Assessing the Impact of Distribution Shift on Reinforcement Learning Performance

Ted Fujimoto[1*] **Joshua Suetterlein**[1] **Samrat Chatterjee**[1,2] **Auroop Ganguly**[2,1]
[1]Pacific Northwest National Laboratory [2]Northeastern University

Abstract

Research in machine learning is making progress in fixing its own reproducibility crisis. Reinforcement learning (RL), in particular, faces its own set of unique challenges. Comparison of point estimates, and plots that show successful convergence to the optimal policy during training, may obfuscate overfitting or dependence on the experimental setup. Although researchers in RL have proposed reliability metrics that account for uncertainty to better understand each algorithm's strengths and weaknesses, the recommendations of past work do not assume the presence of out-of-distribution observations. We propose a set of evaluation methods that measure the robustness of RL algorithms under distribution shifts. The tools presented here argue for the need to account for performance over time while the agent is acting in its environment. In particular, we recommend time series analysis as a method of observational RL evaluation. We also show that the unique properties of RL and simulated dynamic environments allow us to make stronger assumptions to justify the measurement of causal impact in our evaluations. We then apply these tools to single-agent and multi-agent environments to show the impact of introducing distribution shifts during test time. We present this methodology as a first step toward rigorous RL evaluation in the presence of distribution shifts.

1 Introduction

The field of RL has enjoyed some spectacular recent advancements, like reaching superhuman levels at board games [Silver et al., 2016, 2018, Bakhtin et al., 2022], sailboat racing [McKinsey and Company, 2021], multiplayer poker [Brown and Sandholm, 2019], and real-time strategy games [Berner et al., 2019, Vinyals et al., 2019]. Transitioning RL toward a rigorous science, however, has been a more complicated journey. Like other fields in machine learning, progress in RL might be compromised by a lack of focus on reproducibility combined with more emphasis on best-case performance. To remedy this problem, reliability metrics have been proposed to improve reproducibility by making RL evaluation more rigorous. Some examples include the dispersion and risk of the performance distribution [Chan et al., 2020], and interquartile mean with score distributions [Agarwal et al., 2021]. Past work, however, does not assume the presence of distribution shift during test time.

In general, distribution shift in machine learning occurs when there is a difference between the training and test distributions, which can significantly impact performance when the machine learning system is deployed in the real world [Koh et al., 2021]. In supervised learning, distribution shift could cause a decrease in accuracy. We will focus on distribution shift in RL, which could cause a decline in expected returns. This impact on performance is a symptom of overfitting in deep RL, which is a problem that requires carefully designed evaluation protocols for detection [Zhang et al., 2018]. While there are many types of distribution shift, we will focus on test-time adversarial examples and the introduction of new agents in multi-agent ad hoc teamwork.

*Corresponding Author: ted.fujimoto@pnnl.gov

Workshop on Regulatable Machine Learning at the 37th Conference on Neural Information Processing Systems (RegML @ NeurIPS 2023).

Taking inspiration from car crash tests and safety ratings from trusted organizations, like the ratings standards of the National Highway Traffic Safety Administration (NHTSA) or the Insurance Institute for Highway Safety (IIHS), we believe RL would benefit from techniques that evaluate robust performance after training. Here, we contribute some recommendations for evaluation protocols using time series analysis to measure the performance of RL agents that encounter distribution shift at test time. Specifically, we recommend (1) the comparison of time series forecasting models of agent performance, (2) using prediction intervals to capture the distribution and uncertainty of future performance, and (3) counterfactual analysis when distribution shift has been applied by the experimenter. We are not recommending a strict set of rules that must be rigidly followed, but we believe this methodology is a promising start towards reliable comparison of pretrained RL agents exposed to distribution shift in both single and multi-agent environments.

In section 2, we mention past related work on distribution shift and RL reproducibility. In section 3, we argue why time series analysis is needed for evaluation of RL algorithms under distribution shift. In section 4, we outline our recommendations for RL evaluation with time series analysis. In section 5, we provide examples of such analyses in single and multi-agent RL. In section 6, we conclude with suggestions for future research in RL evaluation and describe how it can advance ML safety and regulation.

2 Related Work

Distribution (or dataset) shift occurs when the data distribution at training time differs from the data distribution at test time [Quinonero-Candela et al., 2008]. The focus of this paper is not to detect distribution shift [Rabanser et al., 2019], but to assume it exists while measuring agent performance. There has been work on reproducibility under distribution shift for supervised learning [Koh et al., 2021], but RL will be the focus here. In particular, we focus on how overfitting to the training environment can affect the agent's performance during evaluation in a test environment [Zhang et al., 2018]. Although we are taking a time series perspective in this paper, we are not proposing a new method of machine learning to train time series models [Ahmed et al., 2010, Masini et al., 2023]. We will use time series only as a method of evaluation of RL performance.

The distribution shifts of focus in this paper will be adversarial attacks on images (Atari game observations) and agent switching in multi-agent environments. There has been extensive research on adversarial attacks on supervised learning models, like support vector machines and neural networks [Huang et al., 2011, Biggio et al., 2012, Goodfellow et al., 2014, Kurakin et al., 2016]. While we will focus on the adversarial attacks proposed in Huang et al. [2017], there has been related research on adversarial attacks in single-agent [Kos and Song, 2017, Pattanaik et al., 2017, Rakhsha et al., 2020, Zhang et al., 2020], and multi-agent RL [Gleave et al., 2019, Ma et al., 2019, Figura et al., 2021, Fujimoto et al., 2021, Casper et al., 2022, Cui et al., 2022]. Another set of experiments will test the ad hoc teamwork of the group of agents [Stone et al., 2010, Barrett and Stone, 2015, Rahman et al., 2021, Mirsky et al., 2022], where agent switching will be treated as a distribution shift among the group.

Rigorous evaluation of RL algorithms is still a topic that requires further investigation. Henderson et al. [2018] show that even subtle differences, like random seeds and code implementation, can affect the training performance of deep RL agents. Engstrom et al. [2020] give a more thorough study of RL implementation and provide evidence that seemingly irrelevant code-level optimizations might be the main reason why Proximal Policy Optimization [Schulman et al., 2017] tends to perform better than Trust Region Policy Optimization [Schulman et al., 2015]. Colas et al. [2018] show how the number of random seeds relates to the probability of statistical errors when measuring performance in the context of deep RL. The inherent brittleness in current deep RL algorithms calls into question the reproducibility of some published results. This has led to proposals for rigorous and reliable RL evaluation techniques grounded in statistical practice. Chan et al. [2020] recommended reliability metrics like interquartile range (IQR) and conditional value at risk (CVaR). Jordan et al. [2020] suggested metrics like performance percentiles, proposed a game-theoretic approach to quantifying performance uncertainty, and developed a technique to quantify the uncertainty throughout the entire evaluation procedure. Agarwal et al. [2021] recommended stratified bootstrap confidence intervals, score distributions, and interquartile means. For multi-agent cooperative RL, Gorsane et al. [2022] proposed a standard performance evaluation protocol using RL recommendations from past papers.

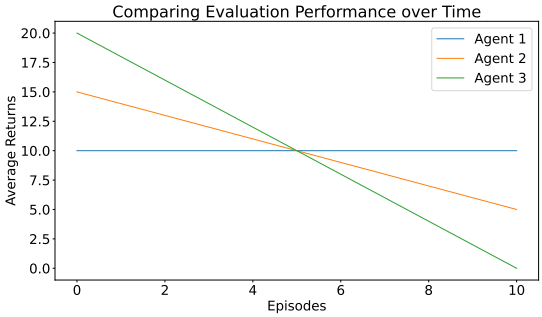


Figure 1: In this simplified plot of agent performance in the presence of worsening distribution shifts over time. All three agents have average returns of 10. It is clear, however, that agent 3 is the least desired agent over time. Even though agent 3 starts out with the highest average returns, it seems to have overfit to the training environment and fails to maintain its superior performance. Point estimates alone would not capture this behavior.

