ATTACKBENCH: TAXONOMY AND EVALUATION FRAMEWORK FOR ADVERSARIAL REINFORCEMENT LEARNING

Nikhil Suri [1] **Thushan Puhalendran** [1] **Sam Clay** [2]

ABSTRACT

This paper introduces a framework for understanding and analyzing adversarial attacks on Reinforcement Learning algorithms. We first provide an overview of common attacks that are used against agents. Next, we classify these attacks into different buckets based on how they are constructed and applied. Finally, using open-source and published tools and code repositories, we conduct several experiments which evaluate the robustness of agents trained with different learning algorithms under our framework. Using the results that we obtain from our experiments, we develop a benchmarking suite for robust Reinforcement Learning agents that can inform future work.

1 INTRODUCTION

Artificial Intelligence is an incredibly promising field of research which, in recent years, has spawned millions of dollars of funding and powered the technology behind companies now worth billions. Reinforcement Learning is one of the principal methods that powers many AI systems today. Recent research in Reinforcement Learning has focused on the area of Deep Reinforcement Learning, which explores the application of deep neural networks to train agents to accomplish tasks that range from flying a drone, to manipulating a Rubik's Cube, to playing video games such as Dota 2 against e-sports professionals.

A commonly cited concern with the area of Deep Reinforcement Learning research, however, is the very experimental and empirical nature of the field. Often, training deep neural networks to perform reasonably well at a task involves some fairly arbitrary and ill-explained choices for network architectures, hyper-parameters, and initialized weight values. Even more importantly, the fact that deep neural networks are still not well understood and we do not have formal guarantees for certain properties that we may wish for them to exhibit has led to studies which show deep neural networks failing at tasks when given minimally perturbed inputs. This has motivated the study of adversarial inputs which can be constructed to fool deep neural networks. Now, many attacks exist. Some require knowledge of the underlying network architecture to construct, while others can be applied as "black-box" attacks.

While attacks on deep neural networks have been studied, the area of deep reinforcement learning offers a new ground of research to study attacks. Due to the iterative state-exploring nature of reinforcement learning, existing attacks against deep neural networks can be applied in novel and clever ways to reinforcement learning models. For example, some attacks are able to not only fool the model at just a single random state, but also at strategically-timed states or to lure the agent into specific low-value states. In multi-agent settings, agents can also learn and exploit other's model rewards structures with no prior knowledge. These different attack methodologies are covered in our framework.

After building a taxonomy and evaluation framework, we also investigate and implement case studies of examples of key adversarial attacks on reinforcement learning including gradient-based attacks on Atari games and end-to-end attacks on text classifiers and dual-agent games. We show that despite their varied formats, these attacks all consistently fit within our performance evaluation framework based on adversarial reward and our self-defined perturbation effort metric in 4.2.2.

The structure of this paper is as follows: In Section 2, we review prior work that has been done in both adversarial machine learning and adversarial reinforcement learning. In Section 3, we briefly cover the theory of reinforcement learning and define what an adversarial attack on reinforcement learning is. In Section 4, we present our framework of adversarial attacks on reinforcement learning and organize common types of attacks into different buckets. Sections 5 and 6 cover our experiments, case studies and provide our results and classifies important existing results within our framework. Section 7 covers our remarks on future work to continue to build upon our framework and benchmarking suite. Finally, we conclude in Section 7.

[1]Harvard College [2]Harvard Graduate School of Design.

2 Related Work

2.1 Attacks

One of the fundamental attacks against deep neural network is known as the Fast Gradient Sign Method (FGSM), introduced by Goodfellow et al. (2014). Given a model θ with input x, label y for input x, and cost $J(\theta, x, y)$ used to train the model, then FGSM obtains a perturbation by computing

$$\eta = \epsilon \operatorname{sign} \nabla_x J(\theta, x, y) \tag{1}$$

The value of the parameter ϵ is often described as the "input variation parameter". Increasing its value tends to result in a more effective attack, but makes the attack more detectable to humans.

Papernot et al. (2016) cover black-box attacks against deep neural network policies. They use the FGSM and another algorithm introduced by Papernot et al. (2015a). "Black-box" attacks are those that have no information about the structure or parameters of the network and do not have access to any large training datasets. They are therefore crafted on the assumption that the adversary can only observe the outputs of the network. In the classic machine learning setting, these are the labels assigned by the model for different inputs. In the reinforcement learning setting, these are the actions taken by the RL agent.

