A study of natural robustness of deep reinforcement learning algorithms towards adversarial perturbations

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deep reinforcement learning (DRL) has been shown to have numerous potential applications in the real world. However, DRL algorithms are still extremely sensitive to noise and adversarial perturbations, hence inhibiting the deployment of RL in many real-life applications. Analyzing the robustness of DRL algorithms to adversarial attacks is an important prerequisite to enabling the widespread adoption of DRL algorithms. Common perturbations on DRL frameworks during test time include perturbations to the observation and the action channel. Compared with observation channel attacks, action channel attacks are less studied; hence, few comparisons exist that compare the effectiveness of these attacks in DRL literature. In this work, we examined the effectiveness of these two paradigms of attacks on common DRL algorithms and studied the natural robustness of DRL algorithms towards various adversarial attacks in hopes of gaining insights into the individual response of each type of algorithm under different attack conditions.

1 Introduction

Deep reinforcement learning (DRL) has seen substantial successes in multiple domains of applications such as design [1], scheduling [2] and robotic control applications in industrial automation [3]. Contrary to supervised learning, RL algorithms train an agent learns to perform a given task in an environment by making sequential actions and observing the resulting rewards to learn an optimal policy. In recent years, advancements in neural networks have led to the popularity of DRL, where a deep neural network represents the RL policy. Although neural networks are powerful function approximators, they are also extremely easy to fool into making erroneous predictions by applying perturbation on the model's inputs [4]. This observation led to numerous studies on the robustness of deep learning algorithms. A study by [5] proved that similar adversarial attacks could also be extended to manipulate RL agents where the RL agent is vulnerable to subtle adversarial attacks that are not perceivable to humans but could cause a significant change in RL policy's actions. Subsequently, this has led to the development of several other successful adversarial attacks [6, 7, 8, 9, 10].

While numerous works have developed DRL algorithms that are robust towards different perturbations [11, 12, 13, 14], to the best of our knowledge, a study that compares the response of popular benchmark DRL algorithms towards common adversarial perturbations is still lacking in the literature and this work aims to fill in such a gap. Specifically, in this work, we analyze the performance of multiple DRL algorithms commonly used in literature when subjected to observation and action perturbations. As a first step, we restrict our experiments to continuous state-action environments, which provide a more realistic proxy to industrial robotic applications, where adversarial attacks are of greater concern. Our experiments aim to answer the following questions: **(1)** Are the existing DRL algorithms especially sensitive to one class of adversarial perturbations over the other? (e.g.,

Submitted to Deep Reinforcement Learning Workshop at the 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

observation vs. action space perturbations), **(2)** Is there a specific DRL algorithm that is naturally more robust than all other algorithms under adversarial perturbations and **(3)** Is there a limitation of the magnitude of the perturbation on the degradation of the DRL performance, i.e., is there an empirically observable threshold in which perturbations below or above this threshold will not affect the behavior of the DRL policy? Overall, our experiments serve to provide insight into existing DRL algorithms' potential natural robustness and act as a stepping stone to developing more robust DRL algorithms.

2 Related Works

Adversarial attacks on deep neural networks were first popularized by Szedegy et al. [15], who performed adversarial attacks on image classification algorithms by adding perturbations to the input images. These attacks aim to trick the model into misclassifying images. Generally, adversarial attacks can be divided into white-, gray- and black-box attacks, depending on the amount of knowledge the adversary has about the machine learning (ML) model it is targeting [16]. In white-box attacks, the adversary has complete knowledge about the target ML model, such as the learned weights, training parameters, and training and testing data. With that information, these attacks can be used to analyze worst-case scenarios [17]. In black-box attacks, the adversary has no knowledge of the model or any of its parameters. Hence, black-box attacks are often established based on the model inputs, confidence scores, or perturbing the feedback of the ML model [18]. Meanwhile, gray-box attacks fall in between the spectrum of white- and black-box attacks, with the adversary having partial knowledge of the ML model being targeted. In the context of adversarial attacks on DRL, in [6], the authors showed that Deep Q-Networks (DQN) are vulnerable to adversarial state perturbations. The adversarial perturbations were generated using the Fast Gradient Sign Method (FGSM) and Jacobian-based Saliency Map Attack (JSMA) [19]. Additionally, they also implemented a black-box and showed a success rate of 70%. In [5], the authors employed similar attacks as in [6] but implemented the attacks on other DRL algorithms such as DQN, Trust Region Policy Optimization (TRPO), and Asynchronous Advantage Actor-Critic (A3C) methods in both white and black-box settings. Their result demonstrates that DQN is more susceptible to adversarial attacks than others. Recent works have also demonstrated that observation-based adversarial perturbations can manifest themselves in a multi-agent RL system where a corrupted agent can behave adversarially to fool the other RL agent [20]. Besides perturbations in the observation space, the attacks can also occur in the action space in the form of perturbations on the actuators. For example, Lee et al. [10] proposed coupled spatial-temporal action space attacks that can reveal the potential vulnerabilities of the DRL model. Additionally, action space perturbations can also manifest in the form of environmental noise or changes in environmental factors [11]. For brevity, we refer interested readers to the more detailed and complete taxonomy of adversarial attacks presented in [21].