3 The Need to Measure Performance Over Time

3.1 A Time Series Perspective

We argue for time series analysis as a solution to the problems that point estimates and confidence intervals alone cannot fix. Because we are not certain of the agent's future performance, forecasting the agent's performance is needed if we assume RL agents will inevitably encounter distribution shifts in its environment after training. Point estimates alone can fail to explain decreases in performance at test time. In Figure 1, we assume three agents that act in an environment that experiences increasing distribution shift as the number of episodes continues. Each episode can be interpreted as a day's performance of an RL agent. This is motivated by a hypothetical scenario where RL developers are deploying their agents in an environment that experiences changes they did not anticipate. All agents achieve the same average returns, but there is a clear difference of performance over time. In this hypothetical example, Agent 3 seems to have overfit to the training environment because it starts high but ends as the lowest performer. In the longer term, Agent 2 is preferable because its decrease in performance is slower, and eventually outperforms Agent 3. Agent 1 represents the ideal RL agent because its performance over time never decreases.

As shown in previous work [Chan et al., 2020, Agarwal et al., 2021], confidence intervals are important to quantify the uncertainty of point estimates of aggregate performance. In evaluating performance in the presence of distribution shift, however, confidence intervals for mean scores are insufficient. If we want to account for the uncertainty of the performance trends, it would be preferable to understand the distribution of values and where we expect our forecasting model to generate the next data point sampled [Buteikis, 2020]. Hence, when evaluating trends in performance under distribution shift, prediction intervals can be more helpful than confidence intervals. Time series analysis can also be used to measure the causal impact of distribution shift on one (or many) agents. One way to accomplish this is counterfactual analysis. That is, we need a model of the agent acting in the environment as if the change had never taken place. In our methodology, we will assume the trained agents have achieved a clear, steady trend in performance (as opposed to noisy or random) in the absence of distribution shift.

We now provide the following definition of RL agent time-series performance measurements, similar to how performance is measured in RL training research, that will be relevant in the next subsection:

Definition 1. Let $X^A(t)$ the time series performance for $t = 1 \dots N$, where A is an RL agent in a simulated deterministic environment $\mathcal{M}$. For each $X^A(t)$, there is an associated sample of performance measurements $x^A(t) = (x^{A,1}(t), \dots, x^{A,M}(t))$. Here, the i in $x^{A,i}(t)$ is a random seed $i \in \mathfrak{S}$, where $\mathfrak{S}$ is a finite set of M random seeds. Hence, we define:

$$X^A(t) = \mathbb{E}[x^{A,i}(t)] = \frac{\sum_{i \in \mathfrak{S}} x^{A,i}(t)}{M} \tag{1}$$

which is the expected performance[2] at time t over all the random seeds in $\mathfrak{S}$.

Within-episode performance is difficult to measure because it requires knowledge of the particular environment's states to know when to make an intervention. To make our methodology more general, we measure the impact of distribution shift over many episodes, where $X^A(t)$ is the expected performance at an episode t. This allows us to measure the positive (or negative) impact of distribution shifts over time. An example of this could be measuring the energy usage or customer satisfaction of a RL-trained HVAC system each day over a week if exposed to distribution shift.

3.2 RL and the Fundamental Problem of Causal Inference

Practitioners of causal inference must remember its fundamental problem:

Fundamental Problem of Causal Inference 3.1 (Holland [1986])**.** *Let $Y_a(u)$ on unit u be the random variable where $a = 1$ denotes exposure to the treatment, and $a = 0$ denotes exposure to the control. Then it is impossible to observe both $Y_{a=1}(u)$ and $Y_{a=0}(u)$ on u and, therefore, it is impossible to observe the treatment effect on u: $Y_{a=1}(u) - Y_{a=0}(u)$.*

Another way of explaining this problem is that if we were to expose a unit u to some intervention, we will never see the world where we never exposed the unit u to that intervention. That is, observing individual treatment effect is impossible. This has not discouraged researchers from developing methods that circumvent this problem [Spirtes et al., 2000, Pearl, 2009, Imbens and Rubin, 2015, Peters et al., 2017]. These methods usually require assumptions that manage the messiness of the real world, which makes it possible to construct valid arguments in favor of causal inference. RL agents in simulated environments, however, allow us to make assumptions that might be too strong for less predictable units (like humans or animals). In a deterministic environment, with a given random seed, regardless of how many times you reach a state s, the agent will choose the same action and receive the same reward at state s. That is, RL agents in deterministic environments with fixed random seeds allow us to circumvent the fundamental problem of causal inference because we can just reset the environment and see what happens when we choose to intervene or not. This does not contradict Henderson et al. [2018], where they argue that different random seeds can lead to different performances and behaviors. Here, we claim that different RL agents with their own random seed may exhibit different behaviors, but will repeat those behaviors if the random seeds are fixed. Now that we are concerned with RL evaluation over time, we can make the following assumption:

RL Fixed Seed Assumption 3.2. *Let T, such that $1 \leq T \leq N$, be the time an intervention occurs. Let G designate groups such that $G = 1$ indicates the treatment group and $G = 0$ indicates the control group. Consider a time series outcome $X^A(t)$ as in Equation 1, where A is a RL agent in a simulated deterministic environment $\mathcal{M}$. Let $X^A(t < T)$ be the performance before time T. Then,*

$$\mathbb{E}[X^A(t < T)|G = 1] = \mathbb{E}[X^A(t < T)|G = 0] \tag{2}$$

Let $X^A_{U=u}(t)$ be the performance measurement as above with the counterfactual intervention $U = u$. We define $U = 1$ and $U = 0$ as being exposed to an intervention (e.g., distribution shift) and not exposed, respectively. Then, on average, the performance of the control group is the counterfactual performance of the treatment group as if it had not been exposed to the intervention:

$$\mathbb{E}[X^A(t)|G = 0] = \mathbb{E}[X^A_{U=0}(t)|G = 1] \tag{3}$$

This assumption basically says that if we have a fixed set of random seeds[3] in a deterministic environment, then, on average, the expected returns of the control group is the same as the expected returns of the treatment group if no out-of-distribution intervention ever occurred. This makes intuitive sense because both groups have the same random seeds, which implies they will exhibit identical behavior in deterministic environments if no outside influence is introduced. This will be helpful when justifying causal inference in the next section.

[2]We use expected performance just as a simple default. For example, in accordance with Agarwal et al. [2021], one could replace expected performance with the interquartile mean.

[3]Reproducibility can be difficult to achieve when running on a GPU. We discuss this further and how to control for it in the appendix.

4 Recommendations for Time Series Evaluation

4.1 Methods

4.1.1 Comparison of Simple Time Series Forecasting Trends

To compare RL algorithms at test time, we compare the performance forecasts of each RL agent. This recommendation is inspired by the use of time complexity or running time to measure the efficiency of an algorithm [Cormen et al., 2022]. As illustrated in Figure 1, looking at performance trends of RL agents is an informative way to observe overfitting during test time. Here, we use simple time series forecasts over more complex time series models that might better predict the performance trend. The reason is that one cannot make too many strong, general assumptions (e.g., seasonality, how many past values to use for autoregression) on the time series trend of agent performance in all environments. For test-time distribution shift, we only assume the trend will be pessimistic and not increase. This is reasonable if we assume the ideal case is when performance never decreases (like Agent 1 in Figure 1). We default to using Holt's linear damped trend method [Gardner Jr and McKenzie, 1985, Holt, 2004, Hyndman and Athanasopoulos, 2018] to model the (likely pessimistic) trend of agent performance when in the presence of distribution shift. Researchers evaluating their own experiments, however, may find that more complex models would be more appropriate for their own studies if they see performance trends that warrant such assumptions.

4.1.2 Prediction Intervals over Future Performance

Along with time series forecasts, we recommend prediction intervals over future average returns [Hyndman and Athanasopoulos, 2018]. The combination of simple time series forecasts and prediction intervals map out the range of average returns over time. As stated previously, point estimates with confidence intervals are not enough to capture the range of returns over time. Prediction intervals act as a compliment to the time series forecasting by visualizing the uncertainty of the possible future average returns. When applying both of these methods, we can specify the most robust RL algorithm as the trend with the most optimistic forecast on performance and the smallest prediction interval. Here, we assume there exists distribution shifts but do not necessarily know when or where. Since we will be using Holt's linear damped trend method, 95% prediction intervals might be too narrow [Hyndman, 2014]. Hence, we will default to 99% prediction intervals.