Gleave et al. (2019) study black-box attacks in a strictly reinforcement learning multi-agent zero-sum game setting in the MuJoCo physics simulator. They train adversarial policies using Proximal Policy Optimization (PPO) and train victim policies using LSTM and MLP architectures.

Behzadan & Munir (2017) craft adversarial examples by training a replica of the target network which observes the actions that the target network takes at every time step. The adversarial network then crafts a attacks to induce adversarial actions. This is similar to the strategically-timed and enchanting attacks introduced by Lin et al. (2017). A variation on these kinds of attacks is the Fractional-state adversary (FSA) shown by Qu et al. (2019).

In response to research on the robustness of neural networks, Carlini & Wagner (2016) also introduce three attacks (the L_0, L_2, and L_∞) which significantly reduce the effectiveness of a common training mechanism used to achieve robustness known as defensive distillation (see Section 2.2).

2.2 Robustness

Given the amount of research that has recently been dedicated to constructing and applying attacks on machine learning and reinforcement learning, studying the robustness of neural networks is another fairly active area. Pattanaik et al. (2017) create robust models by applying adverarial attacks during the training phase to agents using learning algorithms such as DQN and DDPG.

Defensive Distillation is a method of training robust neural networks introduced by Papernot et al. (2015b). They show that at test time, models trained using defensive distillation are less sensitive to adversarial examples than those trained without. Distillation in neural networks (Hinton et al., 2015) aims to reduce the size of the networks by extracting probability vectors to train a smaller model without a loss of accuracy. Distillation, as noted by Papernot et al. (2015b) can also improve the generalizability and resilience of neural networks. Therefore defensive distillation performs distillation without reducing the size or changing the architecture of the network.

In response to the L_2, L_0, and L_∞ attacks published by Carlini & Wagner (2016), Madry et al. (2017) define "first-order" adversaries as those that rely on only first-order information, such as the input or cost function of the model. They next identify projected gradient descent (PGD) as the strongest "first-order" adversary, or adversary which utilizes first-order information. Based on this insight, they show that models trained on PGD made adversarial examples are resistant to white-box and black-box attacks. Most notably, they demonstrate resistance to the attacks introduced by Carlini & Wagner (2016).

2.3 Frameworks

The only other framework similar to the one which we present in this paper is presented by Huang et al. (2017). They evaluate the effectiveness of adversarial examples crafted by the FGSM over models using DQN, DDPG, TRPO, and A3C architectures over different Atari games. They study the vulnerability of each game to white-box and black-box attacks. However, the primary drawback of their study is that they do not present a solid framework or benchmarking suite which others can use to evaluate their own trained agents. They simply show experimental results for a certain subset of Atari games and network architectures.

3 Background

3.1 Deep Reinforcement Learning

A reinforcement learning agent exists within in an environment parameterized by states in state space S and actions in action space A. Given a state s, the agent uses a policy π to determine the action to take from that state. π is therefore a function from $S \rightarrow A$. There is also a transition function T which, given a state s and action a, produces the next state s' and reward r for moving from s to s'. It is important to note that T is not necessarily deterministic. Reinforcement learning trains a policy π^* to optimize the expected reward accumulated over time. Deep Reinforcement learn-

ing trains the policy π^* by using a deep neural network to learn complicated relationships between states and actions. Many deep reinforcement learning algorithms exist such as the Deep Q Network (DQN), Deep Deterministic Policy Gradients (DDPG), Proximal Policy Optimization (PPO), and Asynchronous Actor-Critic Agents (A3C).

3.2 Adversarial Attacks on Deep Reinforcement Learning

We define an adversarial attack on a reinforcement learning agent as an input that causes at least one of the following behaviors:

1. It causes the agent to pick a sub-optimal action at the time it was input to the agent.
2. Over the course of some number of time steps t, during which the agent visits states $s, s_1, \ldots, s_t$, it causes the agent to accumulate less reward than it would have if it had taken the optimal action at every single state.

There is no single way to construct an adversarial attack. As we will go over in our framework, an attack may be constructed in a many number of ways.