3 Methodology

In this section, we provide a brief description of the experiments we conducted to compare the performance and response of common DRL algorithms to adversarial attacks.

3.1 Selection of environments and algorithms

We conducted our experiments on five different continuous control environments based on OpenAI Gym [22] MuJoCo environments. The five selected environments are: i) Ant, ii) HalfCheetah, iii) Swimmer, iv) Walker, and v) Hopper. To facilitate a more accurate comparison, all experiments were run with six random seeds, and for each seed, we ran 100 episodes and reported the average score across all episodes and seeds. In terms of selecting the DRL algorithms to compare, we selected five popular algorithms that are commonly used as benchmarks in continuous control tasks: 1) Proximal Policy Optimization (PPO) [23], 2) Deep Deterministic Policy Gradient (DDPG) [24], 3) Trust Region Policy Optimization (TRPO) [25], 4) Twin Delayed DDPG (TD3) [26], and 5) Soft Actor-Critic (SAC) [27].

3.2 Black-box attacks

Next, we describe the suite of black-box attacks that we implemented as part of our experiments to compare the performance of the DRL agents. As an initial step, we limit the scope of black-box attacks in this paper to simple additive perturbations. However, we highlight that black-box attack strategies extend beyond naive additive perturbations and will be a key focus of future studies. To fully investigate the behavior of the policies in a comprehensive manner, we develop multiple strategies for black-box attacks. These strategies were generated by identifying the three stages where the perturbations can be performed. The first stage consists of the channel of perturbation, where the perturbation can either be added to the observations of the agent or the actions of the agent. The second stage involves the magnitude of perturbation, where the magnitude of perturbation is either random, bounded by the action space, or bounded by the magnitude of the actual action taken by the agent. The third stage involves the direction in which the perturbation is applied to. Since the actions and observations in these environments are multidimensional vectors, the perturbations can be added in four different ways: 1) consistently adding noise following the signs of individual observations/actions, 2) consistently adding noise that is opposite the sign of the individual observations/actions, 3) randomly selecting a direction for each time step and adding the perturbation according to the direction and 4) randomly selecting a direction for each time step and each observation or action and adding the perturbation according to the direction. The suite of all possible black-box attacks can be summarized according to Fig. 1, where selecting a choice at each stage will result in a valid strategy.

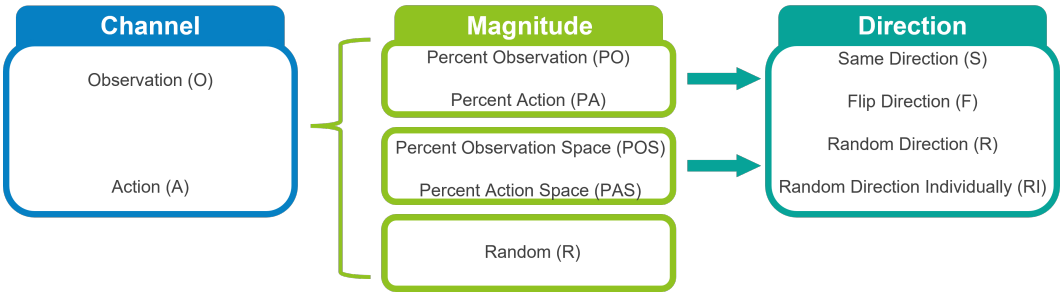


Figure 1: Black-box attack strategies: The flowchart shows the various black-box strategies implemented. The attacks can be mounted on either one of the two channels, with the constraint on the attack following one of the three magnitudes and the specific instantiation following one of the four directions. The names of each attack are denoted by the abbreviation from each stage, e,g: attacks on the observation with constraint from percent of observation with the same direction denotes O_PO_S.

3.3 White-box attacks

Next, we describe the three white-box attack strategies that we selected to test on the common benchmark RL algorithms. All three attacks leverage the gradient information to craft the attacks and these three attack strategies were selected to study both perturbations on the observation and action channels. Specifically, the white-box attacks we implemented are the Fast Gradient Sign Method (FGSM) [4], Projected Gradient Descent (PGD) algorithm [28] and the Myopic Action Space (MAS) attack algorithm [10].