4.1.3 Difference-in-differences Analysis for RL Performance

Difference-in-differences (DiD) measures the causal effect between a treatment and control group, where the treatment group is exposed to an intervention at a certain point in time [Cunningham, 2021, Huntington-Klein, 2021]. Intuitively, it represents how much more the treatment group was affected by the intervention at some time t compared to the change of the unaffected control group at the same time t. Since we are dealing with time series, we will adhere to the DiD formulation in Moraffah et al. [2021]:

Definition 2. If $t < T$ and $t > T$ denote the pre- and post- treatment periods, respectively, then we can calculate the DiD measure using the average treatment effect metric over a time series $X(t)$ as follows:

$$DiD = \{\mathbb{E}[X(t > T)|G = 1] - \mathbb{E}[X(t < T)|G = 1]\} - \{\mathbb{E}[X(t > T)|G = 0] - \mathbb{E}[X(t < T)|G = 0]\} \tag{4}$$

where G indicates the treatment group $(G = 1)$ and the control group $(G = 0)$.

Any methodology that relies on DiD must satisfy the **parallel trends assumption**, which says that *if no treatment had occurred, the difference between the treated group and the untreated group would have stayed the same in the post-treatment period as it was in the pre-treatment period* [Huntington-Klein, 2021]. RL agents in deterministic environments with fixed seeds trivially satisfy this because the agent will take the same action at each state and receive the same reward regardless of how many times the evaluation is repeated.

From Equation 2 of the RL fixed seed assumption, DiD simplifies to the equation below when evaluating RL performance:

$$DiD = \mathbb{E}[X^A(t > T)|G = 1] - \mathbb{E}[X^A(t > T)|G = 0] \tag{5}$$

where A is the RL agent being evaluated. This follows from the pre-treatment averages canceling each other out. Hence, we only need to measure the post-treatment effect of RL performance. Using Equation 3 of the RL fixed seed assumption, DiD becomes the following:

$$DiD = \mathbb{E}[X^A(t > T)|G = 1] - \mathbb{E}[X^A_{U=0}(t > T)|G = 1] \tag{6}$$

Equations 5 and 6 say the following: Measuring the DiD effect is equivalent to measuring the average time series post-treatment effect of agent A between the treatment and control group. If the agents in both the treatment and control groups have the same fixed random seeds, we can interpret the performance measurements of the control group as the counterfactual of the treatment group as if the treatment group was never exposed to the distribution shift intervention. Hence, we have shown that the RL fixed seed assumption justifies causal inference in our time series analysis.

4.2 Recommended Procedure

4.2.1 If you can control when the distribution shift occurs, measure the causal impact

Similar to car crash tests in controlled settings, we propose an evaluation method where the experimenter can control when a RL agent experiences a shift in distribution. First, take a trained RL agent and have it interact in its environment until some time (or episode) T, which will be the pre-treatment period. At time T, the post-treatment period starts with the experimenter applies a shift in distribution. Such distribution shifts include adversarial examples [Goodfellow et al., 2014]. In the multi-agent cooperative tasks, sudden replacement of agents can also cause shifts in distribution [Mirsky et al., 2022]. The experimenter logs the performance scores at each point in time. When the agent reaches the end of the experiment (or some time threshold), evaluate the causal impact from the counterfactual model (the RL agent control group) and the treatment group's post-treatment performance using DiD. The results will show how much the distribution shift impacted the performance of the agent assuming a counterfactual model that represents the agent's performance if the distribution shift never happened.

To show the impact of the distribution shift, we use the template of time-series impact plots from Brodersen et al. [2015]. This template consists of three panels: an original plot, a pointwise plot, and a cumulative plot. In the original plot, the raw performance is shown. The pointwise plot shows the difference between the observed data and the counterfactual predictions, which is the inferred causal impact of intervention. The cumulative plot shows the cumulative impact of the intervention over time. Unlike in Brodersen et al. [2015], instead of Bayesian structural time-series models, we use DiD because we are measuring only one variable (returns) and our assumptions justify its application.

4.2.2 Otherwise, compare agents using simple time series trends with prediction intervals

Here, we assume the experimenter has no control over when the distribution shift will occur. It is also possible that multiple instances of distribution shift can occur at different times. Such scenarios are intended to be a closer representation of real-world RL agent deployment. This implies, without further assumptions, an observational study is the best we can do. We propose comparison of RL performance using time series trends, like Holt's Linear Damped Trend Method, with prediction intervals. An example is provided in Figure 2. Here, we can compare the performance of agents trained on different RL algorithms. The idea is similar to the comparison of agents in Figure 1. The main difference is that we are not just focusing on the agents' measured performance, but also on the forecast of future performance with prediction intervals. Here, robust performance can be interpreted as the the forecast of future performance and the prediction intervals visualize its uncertainty over time. Ideally, the agent that performs the best would have the highest trend line and the smallest prediction interval. Using time series forecasts with prediction intervals are meant to be an improvement over point estimates with confidence intervals to better show the uncertainty of measured RL performance over time. Like in section 3, we interpret each time point as an episode, and $X^A(t)$ represents the RL agent's performance during that episode.

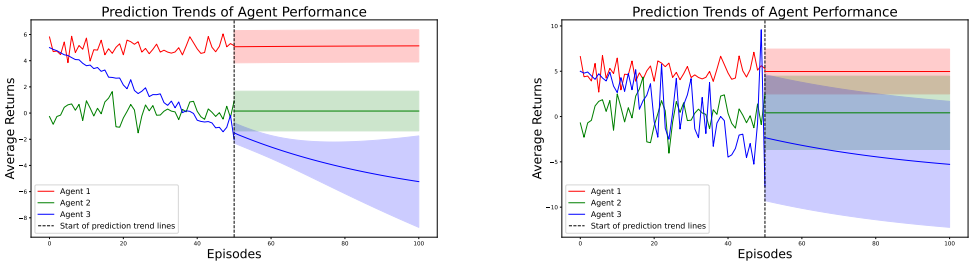


Figure 2: The idea behind these graphs is the same as in Figure 1. The main differences are that we use time series forecasts and prediction intervals to show the predicted performance trend of each agent. Here, the performances between episodes 0-50 are the measured performances of the agents in the environment. The plots at episodes 50-100 are not measured performance, but prediction trends of future performance with prediction intervals. **Left:** The differences in performance are clear because the prediction intervals do not overlap at the end of the plot. Hence, Agent 1 has the best performance because the forecast is not decreasing and the prediction interval is small. The fact that Agent 2's interval does not overlap with Agent 3's prediction interval shows that Agent 1 and Agent 2 have significantly better performance forecasts. **Right:** Here, all agents have noisier performance. Even though Agent 3 still has a downward trend, its much noisier performance briefly spikes up to match Agent 1's performance. Hence, we want prediction intervals that anticipate this uncertainty by showing interval overlap between agent performance over time. There is no longer a significant difference between Agents 2 and 3 because their prediction intervals overlap at every time step.

5 Examples of RL Time-series Analysis

In our time series evaluation, we model the scenario as deploying the agent(s) out in the intended environment. In this scenario, we have the pretrained agents run over a number of episodes. As mentioned previously, the causal impact procedure can be interpreted as the RL version of car crash tests, where RL agent(s) are deployed in a controlled environment and undergo repeated performance testing. When evaluating the causal impact of the distribution shift, we introduce the shift at some halfway point. We measure the performance of the treatment and control groups, then measure the difference and cumulative impact of the shift. This can be interpreted as the human maintainers measuring robust agent performance and quantifying the losses that can accrue when the agent is deliberately exposed to a distribution shift. In the observational case, we can interpret having the agents run over a number of episodes as deploying the agent into the environment each day while recording its performance. The distribution shifts in the observational evaluations happen randomly, so the human maintainers have no control over how the shifts are introduced.

In the single-agent case, we measure the performance of RL agents trained on Atari games [Bellemare et al., 2013] in the presence of adversarial attacks. Here, in Figure 3, we use pretrained A2C and PPO agents implemented in Stable-Baselines3 [Raffin et al., 2021]. For adversarial attacks on the Atari game images, we use the Fast Gradient Sign Method (FGSM) [Goodfellow et al., 2014]. In the multi-agent case (Figure 4), we measure the performance of a cooperative group of decentralized PPO agents in the presence of ad hoc agent switching. The multi-agent environment we use is a framework for power-systems-focused simulations called PowerGridworld [Biagioni et al., 2022]. In both cases, we use the causal impact plot template from Brodersen et al. [2015].