4 Adversarial Attack Framework

4.1 Attack Styles

4.1.1 Gradient Based Attacks

Gradient-based attacks combine an appropriate gradient search method with a specific style of attack. Popular methods are FGSM, Carlini/Wagner, CBFGS, JSMA and Deepfool. Each searches for the minimal amount of perturbation required to cause a misclassification or result in an agent switching from its most preferred action to its least preferred action. For example, FGSM (Goodfellow et al., 2014) computes a perturbation by taking the sign of the gradient of the cost function with respect to the model's input to create a new image that maximizes the loss.

A couple of specific attack styles exist which further exploit this gradient in the context of RL:

- Enchanting: the agent is directionally 'lured' towards a specific target state through a series of actions (Lin et al., 2017).
- Naive: where noise is constantly fed into the image, distorting the agent's actions away from the most preferred action.

A key consideration in an adversarial attack is that the malicious input is designed specifically to cause a model to move off-distribution and cause an agent to react unusually. One metric to measure and contrast different attacks on this front is to minimize perturbation effort, that is, to ensure that inputs are perturbed minimally or are otherwise difficult to detect by human supervision. These attack styles can have their effort throttled in several ways:

- Strategically-timed: detection is minimized by minimizing the number of frames that the perturbation is applied to (Lin et al., 2017).
- Fractional-state adversary (FSA): the adversary perturbs only a small fraction of the input state as opposed to the whole (Qu et al., 2019).

4.1.2 End-to-End Attacks

Our second class of attacks involves two subtypes based on the training and development of an agent specifically for the generation of adversarial examples.

For a dual-agent game, such as Kick and Defend or You Shall Not Pass (Gleave et al., 2019), it is possible to train an adversary to push the victim's policy network off-distribution with less than 3% as many timesteps. In this method, the victim is modelled as playing against an opponent in a two-player Markov game. The adversary is given unlimited black-box access to actions sampled from a victim's fixed stochastic policy. This reduces the adversarial agent's problem to a single-player MDP that the attacker must solve. This simplifies to an RL problem where you maximize an adversarial policy with well-defined state and action spaces, and reward functions based on the victim's fixed policy (Gleave et al., 2019).

Similarly, for a single-agent problem such as building adversarial attacks for text classifiers (Vijayaraghavan & Roy, 2019) it is possible to construct an adversarial agent that will attempt to maximize its adversarial reward over a series of iterations. This adversarial reward can be constructed to maximize not only misclassification or inaccurate response probability by the target agent, but also to minimize differences from expected inputs in order to reduce the likelihood of adversarial perturbations being detected. In the example provided, the adversarial agent for text classifiers was trained based on a rich, three-tiered adversarial award which considered overall misclassification probability, lexical similarity and semantic similarity - thus combining our metrics of maximising misclassification while minimizing perturbation effort.

4.2 Performance Evaluation

Our research also provides the basis for a framework for benchmarking and evaluating the performance of different adversarial attack strategies, which we will apply to specific

case studies. Our interests are in providing a rigorous and holistic approach towards evaluating the performance of a combination of attacks and RL agent policies, which we are testing through various Atari games such as Breakout and Pong - some of which are more susceptible or less susceptible to certain types of attack. In particular, the two classes of metrics used to measure performance are adversarial reward and perturbation effort.

4.2.1 Adversarial Reward

The attack attempts to maximize adversarial reward—which, dependent on the problem, can be per episode or cumulative—and is generally calculated as a measure of likelihood of leading the target off-distribution.

4.2.2 Perturbation Effort

At the same time, the attack should also minimize the amount of effort expended to achieve this reward. This effort is characterized by the type of attack. For fractional-state adversary (FSA), this can be the fraction of the input altered; for strategically-timed attacks, this can be the number of frames altered; for enchanted attacks, effort can be measured by the time taken to reach a target state.

We formalize perturbation effort as the minimum amount of adversarial input perturbation required to achieve any given amount of adversarial reward, globally defined across any possible attack on a given agent. If i is the unperturbed input to the policy network, a_r is adversarial reward and perturbation used to generate said reward is p_r then for a given policy network of an agent $f(i)$ then we can define perturbation effort as follows. For any given agent, the goal of the adversarial agent is to thus find and utilize the appropriate adversarial attack such that they minimize perturbation effort.

$$\arg\min_{p_r} a_r = f(i + p_r) \tag{2}$$

This is important in comparing the efficacy of different adversarial attacks. When two different attacks are capable of generating malicious input and pushing an agent off-distribution, in comparing their efficacy, it is extremely useful to consider how efficiently they were able to achieve the same adversarial rewards.