Fast Gradient Sign Method: FGSM generates a perturbation by taking the sign of the loss's gradient with respect to observation and adding that perturbation to the observation. By adding the perturbation, FGSM seeks to find a perturbation that increases the loss function, hence fooling the agent into taking poor action. Formally, the FGSM method can be defined as follows:

$$\hat{s} = s + \epsilon \times sign \nabla_s(L) \tag{1}$$

where s and $\hat{s}$ denote the original and perturbed observation respectively, ϵ denotes a budget that scales the perturbation to keep it undetectable, and L denotes the policy's loss function. To instantiate these attacks in practice, we used the actor network's loss as the loss function to obtain the gradients to compute the perturbation for each of the RL benchmark algorithms.

Projected Gradient Descent: PGD is an iterative attack method that works similarly to FGSM in principle. While the FGSM attacks only take a single gradient step, the PGD performs multiple gradient steps to maximize the loss function and finally projects the perturbation back into the budget of ϵ. In our implementation, we set the number of iterations of PGD to be 25 after empirically observing that the degradation in performance of the RL policy displays no significant changes after 25 iterations.

Myopic Action Space attack algorithm: MAS is an attack algorithm that generates perturbation that attacks the action channel rather than the observation channel. It follows the same principle of FGSM and PGD of generating perturbations but takes gradients of the reward function with respect to the action instead of the observation. Since the gradients of the reward function with respect to the action might not be accessible, the gradients of the action probabilities or value function are taken as a proxy of the reward function to generate the perturbation, which is subsequently added to the RL agent's actions.

4 Results and discussions

In this section, we present the results of our experiments comparing the performance of the RL algorithms when subjected to the different adversarial attacks as discussed in the previous section. To fully understand the efficacy of each attack, we first trained the five RL policies using PFRL's implementation [29] on the five MuJoCo environments and ensure that the final rewards are similar to the reported scores. As such, the subsequent results are all based on mounting the attacks on the trained RL policies during test time. All experiments were performed on an internal cluster using three GeForce GTX TITAN X GPUs for training the RL agents and Intel(R) Xeon(R) CPU E5-1650 v3 CPUs for testing and mounting the attacks.

4.1 Comparison of different black-box attack strategies

We begin by visualizing and comparing the effects of different attack strategies on the five RL algorithms. To measure the effectiveness of each attack, we measure the percentage change in rewards, denoted as $\%\Delta R$ and defined as the change in rewards due to an attack as a fraction of the original rewards achieved by the trained policy. To compare the attacks, we plot the $\%\Delta R$ for each RL algorithm as a stacked bar plot to measure the overall effectiveness of each attack strategy on all the algorithms. As an illustrative example, we show the comparison for the HalfCheetah environment in Fig. 2 [1]. An important parameter when mounting these attacks is the constraint on the magnitude of the perturbations or the attack budget, ϵ. To obtain a comprehensive comparison, we mounted all the attacks at four budget levels: 25%, 50%, 100%, and 200%.

As shown in Fig 2, the first observation that can be made is that all the different attack strategies resulted in a negative $\%\Delta R$ across all RL algorithms in the HalfCheetah environments across all ϵ values. We did, however, observe certain environments seen in appendix Fig. 8, 10, 9, 11 where some attack strategies resulted in a positive $\%\Delta R$; nonetheless, the overall trend remains negative. The second observation we made is that even the most ineffective attack strategies saturate when the budget is above 100%, with the most drastic changes in $\%\Delta R$ occurring below the budget of 100%. As such, in our following experiments below, we focused only on budget levels below 100% but at a finer resolution.

Comparing the attack strategies on the observation channel versus the action channels, we observed that overall, attacks on the observation channel are as effective and sometimes more effective (in Hopper, Swimmer, and Walker) than action channel attacks up to a certain budget value, specifically for ϵ = 25% and 50%. Beyond ϵ = 50%, the action channel attacks are more effective while the observation channel generally saturates (elaborated in further detail in the following sections). In terms of the different strategies of attacks, overall, we observed that strategies that add a perturbation that has an opposite sign (flip direction) to the original action/observation value are the most effective, while strategies that add a perturbation that has the same sign (same direction) are the least effective. Furthermore, the attack strategy that perturbs the individual elements of the observation channel/action

[1] As similar trends were observed in other environments with the exception of Ant, we only present the comparisons on one environment for brevity. Please refer to the appendix Fig. 8, 10, 9, 11 for the comparisons of other environments.

channel is also slightly more effective than randomly perturbing the entire vector in a random direction. We highlight that these trends are observed consistently across all five benchmark algorithms and all environments, except for the Ant environment.

Based on our observations for the Ant environment (as shown in Appendix Fig 8), the response of the benchmark algorithm towards action channel attacks deviates slightly from the rest of the environments. Specifically, the attacks that add perturbation in an opposing direction in the action space resulted in a positive $\%\Delta R$ for most RL algorithms, while perturbations of the same direction ended up being one of the most effective strategies. We hypothesize that this is possibly due to the more complex 3-dimensional non-linear interaction of the action space of the Ant robot as compared to the rest of the environments, which are restricted to the 2-dimensional plane. This also alludes to the fact that the benchmark RL algorithm has not converged to the optimal policies in practice and the perturbations ended up being less effective.

