Stress Test: Evaluating the 'Importance Judger' Algorithm Against State-of-the-Art Attacks

A Technical Report submitted to the Department of
Computer Science

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Maxwell Lennon
Spring 2022

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Signature _____Maxwell Lennon_____ Date __5/11/22__
Maxwell Lennon

Advisor: Yuan Tian, Department of Computer Science

Stress Test: Evaluating the 'Importance Judger' Algorithm Against State-of-the-Art Attacks

Maxwell Lennon
University of Virginia, mrl7ja@virginia.edu

***Abstract* - It is well-documented that machine learning models are vulnerable to adversarial examples, which are instances deliberately constructed in order to influence the model into making incorrect decisions. Deep Reinforcement Learning (DRL) can be effectively attacked by adding perturbations to observations. An ongoing research endeavor proposes a detector algorithm based on the extent of the effect of changing the action decision made at this observation to the final reward. The algorithm, Importance Judger, detects attacks based on the changes of perceived importance for these observations. Thus far, the detector created using this algorithm has been evaluated under attack FGSM, BIM and CW_2 in three OpenAI gym Atari game environments: Pong, Breakout and Seaquest. The detection accuracy and F_1 score on important observations under different attack scenarios has been observed to exceed 90%. However, a recently developed perturbation attack, PA-AD, has demonstrated superior effectiveness over the above methods. This paper documents the process thus far of ascertaining the utility of Importance Judger against state-of-the-art attack methodologies by evaluating its performance against PA-AD.**

1. Introduction

1.1 Deep Reinforcement Learning

Reinforcement learning (RL) is a sub-field of machine learning (ML), a field which is defined by the process of causing a program's performance on a certain task to improve given data. Within this framework, RL involves optimizing a policy for an agent to follow based on the assignment of algorithmic 'punishments' and 'rewards,' the criteria for which are determined ahead of time. More specifically, deep reinforcement learning (DRL) attempts to achieve complex variations of this goal using specific types of neural network architectures. The uses for deep reinforcement learning range from thought-provoking demonstrations of proficiency in various popular games to high-stakes applications such as autonomous vehicles, robotics, and medical analysis. However, like other subdisciplines of machine learning, DRL has been shown to be vulnerable to adversarial manipulations, in which a malicious outsider attempts to externally alter the actions of a RL-trained agent and induce incorrect behavior.

1.2 Importance Judger

The concurrent research project that inspires this investigation introduces a technique termed an importance judger, which is designed to assess the relevance of a given input towards the eventual assignment of a punishment or a reward. In other words, the importance judger seeks to determine the expected change in the reward function (which the agent's behavior attempts to maximize) based on a known change to the input. The algorithm quantifies the expected impact of an individual observation on the final reward earned by the agent by computing the variance of the action probability vector associated with

a given state observation. Analysis of changes to the expected impact has the potential to reveal compromised inputs, in that a previously important observation or input that changes to being perceived as unimportant is indicative of an adversarial attack. The importance judger thus provides a tool with the potential to identify a compromised agent or model without analyzing the agent's behavior directly (which may not be possible or reliable depending on the nature of the attack).

Table 1: Test accuracy of Importance Judger classifier in different environments.

Game	Normal	Attacked (FGSM)	Attacked (BIM)	Attacked (CW)
Breakout	0.912	0.857	0.817	0.872
Pong	0.988	0.986	0.988	0.987
Seaquest	0.905	0.918	0.926	0.927

The research experimentation with Importance Judger thus far has tested the effectiveness of the algorithm against three perturbation attack methods: Fast Gradient Signed Method (FGSM), Basic Iterative Method (BIM) and Carlini-Wagner. Training classifiers based on observations perturbed via these methods resulted in detection accuracies up to 90% in the chosen test environments (namely, the Atari games Breakout, Pong, and Seaquest).

1.3 PA-AD

Whereas most perturbation-based attacks learn an end-to-end adversary policy that maps observations directly into observation perturbation, a new DRL attack algorithm, called Policy Adversarial Actor Director (PA-AD), decouples the attacking process into two simpler components: policy perturbation and state perturbation, solved by a "director" and an "actor" respectively. The director learns the optimal policy perturbing direction with RL methods, while the actor generates adversarial states at each time step such that the victim policy is perturbed towards the direction specified by the director. The paper proposing this algorithm claims it to be both optimal and efficient, and demonstrates that it is extremely powerful on environments with large observation spaces.

2. Methods

2.1 Implementation Refactoring

The existing codebase implementing this algorithm (used by Suya et al.) employs the deep learning framework Tensorflow 1.x, a computational graph-based Python framework. In order to integrate existing DRL methodologies, including allowing the Importance Judger to interface with attacks generated by PA-Ad, I deemed it necessary to create an implementation of Importance Judger using Tensorflow 2.x, the current version which emphasizes a functional program flow. Converting the codebase to Tensorflow 2 involved re-implementing the deep Q-learning agent model, the importance judger classifier itself, and the previously employed attack methodologies, among other components.

2.2 Agent Training

Resulting from the need to upgrade the codebase to current versions of Tensorflow, the suite of existing pre-trained deep learning agents created to play the chosen test bench Atari games was non-recoverable and needed to be recreated using the updated model and optimization frameworks. This task presented a significant challenge, as variations in the agent's starting behavior and randomness in its (epsilon-greedy) exploration resulted in a non-deterministic process of hyperparameter tuning

in order to achieve an agent with skill level matching the baselines demonstrated in previous research.