4.2.3 Strategically-Timed Effort

For strategically-timed attacks, the RL agent observes a sequence of states $\{s_1, ..., s_2\}$ and instead of attacking or adding noise to every time-step, the attack instead selects a subset of optimal timesteps to strategically add noise to in order to attack the agent and push it off-distribution. This results in a mixed integer programming problem, which is generally difficult to solve. However, Lin et al. (2017) and other authors have suggested heuristic algorithms to break this problem down into two sub-problems: "when-to-attack" and "how-to-attack", which respectively deal with the subset of frames selected and the amount of noise added.

When evaluating performance for strategically-timed attacks, we thus naturally arrive at two distinct metrics which align with our framework. In particular, we are attempting to minimize the number of frames and amount of noise we need to add in order to achieve a desired movement off-distribution in the agent - both of which can be considered forms of perturbation effort.

4.2.4 Enchanting Effort

For an enchanting attack, the goal is to lure a deep RL agent from a given state s_t at time step t to reach a specific target state s_x after y time steps. In order to do so, the adversary must design a series of malicious inputs $s_{t+1} + \delta_{t+1}, ..., s_{t+y} + \delta_{t+y}$ such that the agent is "enchanted" into the the target state over time.

This can again be reduced into two specific sub-tasks - firstly, planning a sequence of sub-steps for reaching the target state from the starting state, and secondly, crafting an adversarial example to lure the agent towards taking the first action of the planned sequence Lin et al. (2017).

This once again reduces elegantly into metrics for perturbation effort. Namely, we can consider the number of steps required in order to bring the agent to the target state. By considering such a metric, we can readily test and compare the vulnerability of different forms of policy (DDPG, PPO, etc.) to an enchanting attack.

4.2.5 Naive Effort

For a naive attack, the goal is similar to an enchanting attack where luring a deep RL agent from a given state s_t at time step t to reach any state s_x after y time steps that is not the original given state. The malicious inputs are naively generated and there is no specific target state.

Naive attacks are an inelegant method, but if the desired target state is not a particular state but a category of classes or even just a state that is not the given state, then naive attacks can be low effort attack vectors.

4.2.6 End-to-End Effort

While we have shown how we can use our metrics for gradient-based attacks above, we can also use our framework to evaluate end-to-end approaches. For example, in the dual-agent setting of You Shall Not Pass (Gleave et al., 2019), the end-to-end attack approach attempts to iteratively

train an adversary in order to interfere with or inhibit the victim.

In this form, the perturbation effort can be calculated as the number of timesteps required to cause the victim to be pushed off-distribution and fail (for example, the timesteps required until it reaches failure). It has been shown by (Gleave et al., 2019) that it is possible to train an adversary via adversarial learning to push the victim's policy network off-distribution with less than 3% as many timesteps as a conventional adversary by leading to exploit unusual behaviours in RL agent policies.

5 EXPERIMENTS

In order to find a rich environment for us to evaluate a range of policies and attacks, we elected to use Atari games such as Pong and Breakout - many of which have previously been shown to be very susceptible to reinforcement learning agents through OpenAI Gym, etc.

In order to show the benefits of our framework, we will be exploring some specific case studies of attacks and agents where we apply and leverage our framework in order to formally and effectively evaluate and benchmark performance.

5.1 Code Repositories

For our case studies on adversarial attacks, in order to allow us to leverage and utilize a mixture of deep reinforcement learning agents trained on different policies effectively, we used a few different code repositories to train our agents for Atari games. This includes the RLMA codebase from Qu et al. (2019) for Breakout, which trains a reinforcement learning agent to play an OpenAI Gym (Brockman et al. 2016) implementation of Breakout at a superhuman level and then attempts to push this agent off-distribution via the minimal perturbation required. We also used the OpenAI Baselines module (Dhariwal et al., 2017) to train an Atari agent, which we then applied attacks to using the Cleverhans module of adversarial attacks (Papernot et al., 2018).