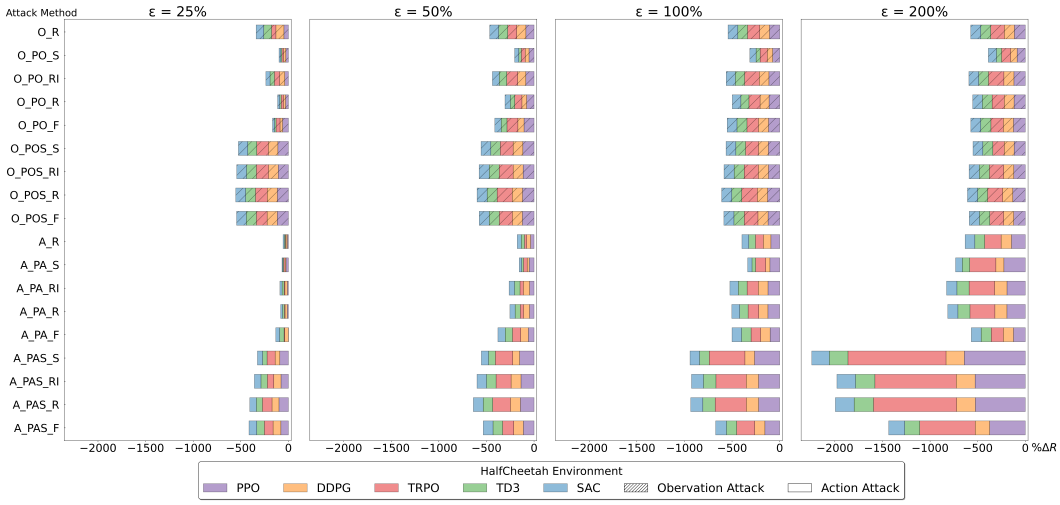


Figure 2: Comparison of black-box attack on HalfCheetah: Vertical axis represents different black-box attack strategies and the horizontal axis denotes the cumulative $\%\Delta R$ across all RL algorithms. The colors represent different RL algorithms, bars with shaded patterns represent observation channel attacks and solid bars represent action channel attacks. Each subplot denotes mounting the attacks with a specific budget ϵ on the magnitude of the perturbations.

4.2 Comparison of the robustness of different policies

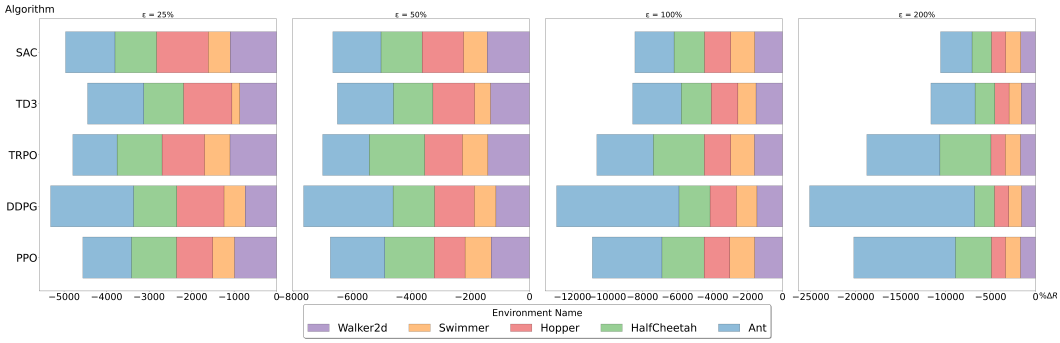


Figure 3: Comparison of the robustness of different policies: Vertical axis represents different RL algorithms and the horizontal axis denotes the cumulative $\%\Delta R$ across all black-box attacks. Different colors denote different environments and each subplot represents mounting the attacks with a specific budget ϵ on the magnitude of the perturbations, in the order of 25%, 50%, 100%, and 200%

Next, we present the comparisons between the overall robustness of different RL policies across all environments and black-box attacks. The sum of all the $\%\Delta R$ across all attacks and all environments

for each policy is illustrated in Fig 3. We hope that such a plot would reveal the natural robustness of each type of policy, i.e., how insensitive an RL algorithm is to perturbations if it wasn't specifically trained to be robust in the first place? In summary, we observed that both TD3 and SAC exhibit the most robustness across all environments and for all values of budget on the magnitude of perturbation, ϵ. Additionally, TD3 and SAC were also the least affected when increasing ϵ, while the other three algorithms $\%\Delta R$ increased significantly, with the Ant and HalfCheetah environments contributing to most of the change. On the other hand, we note that DDPG was overall the policy that is most sensitive to perturbations, especially in the Ant environment. Removing the outlier effect of the Ant environment, TRPO ranks as the policy most vulnerable to perturbations. While it is not clear why DDPG or TRPO are so sensitive, we hypothesize that the reason both SAC and TD3 are more robust is because of their shared implementation of having two Q-values to reduce overestimation. More importantly, SAC also includes an entropy bonus term in the objective function, while TD3 implements a target policy smoothing that includes noise to the action, both of which can be considered an indirect way of incorporating adversarial training in the learning process. However, this hypothesis remains to be further verified.

