Attacking Reinforcement Learning: A Survey on Methods
CS855 Project

Laura Graves
laura.graves@uwaterloo.ca

Abstract

Neural networks are vulnerable to **adversarial examples** that have been crafted to cause incorrect predictions. Reinforcement learning systems that implement deep learning are vulnerable to the same examples. In this paper, we look at the current state of adversarial attacks against reinforcement learning systems. Beginning with an explanation of adversarial attacks and current techniques, we move to an analysis of attacks against reinforcement learning systems showing where systems are vulnerable, what kind of defenses have been proposed, and what the future of this domain may look like.

1 Introduction

In the field of machine learning, adversarial examples are specifically designed examples that are semantically different from the classification provided by a trained ML model. For example, if we have a picture of a goat that we have modified so that our neural network classifies it as a sombrero, it is an adversarial example. These examples not only present a problem for neural network model holders concerned with model classification accuracy, but also highlight the shortcomings of neural networks to semantically understand the images they're given.

As deep learning becomes a bigger and bigger part of reinforcement learning (RL) algorithms, the same problems that plague neural network models also pose a problem for RL. If a party is able to shape the input to an algorithm in an adversarial manner - that is, provide a specifically designed input that causes incorrect behavior from the neural network - they are able to shape the learning or behavior of the algorithm in a manner they want.

In this survey, I will analyze state-of-the-art attacks and defenses against RL systems. Most of these methods rely on research for adversarial attacks against deep learning systems, so I'll provide the background necessary to understand these common attack algorithms and RL systems. Further, I'll analyze other security and privacy risks that neural networks face, and attempt to find ways that they could be used to exploit RL algorithms.

2 Background

Attacks on deep RL systems usually combine the fields of neural network security and deep reinforcement learning. In order to understand how and why these attacks are successful, we must understand how the attacks work against neural networks, as well as how neural networks are implemented in RL systems.

2.1 Adversarial Attacks & Defenses

Adversarial examples first appeared in literature from Szegedy et al. [21], where they noticed that there appeared to be examples that are extremely similar to other examples, yet cause a different classification. Soon afterward Goodfellow et al. [6] showed that these examples (deemed *adversarial examples*) can be found easily using gradient ascent. The attack they introduced - the **fast gradient sign method** (FGSM) calculates the gradient of an example x with label y with regard to the loss J of the network with parameters

θ, $\Delta_x J(\theta, x, y)$. A step in the direction of that gradient with magnitude ϵ is taken, and the resulting example $x_{adv} = x + \epsilon sign \Delta_x J(\theta, x, y)$ is likely to be adversarial. Note that this is a step in the direction of increased loss.

The authors also proposed the defense technique of *adversarial training*, where a neural network is trained for some iterations, and then a set of adversarial examples are created and added to the set of training examples. Over time, the resulting network becomes robust to adversarial attacks, learning more robust decision boundaries around the training data.

In the years since, many other attacks and defenses have been proposed. One important attack is the Jacobian-based Saliency Map Attack [17] which utilize saliency maps (originally designed to help interpret model classifications) to perform efficient attacks with small ℓ_0 bounds. More recently, the Carlini & Wagner Attack [4] was designed to find the smallest perturbation that will cause the model classification to change.

One important consideration is that adversarial examples are not an accident of learning - they are inherently a part of deep learning and are in fact inevitable [20]. In consideration of this, most defense techniques do not attempt to try to avoid the existence of adversarial examples altogether, but simply try to make them as difficult to compute as possible. Many of these defenses are based on obfuscated gradients, but these have been shown to be ineffective [1]. In fact, the property of transferability (that models trained on similar data tend to have similar decision boundaries [15], and thus will make similar predictions on similar inputs) means that black-box attacks where the attacker is given only enough access to input examples and get the resulting prediction vectors are effective. These are done via training a substitute model and creating adversarial examples for that model, and those examples are likely to be adversarial for the black-box model [16]. In some cases these attacks are effective against models that return only truncated prediction vectors or even models that return only the maximum probability label [14]).

2.2 Deep Reinforcement Learning

Deep learning systems are frequently used as components of RL systems. Neural networks are excellent at function approximation, so any RL system that has a function that needs to be updated through a learning process is an excellent candidate for deep learning.