5.2 Agents

5.2.1 *Pong, OpenAI Baselines*

The Python OpenAI Baselines library is built atop OpenA gym and includes several pre-built reinforcement learnin model architectures that are ready to be trained. We use this library to train a Pong agent using a DQN for 1,000,00 time-steps (which took about 11 hours on a regular CPU The agent learns to play Pong incredibly well, averagin scores of about 20-1 against a more basic agent in 21-poi games.

We have previously found that Pong is particularly susceptible to a wide range of attacks - the below frames are from a strategically-timed, naive attack which adds noise in an attempt to push the agent off-distribution. Even with a well-trained model, it is possible to cause meaningful degradation in performance with only minor perturbation effort, which we were able to analyse via merging Cleverhans attacks on our OpenAI-Baselines implementation of Pong.

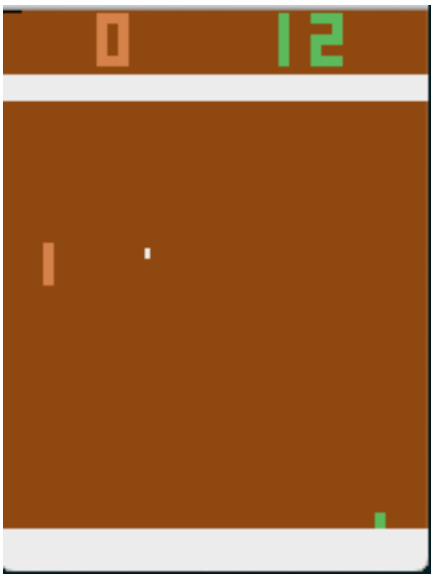

Figure 1. Pong Agent trained for 1,000,000 time-steps using OpenAI Baselines

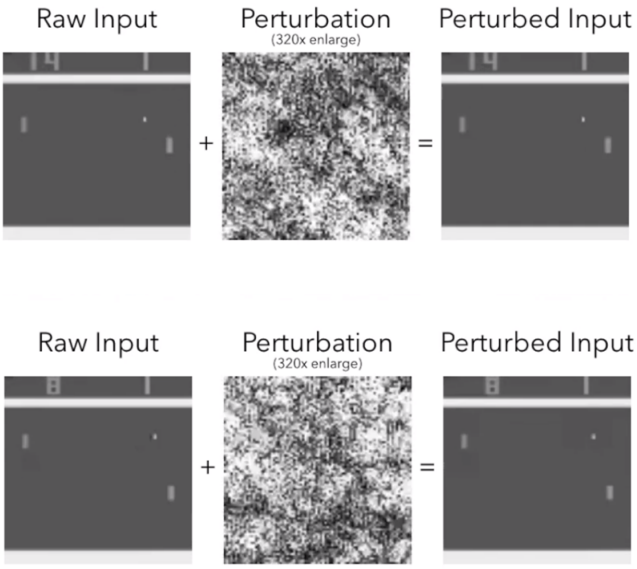

Figure 2. Pong, Naive Gradient-Based Attack (Qu et al., 2019)

5.2.2 *Breakout, PPO Attack*

Using the RLMA repository focused on implementing minimalistic adversarial attacks to push agents off-distribution, we trained and evaluated the performance of an RL-space adversarial attack on an agent designed to play the Atari game Breakout using our framework. The policy used for the agent was Proximal Policy Optimization, and the designed attack was an FGSM gradient-based attack implemented using a Fractional-state adversary.

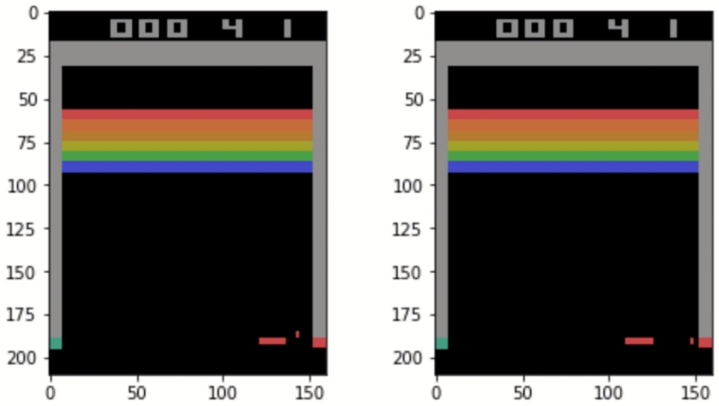

Figure 3. PPO FSA Attack on Breakout RL Agent

5.2.3 *End to End Text Classifier*

It is also possible to generate end-to-end attacks, although due to the substantially higher compute resources and time required this was outside the scope of our project. However, we are able to show that these results for end-to-end learning agents also fit within our framework. Prevoiusly, Vijayaraghavan & Roy (2019) have shown that it is possible to achieve markedly higher misclassification probabilities and thus adversarial rewards relative to other attack strategies while maintaining a black-box approach without access to model parameters or gradients.

