded to make it more complete. The algorithms that have been examined only include DQN, A2C, and PPO, but many other more or less novel algorithms could be added to this work. For instance, DQN has many more variants worth being tested such as C51-DQN[44] and the more powerful Rainbow-DQN[45]. Similarly, future work may also introduce new defenses to make the evaluated policies more robust. Many of them combine adversarial training with auxiliary loss functions to train policies with certified robustness[46] [35] [36] [34]. The main image attack used as a backbone by the policy attacks is FGSM and attacks relying on the two iterative methods, PGD and MI, have only been partially studied in the last experiment. However, this work could be extended by evaluating more diverse white-box attacks such as C&W[22] and Deepfool[40]. Instead of only focusing on black-box attacks, it could be interesting to know whether some black-box attacks could further boost the performance of the policy attacks. The only environments that have been explored in this work are simple Atari games with discrete actions. However, in some real scenarios, it is common to have to solve reinforcement learning tasks whose agent has to take continuous actions to interact with the environments. For example, most environments included in the categories MuJoco and Box2D have a continuous actions space which would cause the current implementations of DQN, A2C, and PPO to be useless. Hence, future work might also take into consideration to test the robustness of models specialized to deal with continuous actions environments such as SAC[47], TD3[48] and DDPG[49]. Another possible scenario is when it is not possible to directly inject noise to the input observations but it is possible to generate adversarial observations by training a malicious agent to interact with the environment in a way such as to fool the victim agent. Under this particular scenario, new kinds of attacks and defenses are possible in order to test the robustness of the target policies.

REFERENCES

[1] Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning[Z]. 2013.

[2] Silver D, Hubert T, Schrittwieser J, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm[Z]. 2017.

[3] Arulkumaran K, Cully A, Togelius J. Alphastar[J/OL]. Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2019. http://dx.doi.org/10.1145/3319619.3321894.

[4] Silver D, Huang A, Maddison C J, et al. Mastering the game of go with deep neural networks and tree search[J/OL]. Nature, 2016, 529: 484-503. http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.

[5] Schrittwieser J, Antonoglou I, Hubert T, et al. Mastering atari, go, chess and shogi by planning with a learned model[J/OL]. Nature, 2020, 588(7839): 604–609. http://dx.doi.org/10.1038/s41586-020-03051-4.

[6] Ecoffet A, Huizinga J, Lehman J, et al. Go-explore: a new approach for hard-exploration problems[Z]. 2019.

[7] Ye D, Liu Z, Sun M, et al. Mastering complex control in moba games with deep reinforcement learning[Z]. 2019.

[8] Huang S, Papernot N, Goodfellow I, et al. Adversarial attacks on neural network policies[Z]. 2017.

[9] Dong Y, Fu Q A, Yang X, et al. Benchmarking adversarial robustness[Z]. 2019.

[10] Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning[Z]. 2016.

[11] Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms[Z]. 2017.

[12] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[M/OL]// Pereira F, Burges C J C, Bottou L, et al. Advances in Neural Information Processing Systems 25. Curran Associates, Inc., 2012: 1097-1105. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[13] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[Z]. 2015.

[14] Zhang H, Wu C, Zhang Z, et al. Resnest: Split-attention networks[J]. arXiv preprint, 2020.

[15] Goodfellow I J, Shlens J, Szegedy C. Explaining and harnessing adversarial examples[Z]. 2014.

[16] Kurakin A, Goodfellow I, Bengio S. Adversarial examples in the physical world[Z]. 2016.

[17] Brendel W, Rauber J, Bethge M. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models[Z]. 2017.

[18] Dong Y, Liao F, Pang T, et al. Boosting adversarial attacks with momentum[Z]. 2017.

[19] Chen P Y, Zhang H, Sharma Y, et al. Zoo[J/OL]. Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security - AISec '17, 2017. http://dx.doi.org/10.1145/3128572.3140448.

[20] Agarwal C, Dong B, Schonfeld D, et al. An explainable adversarial robustness metric for deep learning neural networks[Z]. 2018.

[21] Madry A, Makelov A, Schmidt L, et al. Towards deep learning models resistant to adversarial attacks[Z]. 2019.

[22] Carlini N, Wagner D. Towards evaluating the robustness of neural networks[Z]. 2016.

[23] Papernot N, McDaniel P, Wu X, et al. Distillation as a defense to adversarial perturbations against deep neural networks[Z]. 2016.

[24] Hein M, Andriushchenko M. Formal guarantees on the robustness of a classifier against adversarial manipulation[Z]. 2017.

[25] Wong E, Rice L, Kolter J Z. Fast is better than free: Revisiting adversarial training[Z]. 2020.

[26] Dziugaite G K, Ghahramani Z, Roy D M. A study of the effect of jpg compression on adversarial images[Z]. 2016.

[27] Das N, Shanbhogue M, Chen S T, et al. Shield: Fast, practical defense and vaccination for deep learning using jpeg compression[Z]. 2018.

[28] Xu W, Evans D, Qi Y. Feature squeezing: Detecting adversarial examples in deep neural networks[J/OL]. Proceedings 2018 Network and Distributed System Security Symposium, 2018. http://dx.doi.org/10.14722/ndss.2018.23198.

[29] Lin Y C, Hong Z W, Liao Y H, et al. Tactics of adversarial attack on deep reinforcement learning agents[Z]. 2017.

[30] Huang S, Papernot N, Goodfellow I, et al. Adversarial attacks on neural network policies[Z]. 2017.