The Deep Q-Network (DQN), a Q-Learning algorithm that uses a neural network as a Q-function, was the first implementation of a deep RL system [12]. The system was applied to Atari games, and replaced the Q function (a measure of the value of taking a certain action at a certain state) with a deep neural network.

Another deep RL system that appears in these works is the Asyncronous Advantage Actor-Critic algorithm (A3C) [11], which uses the actor-critic idea and leverages asynchronous training to both increase performance and improve training time. This system implements multiple neural networks, providing plenty of opportunities for attackers.

3 How Vulnerabilities Are Exploited

Almost all vulnerabilities come from the root cause of the failure of neural networks to semantically understand their input. As covered previously, adversarial examples are inputs that are semantically similar, yet cause significantly different predictions. Neural networks are inherently vulnerable to adversarial example [20], and RL systems that use neural networks as a component are not exempt from this vulnerability. As a consequence of this property, any RL system with a neural network as a component can be attacked in similar ways to standard neural networks. We must change the threat model and our goals to suit the changed attack environment (causing a misclassification is no longer a goal of ours, because classification isn't the purpose of these networks).

One of the first decisions we have to make is what the threat model is. Our tactics will change based on what access we have to the RL system and during what time. If we have white-box access during model training, we must use vastly different tactics than if we have black-box access after the system is trained. To this end, we define the following terms:

White-box Access: The attacker has access to the entire RL system, including all parameters.

Black-box Access: The attack is able to run the algorithm on whatever input they choose and see the resulting output, but cannot see any of the parameters or any other aspects of the system.

Training-Time Access: The attacker is able to choose what inputs the system is given as it is learning in order to effect the learning, either by modifying the sensor values/environment or by directly giving input values.

Inference-Time Access: The attacker is able to choose what inputs a trained system is given in order to effect the behavior of the system, either by modifying the sensor values/environment or by directly giving input values.

The combination of accesses the attacker has plays a significant role in what they are able to accomplish.

3.1 Training-Time Attacks

During the training stage, an attacker with access to the system can manipulate the inputs to cause a desired change in the final trained system. These changes can be subtle (slightly changing the value function so some actions are preferred among others) to drastic (changing the entire learned model in a way that compromises use).

The system's **policy** or **value function** can be targeted directly by modifying the inputs given to the system during training. During training, the attacker can selectively give inputs that have been adversarially crafted to cause the neural network to give the wrong output, causing the system to select an incorrect action and get a different reward than normal. Throughout training time, the model learns an incorrect idea of what actions lead to the highest rewards. In this way the attacker can manipulate the training process so the system acts with whatever behavior that the attacker desires, instead of the optimal behavior.

3.2 Inference-Time Attacks

After the model has been trained, the attacker is able to influence the behavior of the system by manipulating the inputs. By slightly modifying the input to be adversarial, the attacker is able to force the system to choose whatever action the attacker desires instead of what action the system has learned is optimal. Interestingly, these attacks are extremely hard to prevent, as adversarial attacks are extremely hard to defend against, even in black-box settings [16].

It is important to note that not all attacks need to modify the inputs directly. If an attacker is able to modify the environment in a certain way, they can also cause improper behavior by modifying the environment in a way that alters the actions chosen by the system. An example of this is [5], where the authors show that in a two-agent competitive game, a malicious agent is able to choose their own actions in a way that is counter-intuitive to winning, yet causes the opponent's actions to be chosen so poorly that they lose the game. Real-world adversarial examples have been shown to exist as well, where a perturbation is crafted in a way that it can be applied directly on a real-world object in a manner that causes computer vision systems to make incorrect predictions on it [2]. The existence of these real-world examples give reason to believe that they could also be leveraged to cause incorrect behaviors from RL systems implemented in the real world, such as self-driving vehicles. This shows that attackers need not have direct access to the input of the system in order to attack it - they have the potential to attack the system simply by modifying the environment in a specific way.