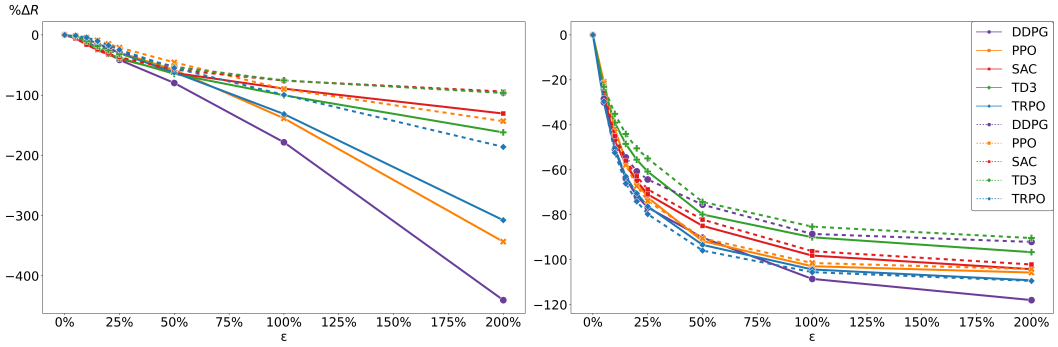


Figure 4: Effect of increasing ϵ on $\%\Delta R$ in HalfCheetah. The x-axis represents ϵ and the y-axis represents the $\%\Delta R$ with respect to ϵ. The solid line represents the average $\%\Delta R$ across all black box attacks and environments, and the dotted line represents the average excluding the Ant environment.

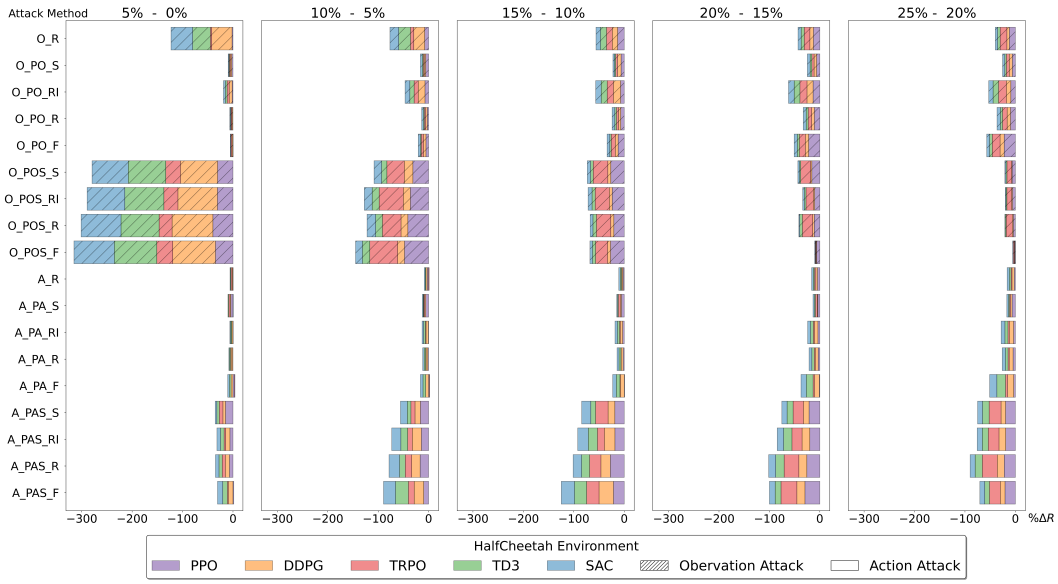


Figure 5: Detailed visualization of the effect of increasing ϵ on $\%\Delta R$ in HalfCheetah: This plot visualizes in detail the effect of increasing ϵ every 5% on $\%\Delta R$. Observe that the largest $\%\Delta R$ occurs for observation channel attacks when ϵ is low while the largest $\%\Delta R$ occurs for action channel attacks when ϵ is higher.