Models	IMDB (CNN-Word)	AG's News (CNN-Char)
Random	2.46%	9.64%
NMT-BT	25.38%	22.45%
DeepWordBug	68.73%	65.80%
No-RL (Ours)	38.05%	33.58%
AEG (Ours)	**79.43%**	**72.16%**

Figure 4. PPO FSA Attack on Breakout RL Agent

By using a self-critical method to ensure lexical and semantic similarities, perturbation effort is also minimized relative to more traditional methods of generating adversarial examples

6 RESULTS

We have published code implementations where we have merged together existing code-bases of attacks and trained reinforcement learning agents such as Cleverhans and OpenAI Baselines on our Github repository (including neural network weights learned by training the agents using the existing implementations ourselves). It is possible to use our framework of attacks in conjunction with other results, such as those published by Huang et al. (2017), to form the foundations for a benchmarking suite for attacks on reinforcement learning agents which can be generalized to build a taxonomy for and evaluate any type of attack in RL-space.

As an example of how we can draw comparisons to evaluate different adversarial attacks, the first bucket of attacks under our framework are gradient-based attacks. Figure 2 shows the results of attacks on Pong and Space Invaders agents which were trained under 3 different algorithms: A3C, TRPO, and DQN. The results are graphed as the average reward of the agent against ϵ, which represents the "level of effort" to create the FSGM perturbation. Three differently constrained FGSM attacks were used: the blue represents an attack with a minimal constraint, yellow a medium constraint, and purple a maximal constraint. We

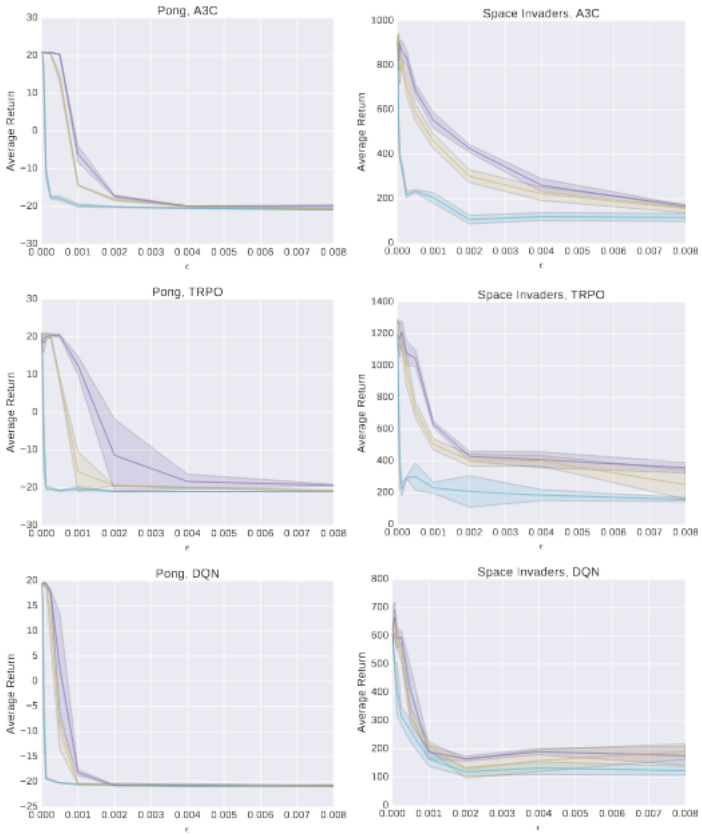

Figure 5. FSGM attacks on Pong and Space Invaders Agents with constraints on perturbation (Huang et al., 2017)