4 Reinforcement Learning Attacks & Defenses

4.1 Training-Time Attacks

The first researchers to analyze the vulnerability of RL systems to adversarial attacks was Behzadan & Munir [3], who developed a system to compromise the policy of a deep RL system. They called their attack the *policy induction attack*, and it fundamentally works on the principle that, if at state s_t, the attacker is able to modify s'_t as well as modify s_{t+1}, where s'_t and s_{t+1} are adversarially crafted to cause the Q-function $\hat{Q}(s_{t+1}, a; \theta_t)$ to choose an incorrect action a' at s_{t+1}, they are able to guide the policy toward learning that a' is the correct choice of action from state s_t.

This novel and relatively simple method started the research area of attacking reinforcement learning methods. The authors not only showed that such an attack is possible, but they implemented their approach in white-box and black-box settings, where the attacker was given the type and format of inputs, the reward function R, and an approximation of how frequently the $\hat{Q}$ function is updated. Notably, the exact reward function is not needed, because an attacker is able to perform inverse reinforcement learning [13] to find an approximation of R. Additionally, an ℓ_2 perturbation bound ϵ was placed on the attacker, limiting the amount they could alter a state s_n to $||s'_n - s_n||_2 \leq \epsilon$. In this black-box environment the authors leverage the concept of *transferability* [15] to attack the system by training a separate system M' that's a simulacrum of the target system M. Adversarial examples against M' have a high probability of also being adversarial against M, and in this manner they are able to effectively attack the black-box deep RL system. The authors experimentally showed the efficacy of both the transferability and the black-box attack, and were able to limit the average reward per epoch of the attacked model to a small fraction of the average reward of the unattacked model. Shortly after, Huang et al. [7] published a robust evaluation of this same attack method, comparing performance of the attack against different tasks, training algorithms, and attack models (both white-box and black-box attack models). They were able to show that in all these settings, regardless of task or algorithm, a significant drop in performance is achievable even with extremely small perturbations (they authors compared effectiveness when attackers were limited based on either an ℓ_1, ℓ_2, or *ell* norm). Simultaneously, Kos & Song [8] published a similar paper, evaluating the same attack algorithm. They found that by measuring the expected reward and only inserting adversarial examples into the training process when the expected reward was high enough was nearly as effective as including adversarial examples at every frame, yet took a fraction of the computational effort. They theorized that if you only disrupt the learning process when it's close to achieving a reward, you are able to effectively stifle effective learning as a whole. They also tested the attack algorithm with two sequential adversarial examples inserted every 10 frames and found this was somewhat as effective as the standard algorithm with adversarial examples inserted every frame, although it wasn't able to hinder learning as effectively as disrupting high-reward frames.

One important note that was raised in [7] is that of defenses against these attacks. An extremely common defense, proposed in [6], is that of *adversarial training.* In this defense method a network is trained for some iterations on a dataset D. After this initial training, a set of adversarial examples A is created and added to the original dataset such that $D = D \cup A$. Some further training is performed on this new dataset, and this process of creating a set of adversarial examples, adding it to the dataset, and training for some further iterations is repeated, with the model becoming more robust to adversarial attacks with each iteration. The authors noted that a system like this could be used to make deep RL systems more robust against these attacks, as a significant amount of the presented attacks are performed against deep RL systems with undefended neural networks. However, the authors presented no application of such a defense. Kos & Song [8] presented the first application of this style of defense, making the models more robust by augmenting standard training with either examples modified with random noise or augmenting the training with examples created to be adversarial (using FGSM). They found that as long as the perturbation bound for adversarial training was not much smaller than the perturbation bound used by the attacker that the adversarial training was remarkably effective against attacks, limiting the attack success to a small fraction of what it was on an undefended model or against a model defended by a much