4.3 The effect of budget ϵ on $\%\Delta R$

While it is clear that the value of ϵ, the budget on the magnitude of perturbation, affects the effectiveness of an attack in a positively correlated manner, we study the relationship between these two variables in more detail in this section. We repeated the experiments shown in Fig. 2 by varying the values of ϵ at a finer resolution of 5%. The solid lines in Fig. 4(a) and (b) represents the average of $\%\Delta R$ across all environments for each RL algorithm for action channel attacks and observation channel attacks respectively. From this plot, we can make several more interesting observations. Firstly, we see that perturbations in the observation channel are effective but have diminishing effectiveness as seen by the saturating trends of the $\%\Delta R$ in Fig. 4(b). In contrast, perturbations in the action space do not display this characteristic as we see that $\%\Delta R$ still decreases at a linear rate as the attack budget increases up until 200%. However, one caveat is that a major contributor of the continued decrease of rewards was due to the attacks mounted on the Ant environment, as discussed in the previous section. Removing the Ant environment from the trends (as shown in the dotted lines of Fig. 4(a) and (b))revealed that the $\%\Delta R$ decreases less drastically for action channel attacks, but is still more significant than observation channel attacks. This further validates our hypothesis that RL agents that operate in environments with a higher degree of freedom are likely to be more sensitive to perturbations and display catastrophic failures.

Another observation that can be made is that in the regime of the $\epsilon < 50\%$, attacks on the agent's observation channel cause a much more significant drop in performance than attacks on the agent's action channel. This observation is further validated when we visualize the $\%\Delta R$ for every 5% increment of ϵ in Fig. 5. Once again, we only present the experiments for HalfCheetah for brevity, with the visualization for the rest of the environments shown in the appendix Fig. 12, 13, 14, 15. From the figure, we can observe that the largest drop in rewards for observation channel attacks (bars with diagonal patterns) occurs when ϵ is between 0 to 10%. Meanwhile, we observe the exact opposite trend in action channel attacks where the initial effect when ϵ is between 0 to 5% was small but the $\%\Delta R$ increases as we increase ϵ.

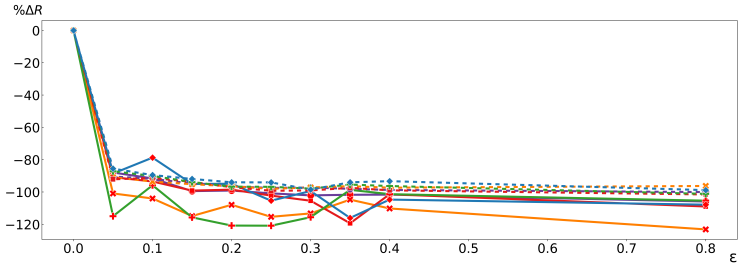
4.4 Comparison of different white-box attacks

Next, we compare the effects of the three white-box attack strategies we had selected on the performance of the benchmark RL policies. While this is by no means a comprehensive experiment of white-box attack strategies, we hope that the results in these sections will provide some initial insights. Fig. 6 illustrates the average $\%\Delta R$ across all environments for each attack strategy. Similar to the black-box attacks we implemented, we observed that all white-box attacks resulted in a general negative trend. Compared to black-box attacks, we observed that the decrease in $\%\Delta R$ is much steeper than the trends observed in Fig. 4. However, it is worth highlighting that the context and range of the ϵ values used in white-box experiments are different. While the values of ϵ in the black-box experiments were expressed as a percentage of the action/observation space or the actual values of the action/observations themselves, the values of ϵ used in the white-box experiments were based on the values reported in the literature. Furthermore, the black-box attack strategies we proposed followed the strategy of adding noise, while the white-box strategy we implemented all incorporated some form of optimization. As such, no direct comparison between white and black-box trends can be made. Nevertheless, an interesting observation we made is that while increasing the value of ϵ resulted in a monotonic decrease in $\%\Delta R$ for black-box attacks, the $\%\Delta R$ for white-box attacks exhibited some form of fluctuations, although we still observe a general decreasing trend.

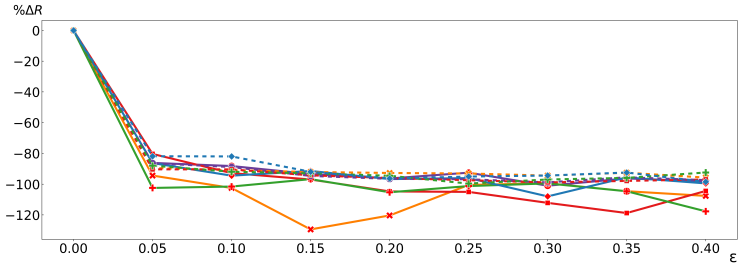
Comparing the attacks on the observation channel (FGSM and PGD) versus attacks on the action channel (MAS), we observe that, in general, both types of attacks perform similarly asymptotically as we increase the value of ϵ. However, at lower values of ϵ, action channel attacks have a higher variance in terms of the $\%\Delta R$ across different algorithms. Specifically, we observe that the smaller values of ϵ increased the $\%\Delta R$ by almost 100% for TD3. However, removing the results of the Ant environment from the trend (dotted lines) showed that the trend for TD3 reverts to a trend that follows the rest of the environment.