[31] Gleave A, Dennis M, Wild C, et al. Adversarial policies: Attacking deep reinforcement learning[Z]. 2020.

[32] Sun J, Zhang T, Xie X, et al. Stealthy and efficient adversarial attacks against deep reinforcement learning[Z]. 2020.

[33] Papernot N, McDaniel P, Wu X, et al. Distillation as a defense to adversarial perturbations against deep neural networks[Z]. 2015.

[34] Oikarinen T, Weng T W, Daniel L. Robust deep reinforcement learning through adversarial loss[Z]. 2020.

[35] Fischer M, Mirman M, Stalder S, et al. Online robustness training for deep reinforcement learning[Z]. 2019.

[36] Zhang H, Chen H, Xiao C, et al. Robust deep reinforcement learning against adversarial perturbations on state observations[Z]. 2020.

[37] Brockman G, Cheung V, Pettersson L, et al. Openai gym[Z]. 2016.

[38] Jiayi Weng A D K Y D Y H S J Z, Minghao Zhang. Tianshou[J/OL]. GitHub repository, 2020. https://github.com/thu-ml/tianshou.

[39] Schaul T, Quan J, Antonoglou I, et al. Prioritized experience replay[Z]. 2016.

[40] Moosavi-Dezfooli S M, Fawzi A, Frossard P. Deepfool: a simple and accurate method to fool deep neural networks[Z]. 2016.

[41] Kim S W, Zhou Y, Philion J, et al. Learning to simulate dynamic environments with gamegan[Z]. 2020.

[42] Li B, Wang S, Jana S, et al. Towards understanding fast adversarial training[Z]. 2020.

[43] Ding G W, Wang L, Jin X. AdverTorch v0.1: An adversarial robustness toolbox based on pytorch[J]. arXiv preprint arXiv:1902.07623, 2019.

[44] Bellemare M G, Dabney W, Munos R. A distributional perspective on reinforcement learning[Z]. 2017.

[45] Hessel M, Modayil J, van Hasselt H, et al. Rainbow: Combining improvements in deep reinforcement learning[Z]. 2017.

[46] Wang F, Zhong C, Gursoy M C, et al. Adversarial jamming attacks and defense strategies via adaptive deep reinforcement learning[Z]. 2020.

[47] Haarnoja T, Zhou A, Abbeel P, et al. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor[Z]. 2018.

[48] Fujimoto S, van Hoof H, Meger D. Addressing function approximation error in actor-critic methods[Z]. 2018.

[49] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning[Z]. 2019.

ACKNOWLEDGEMENTS

I want to thank my supervisor Zhu Jun for the support and guidance he gave me during the realization of this work. In particular, he provided me important feedback about the results that have been obtained and suggested me some interesting experiments to include in my work. I have also to thank Weng Jiayi, the main developer of Tianshou, for his useful discussions about the implementations of the reinforcement learning algorithms and the way to fine-tune their hyper-parameters specifically for the Atari games. Finally, I would like also to thank my family for the support they gave me during these two years at Tsinghua University and all the friends I met on campus for all the fantastic experiences lived together.

声 明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签 名：____________ 日 期：____________

RESUME

Davide Liu was born on 27th August 1997 in Padova, Veneto, Italy.

He began his bachelor's study in the Department of Computer Science, Padova University in September 2016, majoring in computer science, and got a Bachelor of Science degreeth in July 2019.

He began his master's study in the Department of Computer Science, Tsinghua University in September 2019, and is expected to get a Master of Science degree in Computer Science in July 2021.

COMMENTS FROM THESIS SUPERVISOR

深度学习模型的对抗攻击与防御是当前研究的热点，是深度学习模型在应用过程中需要重点关注的问题。目前在图像识别等任务上，相关研究进展较多。但对于使用了深度神经网络的深度强化学习，这方面的研究相对较少。此外，对抗攻击与防御是矛与盾的关系，一方面的进展会促进另一方面的提升，因此，如何进行充分的评测是其中要解决的重要问题。为此，该论文针对一些常见的深度强化学习算法的对抗鲁棒性进行评测，通过多个决策任务下的实验，展示它们在恶意策略攻击下的迁移性，同时，也对基于图像的对抗攻击进行了实验测试。该论文的选题具有实际应用价值。作为一个实验验证的论文，相关实验在多个任务场景下开展，工作量充足，结论合理。论文写作符合规范。满足硕士论文的要求。

RESOLUTION OF THESIS DEFENSE COMMITTEE

Adversarial attack and defense for deep learning is an important and timely topic that concerns the robustness of deep learning in real-world applications. Much work has been done on pattern recognition tasks, while the work on reinforcement learning is less yet increasing. Moreover, there is typically an arm race between attack and defense, thereby calling for comprehensive benchmarks.

In this thesis, the author systematically studied adversarial robustness on deep reinforcement learning (DRL). It carried out a series of empirical studies of both attack and defense on various video games. Through the experimental results, the author shows the vulnerability of several representative DRL algorithms against attacks that design malicious policies, their transferability, and the ability of image-based attacks to craft effective adversarial observations, even in presence of defense mechanisms.

Through this thesis work, 刘大为 shows his solid knowledge on theoretical fundamentals, his professional knowledge, and his ability to conduct research independently. The thesis is well organized and written. During the oral defense, he presented his work clearly and answered questions correctly.

The defense committee voted unanimously in favor of 刘大为's thesis, and recommended that the degree of Master of science be conferred on him.