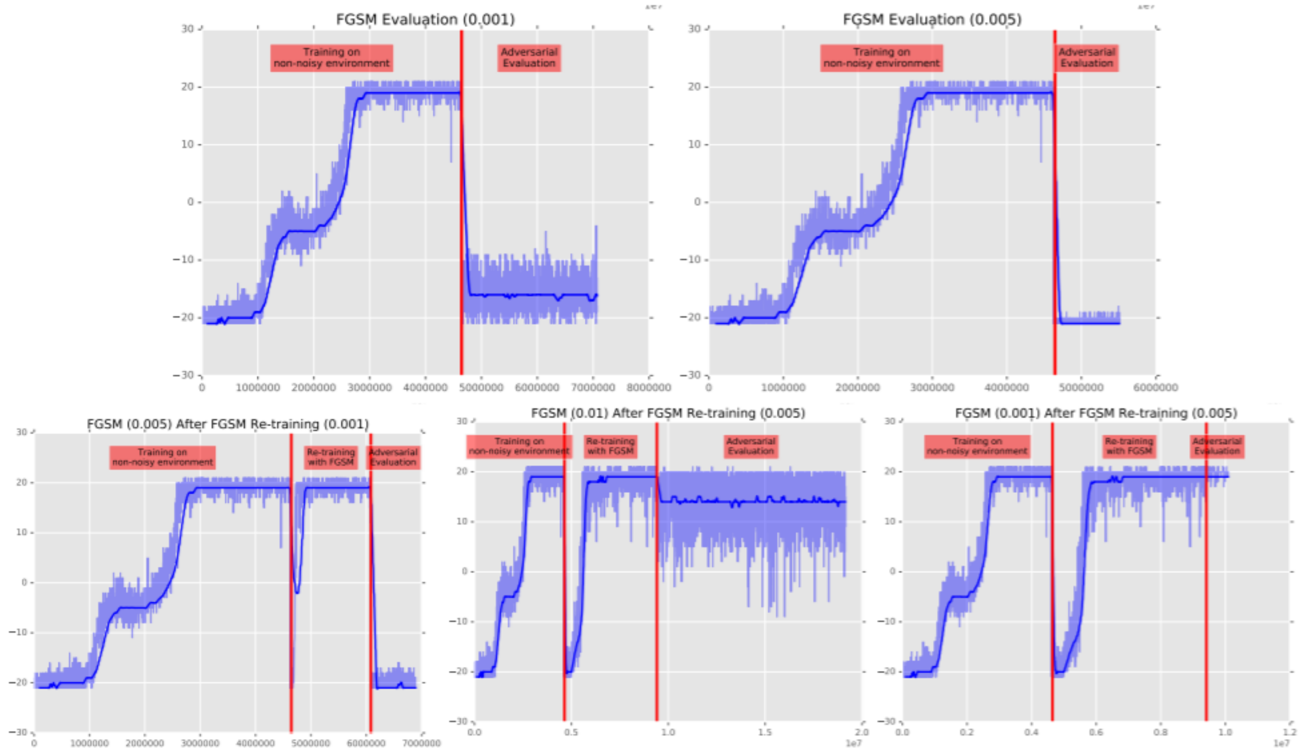


Figure 1: Top: FGSM attack against undefended model. Bottom: FGSM attack against models adversarially trained with FGSM

smaller perturbation bound (fig 1).

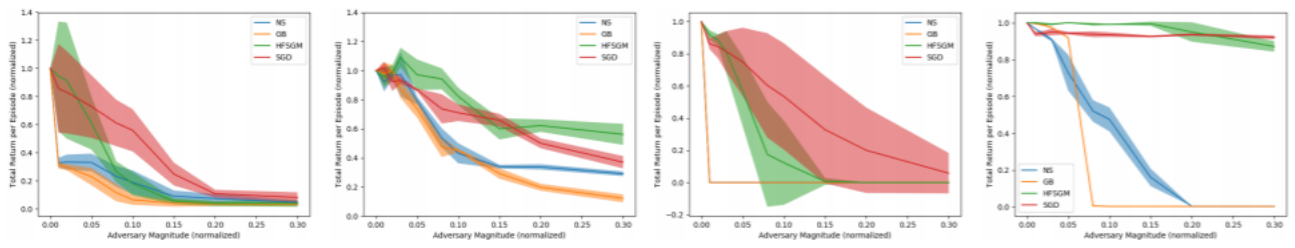
These defenses were evaluated much more fully by Pattanaik et al. [18]. Inspired by robust control systems which seek to find a best case policy over a set of worst possible parameters of a system, they introduced a novel approach of viewing training as a problem of finding the best possible policy with a present adversary. With this problem in mind, they trained a model with a present adversary using their own gradient-descent attack that is similar to FGSM, but where the magnitude of the step for each feature is given by sampling noise instead of taking a fixed step with magnitude ϵ, and instead of the gradient of loss, the gradient of the Q-function is used and is constrained on an ℓ_2 bound. That is, instead of calculating the adversarial example $x_{adv} = x + \epsilon sign\Delta_x J(\theta, x, y)$, they instead sample a noise vector $n_i \sim beta(\alpha,)$ and then calculate $x_{adv} = x - n_i \frac{\Delta_x Q^{target}(x,a)}{||\Delta_x Q^{target}(x,a)||}$. This approach not only resulted in improved robustness against adversarial attacks (fig 2), but resulted in a more robust model against signal noise and changes in simulation parameters such as weight and friction.