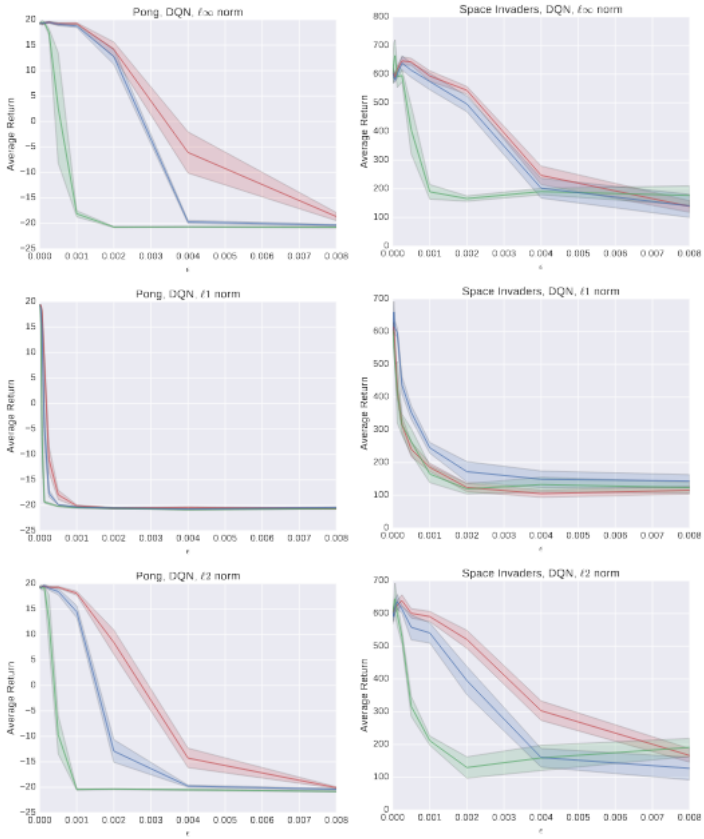

Figure 6. FSGM attacks on Pong and Space Invaders Agents with constraints on perturbation (Huang et al., 2017)

see from these results that the DQN algorithm is particularly susceptible to adversarial attacks as its rewards decrease the most with the least effort.

Often, when dealing with a model that is deployed in the real world, adversaries do not necessarily have access to the underlying architecture of the model. Therefore, it is likely that gradient-based methods will be applied as black-box attacks. In these situations, the adversary may have varying levels of information on the structure of the underlying model. For example, the adversary may have information on the model architecture, but not its initialization. Or, the adversary may have no knowledge of the model architecture and only be armed with an attack generated using some other model. In particular, investigating the susceptibility of the DQN further (given that it is the most susceptible to the basic attacks) shows that it is also quite vulnerable to transfer policies constructed using the medium constraint on FGSM (figure 3; l_∞, l_1, and l_2 represent the minimal, medium, and maximal constraints, respectively; the green line represents the white-box attack, the blue represents an attack with knowledge of the model architecture, and the red represents an attack with zero knowledge).

7 Discussion and Further Work

The space of adversarial learning when applied to reinforcement learning is a nascent field that has only begun to be developed in the last one or two years. As a result, individual papers focus on individual, ad-hoc and specialized attacks but there is little in the ways of a framework or overarching taxonomy in order to compare and evaluate attack performance across RL-space. While libraries such as Cleverhans are a good start at providing example implementations for various attacks and agents, it does not actively evaluate performance of these attacks they are not well documented and lack clear APIs. Other code repositories are few and far between and those that are publicly available, such as RLMA, have even worse documentation and are often incomplete, defunct or otherwise poorly-maintained. Creating a single framework of gradient-based and end-to-end methods which can be applied as white-box or black-box attacks, which could seamlessly integrate with existing state-of-the-art model implementations (such as OpenAI baselines or Keras-rl) and reinforcement learning environments (such as OpenAI Gym) would go a very long way to accelerating the pace of research and testing for ro-

bustness. This can be achieved through the unifying and bridging together of fragmented code repositories such as the ones which we have referenced and included within our Github repository, and begun to bridge together. This can allow us to achieve real gains in improving the robustness of RL agents to these attacks - which have repeatedly been shown to be highly susceptible to even minimal perturbation efforts when these have been carefully designed via enchanting, strategically-timed or other throttling efforts (Pattanaik et al., 2017) (Gleave et al., 2019).