For the observational evaluations, the single-agent and multi-agent cases use different approaches to randomly introducing distribution shift. In the single-agent case, we define a probability threshold for adversarial attacks. If the random number generator (like `random.random()` in Python) gives a number above the threshold, the Atari game image is attacked. In the multi-agent case, we use a strategy similar to Rahman et al. [2021], where the number of steps an agent is switched out is drawn from a uniform distribution. Figure 5 provides some Atari game examples of observational studies. We use 10 random seeds for both causal and observational time-series evaluation. More information on the plots is provided in the appendix.

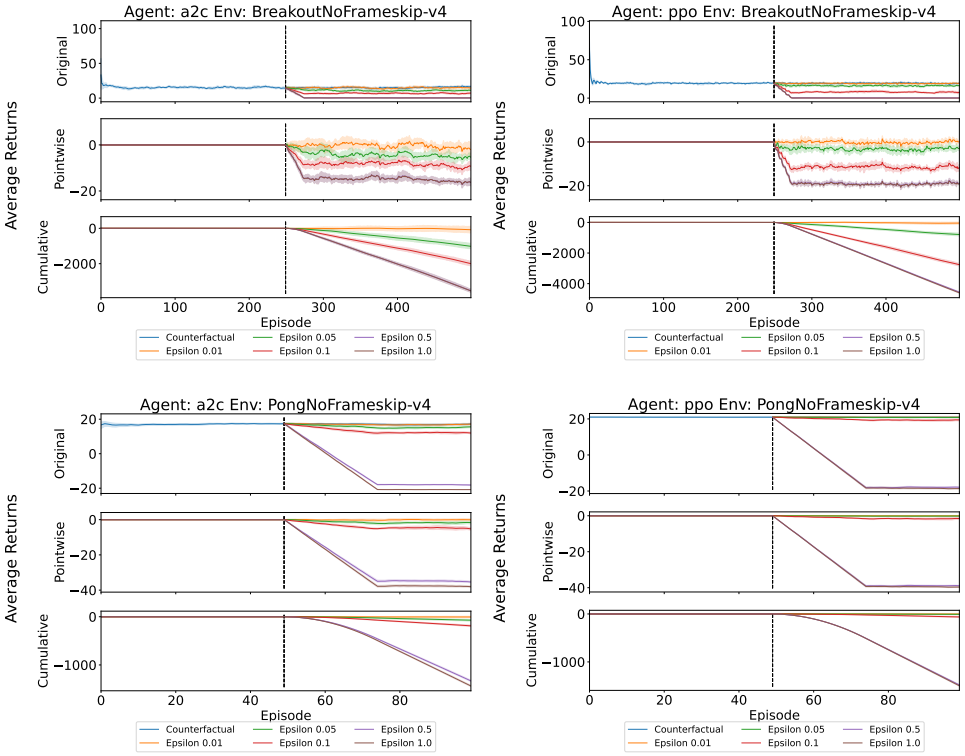


Figure 3: The causal impact plots shown here illustrate the impact of FGSM adversarial attacks on RL agents trained on the Breakout and Pong Atari games. Each row represents an Atari game. Each column represents a RL algorithm (A2C or PPO). The original plots here show the rolling mean of the rewards over time. The pointwise plots show the difference between the counterfactual performance and the performance when the agent is attacked. The cumulative performance is the summation of the rewards gained or lost over time. As expected, the performance tends to drop as ϵ increases.

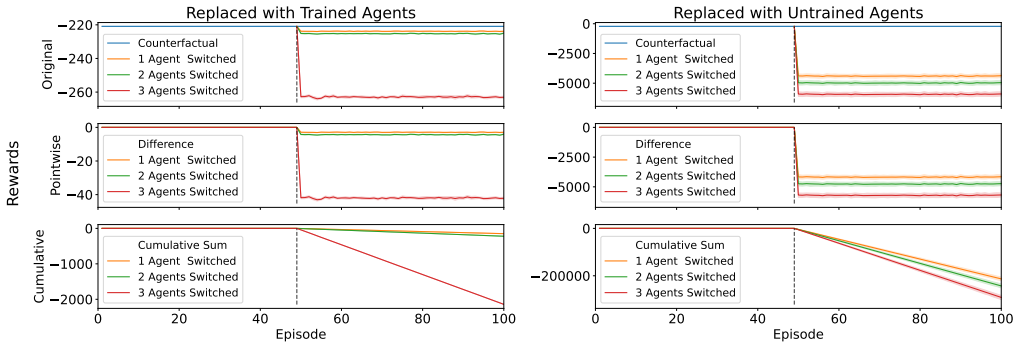


Figure 4: The plots here show the impact of the ad hoc switching of agents in a group of 5 in the PowerGridworld environment. **Left Column:** We replace 1, 2, or 3 agents out of the group of 5 with agents that trained with a different group. While there is little change when only replacing 1 or 2 agents, we see that performance dramatically decreases when 3 agents have been switched out. **Right Column:** We see that just switching out 1 agent in the group with 1 untrained agent causes a significantly large decrease in group performance.

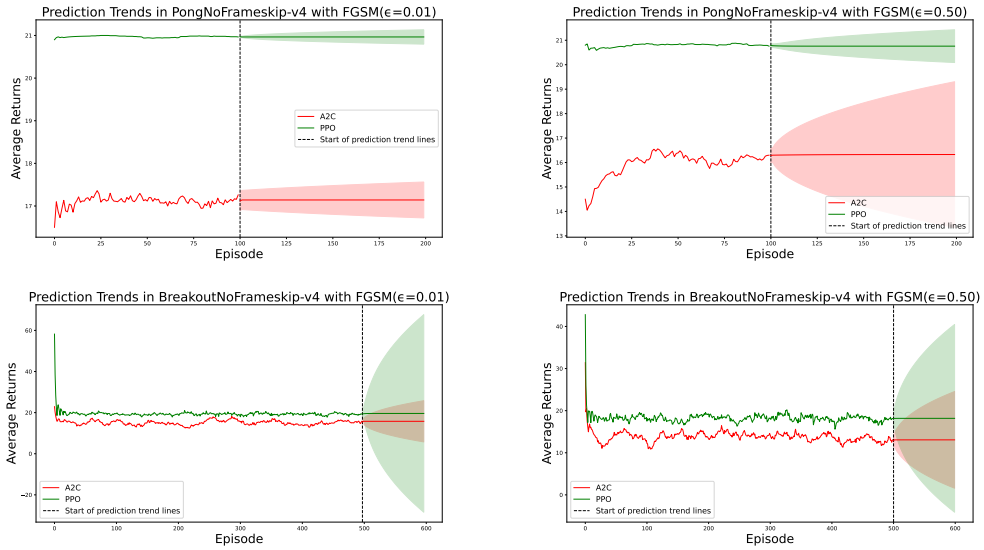


Figure 5: These plots take the rolling mean (window=25) of the observational performance up to a certain time point with some probability of being in the presence of adversarial attacks. After that time point, it shows the time series forecast with 99% prediction intervals. **Top Row:** In the PongNoFrameskip-v4 plots, PPO performs significantly better and has smaller prediction intervals. The plot on the right, however, uses attacks with higher ϵ, where both agents have larger prediction intervals. **Bottom Row:** In the BreakoutNoFrameskip-v4 plots, the prediction intervals overlap. One noticeable difference is that the stronger attacks cause the PPO interval to shrink in size while the mean rewards are slightly less. This decrease in variability accounts for the decrease in the maximum rewards achieved.

6 Conclusion

In this work, we show that relying on point estimates, even alternatives that are more reliable and less biased than what is typically used in practice, are limited in their ability to evaluate RL performance in the presence of distribution shifts. In particular, we argue that the methodology proposed here provides a necessary emphasis on test-time evaluation, and is general enough to assess both single and multi-agent performance. If, or when, we choose to deploy RL agents into the real world, how such agents perform after training will matter more than during training. The decline in performance during ad hoc agent switching in the PowerGridworld shows what could happen if we need to start replacing intelligent agents to ensure energy usage is minimized. Such experiments reveal a workflow that will likely be closer to representing real-world safety checks and maintenance of AI systems.

There is still work to accomplish in ensuring RL research is reproducible, like protocols for non-deterministic environments and developing new evaluation protocols at test time. Investigating if more complex time series models would be more appropriate in certain scenarios would be an obvious next step. Time series clustering and motifs [Mueen et al., 2009, Imani et al., 2021] can be used to better understand how different types of distribution shift can affect RL performance. One could also investigate if time series ensemble models [Wichard and Ogorzalek, 2004] provide better forecasts than simpler models.