The authors surprisingly note that their naive strategy of sampling random noise to attempt to find adversarial examples (NS) outperforms both Huang et al.'s attack [7] and an SGD based attack. Their attack of choice, the Gradient Based attack, outperforms all others in the environments they tested.

4.2 Inference-Time Attacks

The research into inference-time attacks is much sparser than the research into training-time attacks, perhaps because directly influencing the policy is viewed as a more worthwhile goal than simply causing errant behavior on certain inputs. Nevertheless, there is some fascinating research that shows how easily systems can be compromised even after comprehensive and robust training.

Unlike training-time attacks, inference-time attacks choose to manipulate the environment or input in a way that makes the system take actions it wouldn't normally choose. This can take a variety of forms, such as manipulating the input directly with adversarial attacks or even having another agent in a multi-agent



(a) DDQN Cart Pole (b) RBF Q Cart Pole (c) DDQN Mountain Car (d) RBF Q mountain car

Figure 2: Comparison of attacks on DDQN and RBF based Q-learning systems in mountain cart and cart-pole problem environments.

system choose actions in a way that causes the target to choose incorrectly. Lin et al. [9] developed two inference-time attacks that achieve two different goals. First, the introduced the *strategically-timed attack*, which attacks the system in a similar way to Kos & Song [8], choosing to attack only when doing so is optimal. Instead of choosing to attack only when the reward is high enough as in Kos & Song, they choose to attack only when the *relative reward*, or the difference between the maximum reward and minimum reward, is high enough. This helps the attacker prioritize when to manipulate the input to only those instances where interference can make a significant difference. As an example, they give the scenario of an agent trained to play the video game Pong. When the paddle is far from the ball the agent has virtually no preference in terms of action selection, so causing an errant action has little effect. Conversely, when the ball is close to and approaching the paddle, choosing to move the paddle in the wrong direction has a significant impact on the outcome. The Strategically-Timed Attack would prioritize the latter scenario over the former, increasing the probability that an attacker can be effective even when they only manipulate a small number of inputs. When tested against a variety of simulations such as Pong, Seaquest, or Mrs. Pacman, the attacker was able to lower the reward significantly while effecting as little as 1/20th of the frames.

The second attack they propose is the *enchanting attack*, which combines future state prediction and adversarial attacks to lure an agent at state s_t at time step t to a target state s_g. They attacker is given H time steps to achieve this goal, and the problem becomes that of creating a sequence of adversarial examples $s'_{t+1}, s'_{t+2}, ..., s'_{t+H}$ such that the agent is lured to state s_g. They approach this problem by considering two sub-problems: a future state prediction problem and an action planning problem. In the future state prediction problem they use a generative model to predict future states given a state or series of states. Once this is done, they are able to use the generative model to find a sequence of actions that lead the agent as close as possible to s_g. This action planning process results in a series of actions, and from there they are able to use a standard adversarial attack system to find adversarial perturbations for each state along the planning path so that the desired action is chosen. When tested against a variety of simulations such as Pong, Seaquest, or Mrs. Pacman, the attacker (given an H value of less than 40) was able to reach the goal state in 70% of simulations. With an H value of 120, the attacker was able to reach the goal state against almost every simulation and agent.

Gleave et al. [5] has perhaps the most interesting inference-time attack algorithm. They analyze a multi-agent system where humanoid agents play a zero-sum game against each other, and the games range from simulated sumo wrestling to simulated soccer. In this application the attacker plays one agent and the victim plays another, but the attacker is not given the ability to effect the victim's input in any way. Instead, the attacker must manipulate the environment in any way they can in order to make the victim choose sub-optimal choices. They note that the attacker wins by creating natural adversarial observations, instead of winning through good action selection, and they establish the validity of this through testing the attacker against an opponent who can't observe the attacker's position. The attacker acts in an unexpected manner, contorting themselves into strange positions and falling to the ground instead of walking, kicking, or attempting to block their opponent.