Additionally, there are more considerations that can be made in terms of hardware. Clarification and standardization of the hardware that is being used in experiments is incredibly important in any benchmarking suite. An essential component of analysis that is missing from almost every paper on adversarial deep learning is the demands that attacks make on the underlying hardware. When considering efficiency tradeoffs, are there attacks which consume more power? In particular, for adversarial reinforcement learning, does one class of attack make more demands on computing power than another (for example, a strategically-time vs. enchanting attack)? This kind of practical analysis brings us closer to the real-world applicability of attacks. Any complete benchmarking suite must evaluate models on which attacks are most likely to be applied in the real world. Intuitively, one may reason that black-box attacks will be more common as it is rare to have prior knowledge of the underlying trained model.

8 Conclusion

Our hope is that this work can be used as a guide for future research in classifying and evaluating adversarial reinforcement learning. We have highlighted state-of-the-art types of attacks, created a taxonomy within which all known adversarial reinforcement learning techniques can be classified, and compiled a substantial body of related work which can serve as an introduction to the area. We have also discussed key performance characteristics of effective adversarial attacks and evaluation methods for different throttling methods when these attacks are applied to RL-space and shown how existing results fit into our framework. The field of adversarial reinforcement learning is very new, but quickly growing. While many papers have been published which show experimental results on various agents, there has been no unified way to understand the field. This understanding can be improved through better code implementations and expanding performance analyses to be able to more effectively understand the vulnerabilities of different agents and policies to different styles of attacks, such that we can also begin to defend against them more effectively when designing agents in future.

Acknowledgements

We would like to thank the CS249r course staff—Professor Vijay Janapa Reddi, Brian Plancher, and Shruti Mishra—for all of their incredibly helpful advice and support. We also additionally thank the engineers at Robust Intelligence for helping guide the direction and focus of our research into adversarial learning.

References

Behzadan, V. and Munir, A. Vulnerability of deep reinforcement learning to policy induction attacks. *CoRR*, abs/1701.04143, 2017. URL http://arxiv.org/abs/1701.04143.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Carlini, N. and Wagner, D. A. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016. URL http://arxiv.org/abs/1608.04644.

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., and Zhokhov, P. Openai baselines. https://github.com/openai/baselines, 2017.

Gleave, A., Dennis, M., Kant, N., Wild, C., Levine, S., and Russell, S. Adversarial policies: Attacking deep reinforcement learning. *CoRR*, abs/1905.10615, 2019. URL http://arxiv.org/abs/1905.10615.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples, 2014.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network, 2015.

Huang, S. H., Papernot, N., Goodfellow, I. J., Duan, Y., and Abbeel, P. Adversarial attacks on neural network policies. *CoRR*, abs/1702.02284, 2017. URL http://arxiv.org/abs/1702.02284.

Lin, Y., Hong, Z., Liao, Y., Shih, M., Liu, M., and Sun, M. Tactics of adversarial attack on deep reinforcement learning agents. *CoRR*, abs/1703.06748, 2017. URL http://arxiv.org/abs/1703.06748.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks, 2017.

Papernot, N., McDaniel, P. D., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. *CoRR*, abs/1511.07528, 2015a. URL http://arxiv.org/abs/1511.07528.

Papernot, N., McDaniel, P. D., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. *CoRR*, abs/1511.04508, 2015b. URL http://arxiv.org/abs/1511.04508.

Papernot, N., McDaniel, P. D., Goodfellow, I. J., Jha, S., Celik, Z. B., and Swami, A. Practical black-box attacks against deep learning systems using adversarial examples. *CoRR*, abs/1602.02697, 2016. URL http://arxiv.org/abs/1602.02697.

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

Pattanaik, A., Tang, Z., Liu, S., Bommannan, G., and Chowdhary, G. Robust deep reinforcement learning with adversarial attacks. *CoRR*, abs/1712.03632, 2017. URL http://arxiv.org/abs/1712.03632.

Qu, X., Sun, Z., Ong, Y.-S., Wei, P., and Gupta, A. Minimalistic attacks: How little it takes to fool a deep reinforcement learning policy, 2019.

Vijayaraghavan, P. and Roy, D. Generating black-box adversarial examples for text classifiers using a deep reinforced model, 2019.