When we compare the performance across different RL policies, we observe that most of the algorithms had similar robustness, with DDPG and TD3 being the most sensitive to perturbations and displaying the largest $\%\Delta R$ when subjected to observation channel attacks. Once again, these trends became less extreme once removing the effect of the Ant environment. In terms of action channel attack (MAS), one interesting observation is that most RL algorithms performed similarly except for

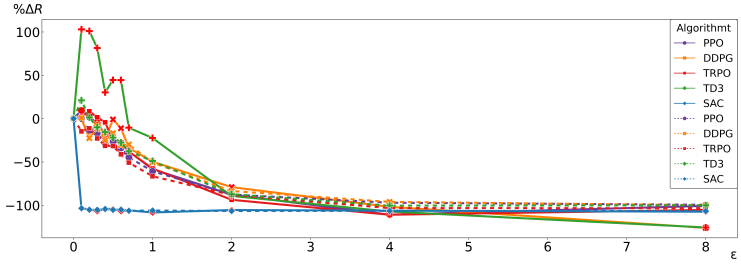
SAC, which displayed a large drop in performance even with a small value of ϵ. This is a surprising observation as SAC was one of the most robust policies in the black-box attack experiments, and even removing the effect of the Ant environment did not change the trends significantly. As such, it would be interesting for future studies to investigate why SAC is robust towards observation channel perturbation but becomes particularly sensitive to action space perturbation, specifically the white-box MAS attack.



(a) FGSM Attack



(b) PGD Attack

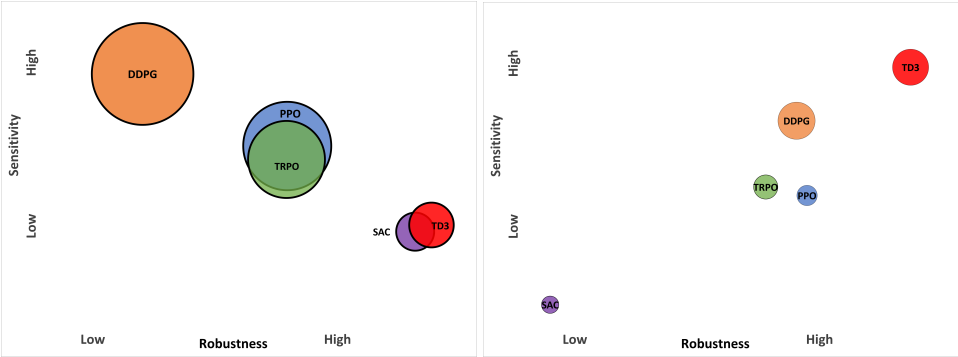


(c) MAS Attack

Figure 6: White-box attacks trends for HalfCheetah: The plots show the relationship between the value of ϵ (x-axis) and $\%\Delta R$ (y-axis). Line markers in the plots represent experiments we ran with a specific value of ϵ. The solid line represents the average $\%\Delta R$ across all environments, and the dotted line represents the average excluding the Ant environment.

4.5 Summary and discussions

To summarize our findings, we compile our observations into Fig. 7 and rank the benchmark RL algorithms according to three criteria: robustness, range of robustness, and sensitivity. Furthermore, we classified the algorithm's characteristics according to black-box attacks Fig. 7a and white-box attacks Fig. 7b. The horizontal axis represents an algorithm's robustness, where we define robustness as the average $\%\Delta R$ across all attacks and all environments. The vertical axis represents an algorithm's sensitivity. We define sensitivity by taking the average difference for all $\%\Delta R$ across all possible pairs of strategies and computing the absolute value of it. Intuitively, the sensitivity gives us a sense of how much we can expect the performance of an RL policy will change when subjected to different attacks. Finally, the size of the circles in Fig. 7 represents the range of the robustness of an algorithm by taking the difference between the maximum and minimum $\%\Delta R$ under the white-box and black-box attack scenario, respectively. Generally speaking, we observe that



(a) BlackBox Attack (b) WhiteBox Attack

Figure 7: Summary of observations: Visualization of the relative sensitivity and robustness of common RL benchmark algorithms. The x-axis denotes the robustness of the algorithms and the y-axis denotes the sensitivity of the algorithms. Colors represent the five examined algorithms, the subplot on the left indicates the performance of the algorithm under black-box attacks while the subplot on the right indicates the performance under white-box attacks. The circle sizes indicate the range of robustness by computing the differences between the maximum and the minimum $\%\Delta R$.