Nations, like the United States [White House, 2022], have signed legislation to invest in modernization of infrastructure and communities to meet the growing challenges of the 21st century (e.g., cybersecurity and climate change). RL is advancing the automation of complex decision making in real-world infrastructure systems (e.g., energy and transportation). Like other areas in AI, the role of RL in critical infrastructure and other high-consequence applications requires robust, reliable, repeatable, and standardized evaluation protocols [European Union, 2021, Biden Jr, 2023]. The methodology we propose here, inspired by the NHTSA ratings standards, is a start toward a general, standardized protocol for evaluating pretrained RL performance over time.

Acknowledgments

This research was supported by the National Infrastructure Simulation and Analysis Center (NISAC), a program of the U.S. Cybersecurity and Infrastructure Security Agency's (CISA's) National Risk Management Center. Pacific Northwest National Laboratory is operated by Battelle Memorial Institute for the U.S. Department of Energy under Contract No. DE-AC05-76RL01830.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Nesreen K Ahmed, Amir F Atiya, Neamat El Gayar, and Hisham El-Shishiny. An empirical comparison of machine learning models for time series forecasting. *Econometric reviews*, 29(5-6): 594–621, 2010.

Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, Athul Paul Jacob, et al. Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, 2022.

Nikhil Barhate. Minimal pytorch implementation of proximal policy optimization. `https://github.com/nikhilbarhate99/PPO-PyTorch`, 2021.

Samuel Barrett and Peter Stone. Cooperating with unknown teammates in complex domains: A robot soccer case study of ad hoc teamwork. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

David Biagioni, Xiangyu Zhang, Dylan Wald, Deepthi Vaidhynathan, Rohit Chintala, Jennifer King, and Ahmed S. Zamzam. Powergridworld: A framework for multi-agent reinforcement learning in power systems. In *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, e-Energy '22, page 565–570, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393973. doi: 10.1145/3538637.3539616. URL `https://doi.org/10.1145/3538637.3539616`.

Joseph Biden Jr. Executive order on the safe, secure, and trustworthy development and use of artificial intelligence. *White House*, 2023. URL `https://www.whitehouse.gov/briefing-room/presidential-actions/2023/10/30/executive-order-on-the-safe-secure-and-trustworthy-development-and-use-of-artificial-intelligence/`. [Accessed November 2, 2023].

Battista Biggio, B Nelson, P Laskov, et al. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, pages 1807–1814. ArXiv e-prints, 2012.

Kay H Brodersen, Fabian Gallusser, Jim Koehler, Nicolas Remy, and Steven L Scott. Inferring causal impact using bayesian structural time-series models. *The Annals of Applied Statistics*, pages 247–274, 2015.

Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456): 885–890, 2019.

A Buteikis. Practical econometrics and data science. *Vilnius University: Vilnius, Lithuania*, 2020.

Stephen Casper, Dylan Hadfield-Menell, and Gabriel Kreiman. White-box adversarial policies in deep reinforcement learning. *arXiv preprint arXiv:2209.02167*, 2022.

Stephanie CY Chan, Samuel Fishman, Anoop Korattikara, John Canny, and Sergio Guadarrama. Measuring the reliability of reinforcement learning algorithms. In *International Conference on Learning Representations*, 2020.

Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. How many random seeds? statistical power analysis in deep reinforcement learning experiments. *arXiv preprint arXiv:1806.08295*, 2018.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

Jiaxun Cui, Xiaomeng Yang, Mulong Luo, Geunbae Lee, Peter Stone, Hsien-Hsin S Lee, Benjamin Lee, G Edward Suh, Wenjie Xiong, and Yuandong Tian. Macta: A multi-agent reinforcement learning approach for cache timing attacks and detection. In *The Eleventh International Conference on Learning Representations*, 2022.

Scott Cunningham. *Causal inference: The mixtape*. Yale university press, 2021.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep policy gradients: A case study on ppo and trpo. *arXiv preprint arXiv:2005.12729*, 2020.

European Union. Laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts. proposal for a regulation of the european parliament and of the council, 2021. URL `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206`.

Martin Figura, Krishna Chaitanya Kosaraju, and Vijay Gupta. Adversarial attacks in consensus-based multi-agent reinforcement learning. In *2021 American Control Conference (ACC)*, pages 3050–3055. IEEE, 2021.

Ted Fujimoto, Timothy Doster, Adam Attarian, Jill Brandenberger, and Nathan Hodas. Reward-free attacks in multi-agent reinforcement learning. *arXiv preprint arXiv:2112.00940*, 2021.

Everette S Gardner Jr and ED McKenzie. Forecasting trends in time series. *Management science*, 31 (10):1237–1246, 1985.

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*, 2019.

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

Rihab Gorsane, Omayma Mahjoub, Ruan John de Kock, Roland Dubb, Siddarth Singh, and Arnu Pretorius. Towards a standardised performance evaluation protocol for cooperative marl. *Advances in Neural Information Processing Systems*, 35:5510–5521, 2022.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Paul W Holland. Statistics and causal inference. *Journal of the American statistical Association*, 81 (396):945–960, 1986.

Charles C Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting*, 20(1):5–10, 2004.

Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and J Doug Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58, 2011.

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

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3): 90–95, 2007. doi: 10.1109/MCSE.2007.55.

Nick Huntington-Klein. *The effect: An introduction to research design and causality*. CRC Press, 2021.

Rob J Hyndman. Prediction intervals too narrow, Oct 2014. URL `https://robjhyndman.com/hyndsight/narrow-pi/`.

Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

Shima Imani, Alireza Abdoli, and Eamonn Keogh. Time2cluster: Clustering time series using neighbor information. In *Time Series Workshop at the 38th International Conference on Machine Learning (ICML)*, 2021.

Guido W Imbens and Donald B Rubin. *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press, 2015.

Scott Jordan, Yash Chandak, Daniel Cohen, Mengxue Zhang, and Philip Thomas. Evaluating the performance of reinforcement learning algorithms. In *International Conference on Machine Learning*, pages 4962–4973. PMLR, 2020.

Hoki Kim. Torchattacks: A pytorch repository for adversarial attacks. *arXiv preprint arXiv:2010.01950*, 2020.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

Jernej Kos and Dawn Song. Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452*, 2017.

Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations*, 2016.

Markus Löning, Franz Király, Tony Bagnall, Matthew Middlehurst, Sajaysurya Ganesh, George Oastler, Jason Lines, Martin Walter, ViktorKaz, Lukasz Mentel, chrisholder, Leonidas Tsaprounis, RNKuhns, Mirae Parker, Taiwo Owoseni, Patrick Rockenschaub, danbartl, jesellier, eenticott shell, Ciaran Gilbert, Guzal Bulatova, Lovkush, Patrick Schäfer, Stanislav Khrapov, Katie Buchhorn, Kejsi Take, Shivansh Subramanian, Svea Marie Meyer, AidenRushbrooke, and Beth rice. sktime/sktime: v0.13.4, September 2022. URL `https://doi.org/10.5281/zenodo.7117735`.

Yuzhe Ma, Xuezhou Zhang, Wen Sun, and Jerry Zhu. Policy poisoning in batch reinforcement learning and control. *Advances in Neural Information Processing Systems*, 32, 2019.

Ricardo P Masini, Marcelo C Medeiros, and Eduardo F Mendes. Machine learning advances for time series forecasting. *Journal of economic surveys*, 37(1):76–111, 2023.

McKinsey and Company, Mar 2021. URL `https://www.mckinsey.com/capabilities/mckinsey-digital/how-we-help-clients/flying-across-the-sea-propelled-by-ai`.

Reuth Mirsky, Ignacio Carlucho, Arrasy Rahman, Elliot Fosong, William Macke, Mohan Sridharan, Peter Stone, and Stefano V Albrecht. A survey of ad hoc teamwork research. In *Multi-Agent Systems: 19th European Conference, EUMAS 2022, Düsseldorf, Germany, September 14–16, 2022, Proceedings*, pages 275–293. Springer, 2022.

Raha Moraffah, Paras Sheth, Mansooreh Karami, Anchit Bhattacharya, Qianru Wang, Anique Tahir, Adrienne Raglin, and Huan Liu. Causal inference for time series analysis: Problems, methods and evaluation. *Knowledge and Information Systems*, 63:3041–3085, 2021.