In particular, games requiring complicated policies in order to achieve player success proved especially difficult to train. For example, in the game Seaquest, the player must rescue divers as a submarine while avoiding enemies in an underwater scene. Touching an enemy will result in the player resetting and losing a life; this straightforward relationship is quickly learned and understood by a deep-Q-learning agent in most cases. However, the player can also lose a life if it fully depletes the oxygen meter, which steadily decreases while the player is below the surface. Surfacing replenishes the player's oxygen level, but also kills the player if no divers are onboard the submarine. Thus, in order to survive for as long as possible and gain the maximum amount of points, the agent must learn to surface regularly if and only if a diver has been rescued. This behavior pattern requires multiple display elements to be analyzed simultaneously in an interwoven relationship, and converging on said behavior proved difficult for an agent learning through a combination of existing behavioral tendencies and random exploration.

Owing to the challenges associated with finding the right combination of hyperparameters while also happening upon optimal behavior through semi-random exploration, I have thus far not able to train a Seaquest player to the standards of the existing research, and so have elected to exclude the environment from the reported results.

2.3 PA-AD Implementation

The only existing codebase implementing the PA-AD algorithm proposed by Huang et al. is written using the deep learning framework Pytorch, which is distinct from both Tensorflow 1.x and 2.x. In order to connect the attacks produced by PA-AD with the defense of Importance Judger, it was necessary to create an implementation of PA-AD using Pytorch, involving a similar re-implementation process to converting the Importance Judger codebase to Tensorflow 2.

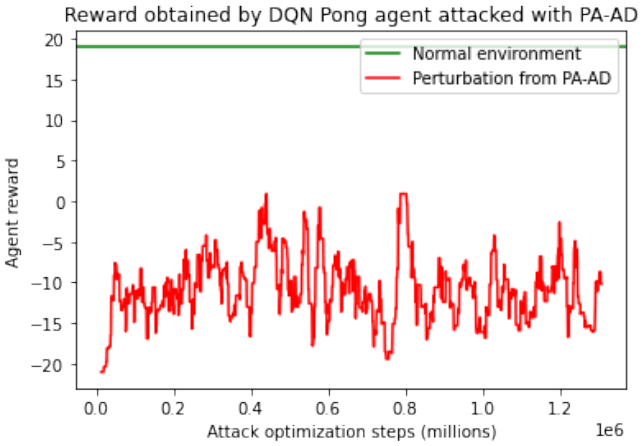
3. Results / Progress

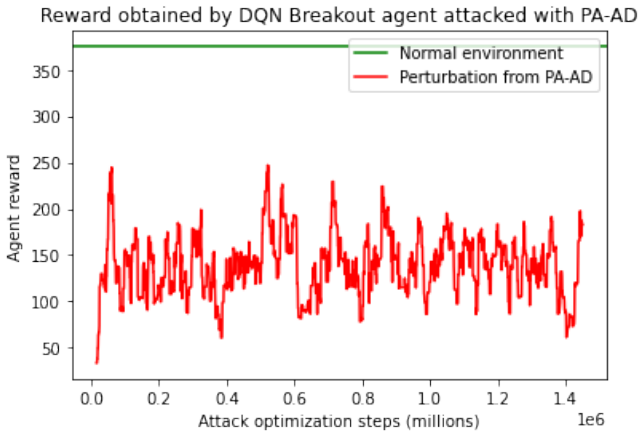
3.1 Trained Agents

Using the reimplementation of the original Importance Judger codebase including deep learning agent training, I created an agent capable of achieving an average reward of 18.95 (representing margin of victory in a game to 21 points) in Pong. Additionally, I trained a Breakout agent that achieves an average reward of 375 (representing the raw in-game score). Videos of agents playing these respective games are available as part of the supplementary materials.

3.2 PA-AD Attacks

Using the Tensorflow implementation of PA-AD, I created attacks which severely decrease the performance of an otherwise skilled agent in both Pong and Breakout.





4. Limitations

Several factors presented challenges in integrating Importance Judger and PA-AD. One such challenge was the different environment specifications between the two code-bases. Both Importance Judger and PA-AD rely natively on OpenAI's "gym" package to create their Atari environment, but PA-AD included the use of wrapper code intended to speed up training by vectorizing the environments across multiple simultaneous processes. The agents have proven to be sensitive to slight changes in their environment, and so work to transfer agents trained in one codebase to operate in the other is ongoing.

Additionally, the search process for PA-AD involves stochastic elements similar to the process of training a DRL agent, and so settling on an optimal perturbation strategy is not guaranteed. As can be seen in the provided plots, the reward of the attacked agent fluctuates in both directions as the actor and director explore their perturbation space.

5. Future Work

Beyond the goal of this research project, which is to evaluate the performance of a classifier created using the Importance Judger algorithm in detecting attacks created with PA-AD, one possible direction for expansion of this work would involve creating a version of PA-AD that has access to the importance judger itself, instead of only the DRL agent. It is possible that the perturbation space could allow PA-AD to craft an adversarial instance that exploits vulnerabilities in the model used by Importance Judger.

Work Cited

Huang et al, “Who Is the Strongest Enemy? Towards Optimal and Efficient Evasion Attacks in Deep RL” In the 10th International Conference on Learning Representations (ICLR 2022)