TD3 exhibits the best robustness across both white-box and black-box attacks, while SAC performs well under black-box attacks but performs extremely poorly on white-box attacks. We also note that PPO and TRPO are robust to a certain extent with medium sensitivities, but DDPG ranks the lowest in terms of having low robustness and high sensitivity. Finally, we also highlight that black-box attacks have a larger range of effects on the RL policies in general (larger circles) when compared to white-box attacks, which have more consistent effects (smaller circles).

5 Conclusion

In this work, we compared commonly used benchmark RL algorithms' robustness towards various types of perturbation during test time. We designed a suite of simple black-box attack strategies to perturb the RL agent's observation and action channels, and we also implemented three commonly used white-box optimization-based attacks that perturbed the agent's observation and action channels. From our experiments, we made the following conclusions: Firstly, from the black-box attack strategies we tested, a recurring theme is that observation channel attacks are more effective than action channel attacks, but only until a certain threshold on the magnitude of the perturbation. Beyond this threshold, the effects of observation attacks saturate while action channel attacks may continue to have some effect. We also find that the Ant environment generally amplifies the effect of attacks. In terms of the robustness of different policies under black-box attacks, SAC and TD3 were generally robust, while DDPG and TRPO were the most sensitive to perturbations. When subjected to optimization-based white-box attacks in the observation channel, most policies performed similarly, with DDPG and TD3 being the most sensitive, while SAC was found to be extremely sensitive to action channel attacks. We find it intriguing that two of the most robust policies under black-box attacks ended up being the most sensitive to attacks under white-box attacks, and future work will seek to further understand this phenomenon. Furthermore, we will extend this study to include a more comprehensive comparison of existing optimization-based black-box attacks and white-box attacks.

References

[1] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Ng. An application of reinforcement learning to aerobatic helicopter flight. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006.

[2] Yuandou Wang, Hang Liu, Wanbo Zheng, Yunni Xia, Yawen Li, Peng Chen, Kunyin Guo, and Hong Xie. Multi-objective workflow scheduling with deep-q-network-based multi-agent reinforcement learning. *IEEE Access*, 7:39974–39982, 2019.

[3] Mohd Aiman Kamarul Bahrin, Mohd Fauzi Othman, Nor Hayati Nor Azli, and Muhamad Farihin Talib. Industry 4.0: A review on industrial automation and robotic. *Jurnal Teknologi*, 78(6-13), 2016.

[4] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

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

[6] Vahid Behzadan and Arslan 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.

[7] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748*, 2017.

[8] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. *arXiv preprint arXiv:1712.03632*, 2017.

[9] Chaowei Xiao, Xinlei Pan, Warren He, Jian Peng, Mingjie Sun, Jinfeng Yi, Mingyan Liu, Bo Li, and Dawn Song. Characterizing attacks on deep reinforcement learning. *arXiv preprint arXiv:1907.09470*, 2019.

[10] Xian Yeow Lee, Sambit Ghadai, Kai Liang Tan, Chinmay Hegde, and Soumik Sarkar. Spatiotemporally constrained action space attacks on deep reinforcement learning agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4577–4584, 2020.

[11] Kai Liang Tan, Yasaman Esfandiari, Xian Yeow Lee, Soumik Sarkar, et al. Robustifying reinforcement learning agents via action space adversarial training. In *2020 American control conference (ACC)*, pages 3959–3964. IEEE, 2020.

[12] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. *Advances in Neural Information Processing Systems*, 33:21024–21037, 2020.

[13] Huan Zhang, Hongge Chen, Duane Boning, and Cho-Jui Hsieh. Robust reinforcement learning on state observations with learned optimal adversary. *arXiv preprint arXiv:2101.08452*, 2021.

[14] Janosch Moos, Kay Hansel, Hany Abdulsamad, Svenja Stark, Debora Clever, and Jan Peters. Robust reinforcement learning: A review of foundations and recent advances. *Machine Learning and Knowledge Extraction*, 4(1):276–315, 2022.

[15] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[16] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

[17] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*, 2017.

[18] Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. Simple black-box adversarial attacks. In *International Conference on Machine Learning*, pages 2484–2493. PMLR, 2019.

[19] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram 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.

[20] Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. In *International Conference on Learning Representations*, 2020.

[21] Tong Chen, Jiqiang Liu, Yingxiao Xiang, Wenjia Niu, Endong Tong, and Zhen Han. Adversarial attack and defense in reinforcement learning-from ai security view. *Cybersecurity*, 2(1):1–22, 2019.

[22] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[23] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[24] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[25] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[26] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[27] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[28] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[29] Yasuhiro Fujita, Prabhat Nagarajan, Toshiki Kataoka, and Takahiro Ishikawa. Chainerrl: A deep reinforcement learning library. *Journal of Machine Learning Research*, 22(77):1–14, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of computing and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]