Abdullah Mueen, Eamonn Keogh, Qiang Zhu, Sydney Cash, and Brandon Westover. Exact discovery of time series motifs. In *Proceedings of the 2009 SIAM international conference on data mining*, pages 473–484. SIAM, 2009.

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

Judea Pearl. *Causality*. Cambridge university press, 2009.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

Joaquin Quinonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. Mit Press, 2008.

Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. Failing loudly: An empirical study of methods for detecting dataset shift. *Advances in Neural Information Processing Systems*, 32, 2019.

Antonin Raffin. Rl baselines3 zoo. `https://github.com/DLR-RM/rl-baselines3-zoo`, 2020.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL `http://jmlr.org/papers/v22/20-1364.html`.

Muhammad A Rahman, Niklas Hopner, Filippos Christianos, and Stefano V Albrecht. Towards open ad hoc teamwork using graph-based policy learning. In *International Conference on Machine Learning*, pages 8776–8786. PMLR, 2021.

Amin Rakhsha, Goran Radanovic, Rati Devidze, Xiaojin Zhu, and Adish Singla. Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning. In *International Conference on Machine Learning*, pages 7974–7984. PMLR, 2020.

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.

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

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419): 1140–1144, 2018.

Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causation, prediction, and search*. MIT press, 2000.

Peter Stone, Gal Kaminka, Sarit Kraus, and Jeffrey Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 1504–1509, 2010.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60): 3021, 2021. doi: 10.21105/joss.03021. URL `https://doi.org/10.21105/joss.03021`.

White House. A guidebook to the bipartisan infrastructure law for state, local, tribal, and territorial governments, and other partners. *White House*, 2022. URL `https://www.whitehouse.gov/wp-content/uploads/2022/05/BUILDING-A-BETTER-AMERICA-V2.pdf`. [Accessed November 2, 2023].

Jorg D Wichard and Maciej Ogorzalek. Time series prediction with ensemble models. In *2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541)*, volume 2, pages 1625–1630. IEEE, 2004.

Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.

Xuezhou Zhang, Yuzhe Ma, Adish Singla, and Xiaojin Zhu. Adaptive reward-poisoning attacks against reinforcement learning. In *International Conference on Machine Learning*, pages 11225–11234. PMLR, 2020.

Appendix

A Further Notes on the RL Fixed Seed Assumption

Reproducibility can be difficult to achieve when running on a GPU. As described in PyTorch's webpage on reproducibility, even identical seeds might not provide reproducible results. Some reasons include nondeterministic algorithms that improve performance and the use of different hardware can affect the selection of such algorithms. To control these sources of randomness in our experiments, we adhere to the reproducibility suggestions provided on the webpage.

B Further Notes on Environments Used in Evaluations

All plots use 10 random seeds. In multi-agent settings, each agent shares the same seed during an evaluation run. For example, if our seed numbers are 1 and 42, then the first evaluation run sets

B.1 Atari Games

The games of focus are a subset of Atari games AsteroidsNoFrameskip-v4, BeamRiderNoFrameskip-v4, BreakoutNoFrameskip-v4, MsPacmanNoFrameskip-v4, PongNoFrameskip-v4, QbertNoFrameskip-v4, RoadRunnerNoFrameskip-v4, SeaquestNoFrameskip-v4, and SpaceInvadersNoFrameskip-v4. The agents we evaluate are pretrained agents from RL Baselines3 Zoo [Raffin, 2020] that are available on SB3's Huggingface repository of models. The adversarial attacks (FGSM) were implemented in torchattacks [Kim, 2020].

B.2 PowerGridworld

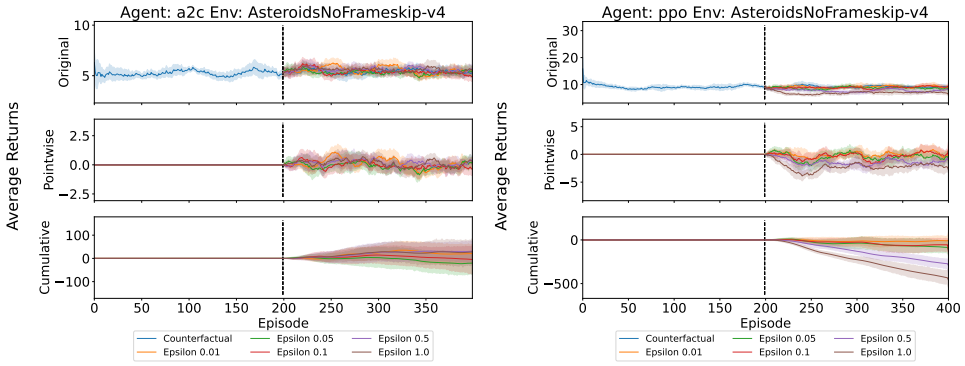
PowerGridworld is a modular, customizable framework for building power systems environments to train RL agents. Because of this, we use an environment provided in one of the example scripts. The class name is called `CoordinatedMultiBuildingControlEnv`, which is a multi-agent coordination environment. In addition to the original agent-level reward, grid-level reward/penalty and system-level constraint(s) are considered. In particular, we consider the voltage constraints: agents need to coordinate so the common bus voltage is within the ANSI C.84.1 limit. If the constraints are not satisfied, the voltage violation penalty will be shared by all agents. The agents in this scenario are minimal implementations [Barhate, 2021] of PPO.

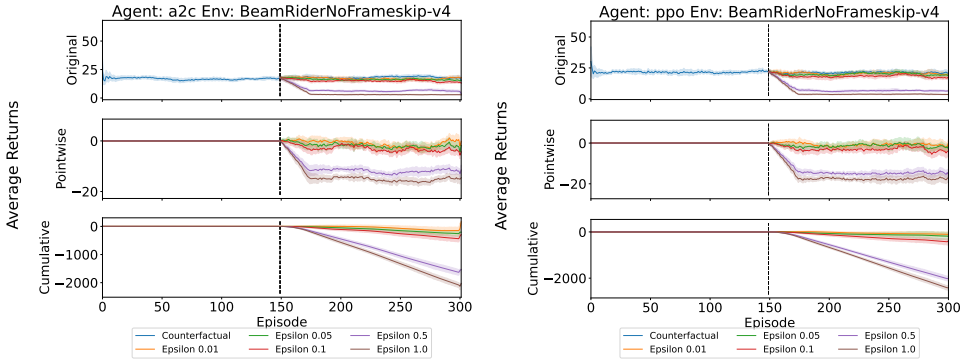
B.3 Time Series Tools

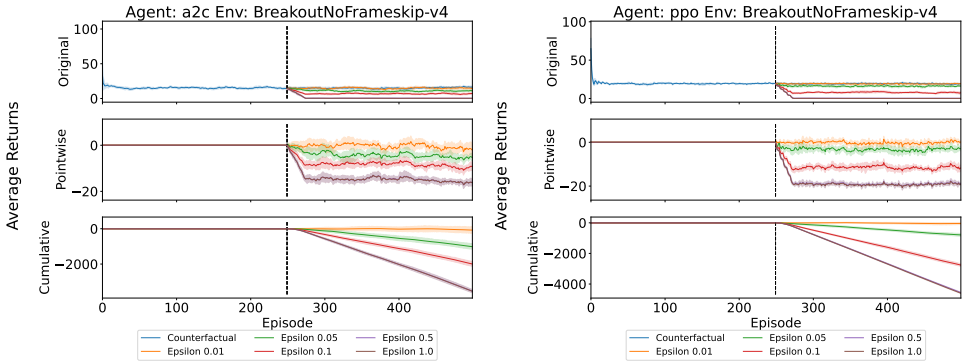
Trends and prediction intervals were implemented in the Python package sktime [Löning et al., 2022]. Other Python visualization tools include Matplotlib [Hunter, 2007] and Seaborn [Waskom, 2021].