These attacks, while executed in an unusual way, are computationally very similar to others we've seen. The attacker is given black-box access to actions selected from the victim's policy π_v, but is not given any other access. The attacker must then solve the MDP $M_\alpha = T(S, A_\alpha, T_\alpha, R'_\alpha)$, where the transition and reward function depend on both the attacker and victim policies in the following way:

$$T_\alpha(s, a_\alpha) = T(s, a_\alpha, a_v) \qquad\qquad R'_\alpha(s, a_\alpha, s') = R_\alpha(s, a_\alpha, a_v, s')$$

The attacker is then able to find an adversarial policy π_α that maximizes the sum of discounted rewards. The authors solved this problem using Proximal Policy Optimization [19]. The end result is an adversary that chooses actions designed to force the opponent to choose poorly, despite being very unintuitive - for example, in the 'kick and defend' game, the attacker never stands up and instead contorts their body to induce an adversarial environment that causes the opponent to choose poorly. When viewing t-SNE activations caused by standard opponents and adversarial opponents, they showed that the adversarial observations created by the attacker result in a set of activations that is largely separated from the activations cause by a standard opponent.

5 What the Future May Hold

Considering the ubiquity of deep RL systems, I feel there is a large space for future work in this area. Most of the attack and defense techniques are fairly rudimentary compared to the state-of-the-art in robust deep learning research, and I believe most of these techniques could benefit from a more robust application of modern techniques. Additionally, these attacks could be applied to real-life systems that use deep RL - for example, the Gleave et al. attack [5] could be used against systems such as Alphastar [22] that use deep RL to play the video game Starcraft. Human experimentation has already shown real-life examples of this sort of attack exist, where the human adversary can take actions such as repeatedly retreating and advancing the same squad of units, causing the RL agent to get stuck in a loop of chasing the squad and retreating once it gets sufficiently far from it's base. It could be interesting to see how these opponents fare against an adversary that has no motivation but to cause the opponent to choose as poorly as possible. There could be additional research value here because Starcraft is a game of perfect but not complete information and the attacker must be able to adversarially effect the opponent without knowing their current state.

I think another fascinating potential research direction would be that of data poisoning attacks. The research surveyed here has shown that an attacker can easily influence the policy of a deep RL system. If the attacker is able to select inputs, they are likely able to train the system on a series of regular observations as well as watermarked observations, in a similar method to Liu et al. [10]. If a system is trained in this way, it can be deployed and act perfectly fine for any length of time without detection, but when the time is right an attacker could insert their watermark and cause the desired harmful action. I believe a system like this could be easily implemented in domains that rely on vision information such as video game agents that rely on frame data.

6 Conclusion

Deep RL systems are remarkably susceptible to attacks that target their neural network, either to detrimentally effect the policy or to cause the agent to take poor actions during inference. Although most of the research in this area is focused on toy examples in easy domains, they serve as a proof of concept that should be worrying to those implementing systems in safety-critical or other high-risk areas. Further research is necessary to protect systems and their owners, but also to find out the applicability of these attacks to real-world systems.

References

[1] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

[2] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293, 2018.

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

[4] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[5] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

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

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

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

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

[10] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang. Trojaning attack on neural networks. 2017.

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

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[13] A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[14] T. Orekondy, B. Schiele, and M. Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4954–4963, 2019.

[15] N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[16] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[17] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[18] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary. Robust deep reinforcement learning with adversarial attacks. *arXiv preprint arXiv:1712.03632*, 2017.

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

[20] A. Shafahi, W. R. Huang, C. Studer, S. Feizi, and T. Goldstein. Are adversarial examples inevitable? *arXiv preprint arXiv:1809.02104*, 2018.

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

[22] O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu, M. Jaderberg, W. M. Czarnecki, A. Dudzik, A. Huang, P. Georgiev, R. Powell, et al. Alphastar: Mastering the real-time strategy game starcraft ii. *DeepMind blog*, page 2, 2019.