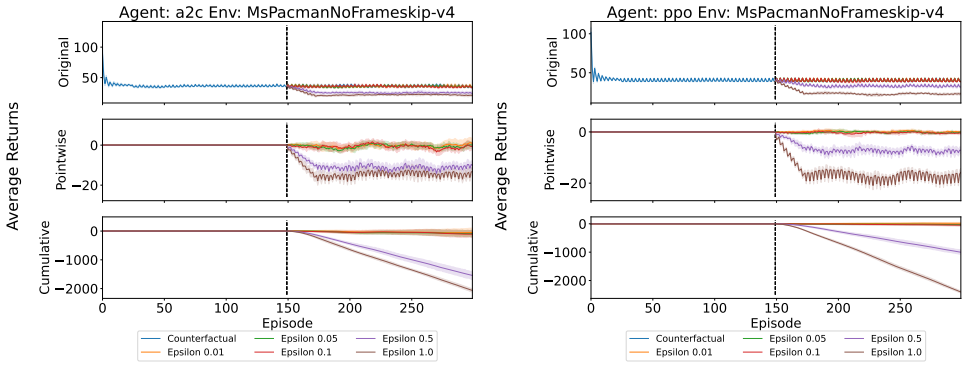
C More Plots

C.1 Atari Game Causal Impact Plots

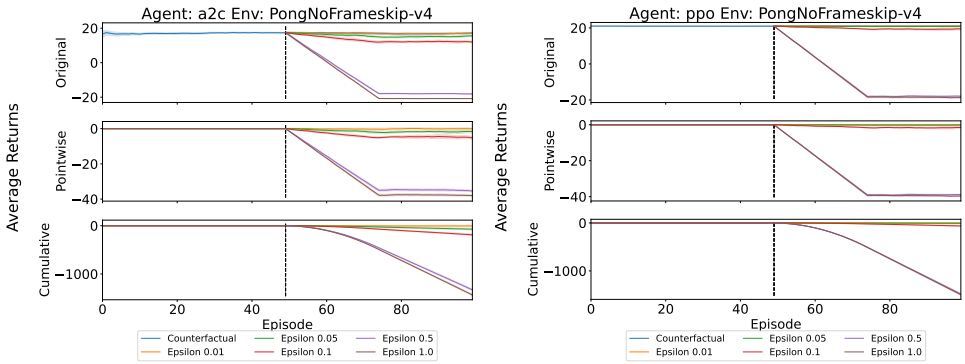




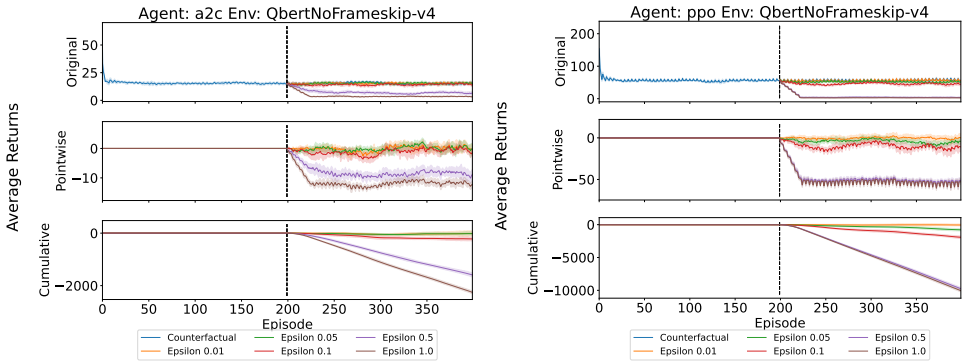



Agent: a2c Env: MsPacmanNoFrameskip-v4
Average Returns
Original
Pointwise
Cumulative
Episode
Counterfactual
Epsilon 0.01
Epsilon 0.05
Epsilon 0.1
Epsilon 0.5
Epsilon 1.0

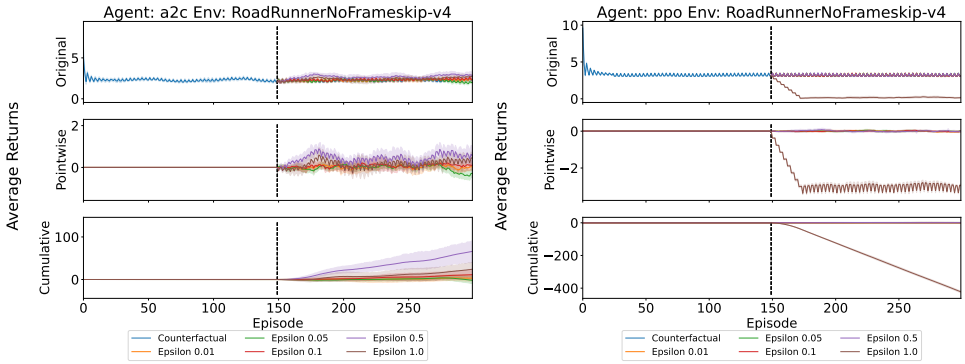
Agent: ppo Env: MsPacmanNoFrameskip-v4
Average Returns
Original
Pointwise
Cumulative
Episode
Counterfactual
Epsilon 0.01
Epsilon 0.05
Epsilon 0.1
Epsilon 0.5
Epsilon 1.0


Agent: a2c Env: PongNoFrameskip-v4
Average Returns
Original
Pointwise
Cumulative
Episode
Counterfactual
Epsilon 0.01
Epsilon 0.05
Epsilon 0.1
Epsilon 0.5
Epsilon 1.0

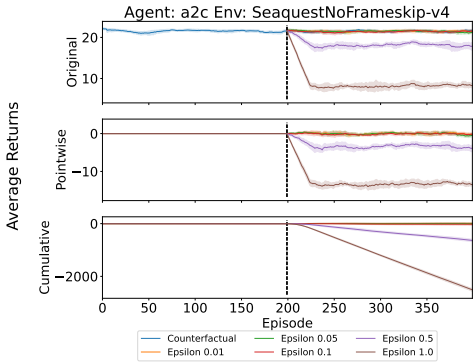
Agent: ppo Env: PongNoFrameskip-v4
Average Returns
Original
Pointwise
Cumulative
Episode
Counterfactual
Epsilon 0.01
Epsilon 0.05
Epsilon 0.1
Epsilon 0.5
Epsilon 1.0

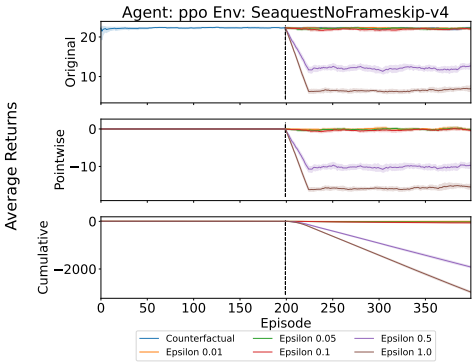

Agent: a2c Env: QbertNoFrameskip-v4
Average Returns
Original
Pointwise
Cumulative
Episode
Counterfactual
Epsilon 0.01
Epsilon 0.05
Epsilon 0.1
Epsilon 0.5
Epsilon 1.0

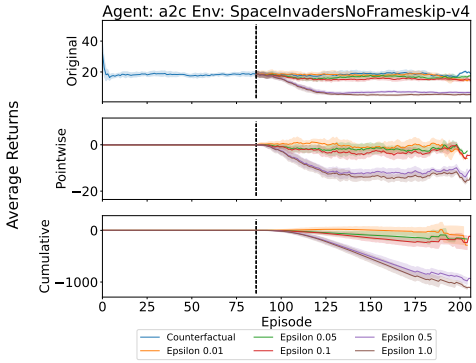
Agent: ppo Env: QbertNoFrameskip-v4
Average Returns
Original
Pointwise
Cumulative
Episode
Counterfactual
Epsilon 0.01
Epsilon 0.05
Epsilon 0.1
Epsilon 0.5
Epsilon 1.0

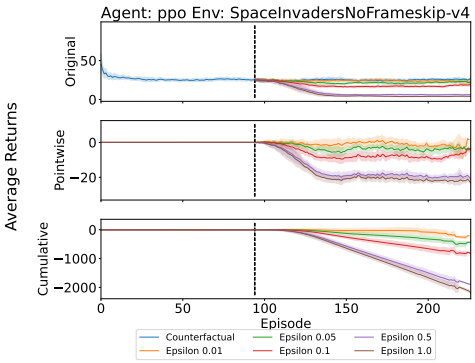

Agent: a2c Env: RoadRunnerNoFrameskip-v4
Average Returns
Original
Pointwise
Cumulative
Episode
Counterfactual
Epsilon 0.01
Epsilon 0.05
Epsilon 0.1
Epsilon 0.5
Epsilon 1.0

Agent: ppo Env: RoadRunnerNoFrameskip-v4
Average Returns
Original
Pointwise
Cumulative
Episode
Counterfactual
Epsilon 0.01
Epsilon 0.05
Epsilon 0.1
Epsilon 0.5
Epsilon 1.0


Agent: a2c Env: SeaquestNoFrameskip-v4
Average Returns
Original
Pointwise
Cumulative
Episode
Counterfactual
Epsilon 0.01
Epsilon 0.05
Epsilon 0.1
Epsilon 0.5
Epsilon 1.0


Agent: ppo Env: SeaquestNoFrameskip-v4
Average Returns
Original
Pointwise
Cumulative
Episode
Counterfactual
Epsilon 0.01
Epsilon 0.05
Epsilon 0.1
Epsilon 0.5
Epsilon 1.0


Agent: a2c Env: SpaceInvadersNoFrameskip-v4
Average Returns
Original
Pointwise
Cumulative
Episode
Counterfactual
Epsilon 0.01
Epsilon 0.05
Epsilon 0.1
Epsilon 0.5
Epsilon 1.0


Agent: ppo Env: SpaceInvadersNoFrameskip-v4
Average Returns
Original
Pointwise
Cumulative
Episode
Counterfactual
Epsilon 0.01
Epsilon 0.05
Epsilon 0.1
Epsilon 0.5
Epsilon 1.0

C.2 Observational Plots

All observational plots show forecasts with prediction intervals 100 episodes after the last measurement.

C.2.1 PowerGridworld

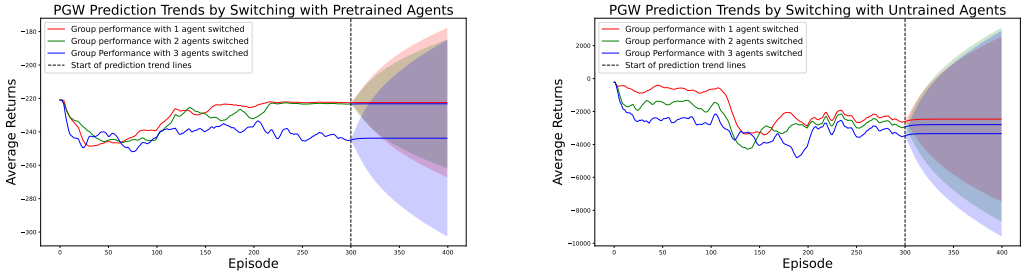
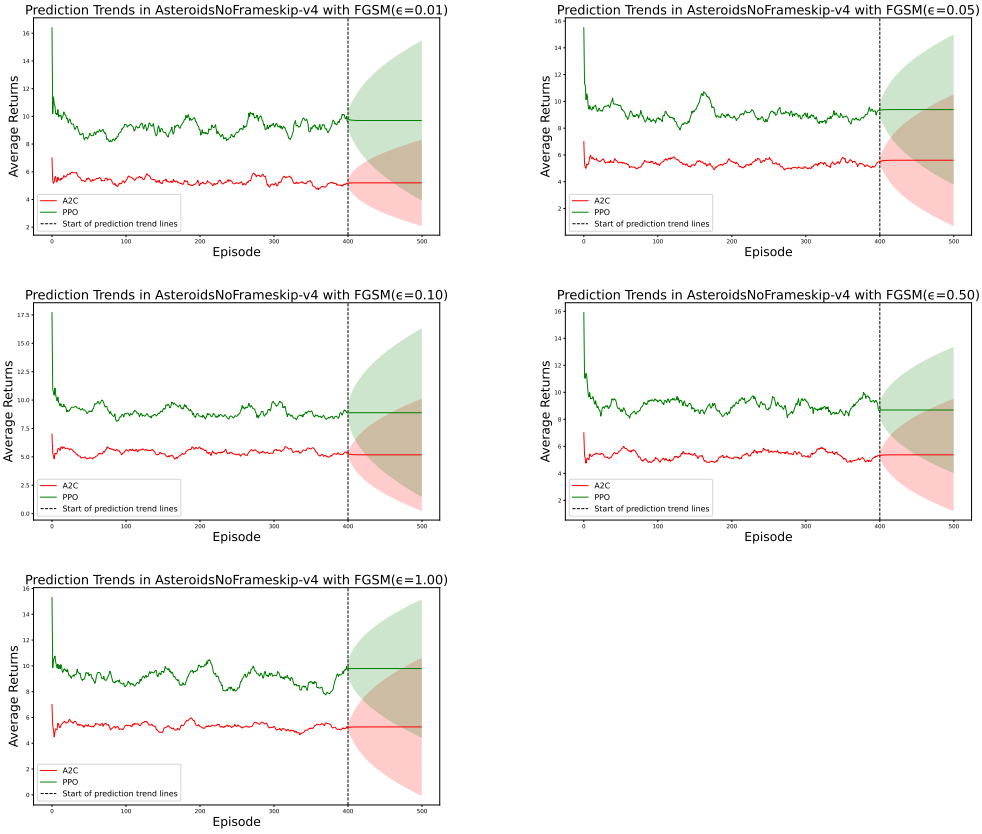
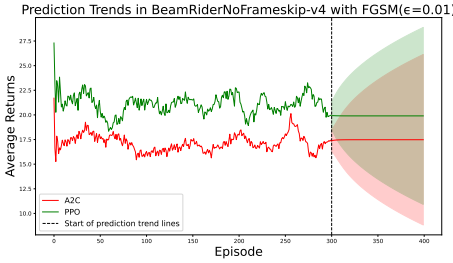
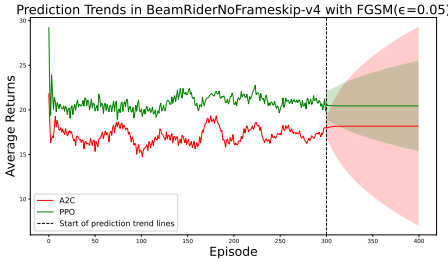


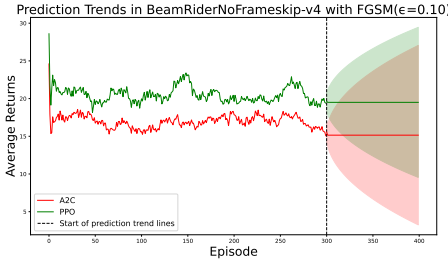
Figure 6: PowerGridworld observational plots. **Left:** Comparing random switching with pretrained agents at each episode. **Right:** Comparing random switching with untrained agented at each episode.

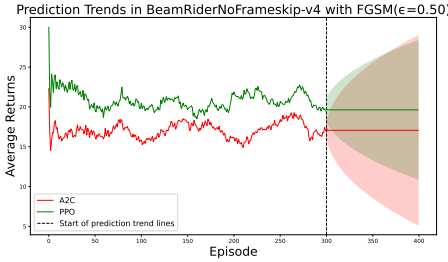
C.2.2 Atari Games

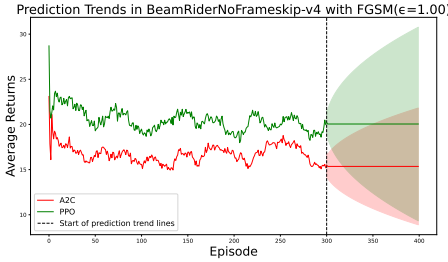


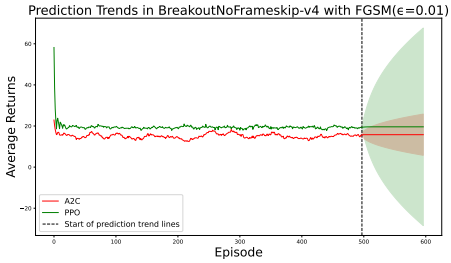

Prediction Trends in BeamRiderNoFrameskip-v4 with FGSM(ϵ=0.01)
Average Returns
Episode
A2C
PPO
Start of prediction trend lines

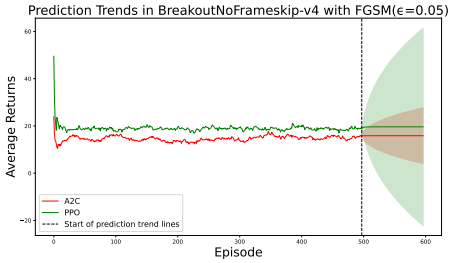

Prediction Trends in BeamRiderNoFrameskip-v4 with FGSM(ϵ=0.05)
Average Returns
Episode
A2C
PPO
Start of prediction trend lines

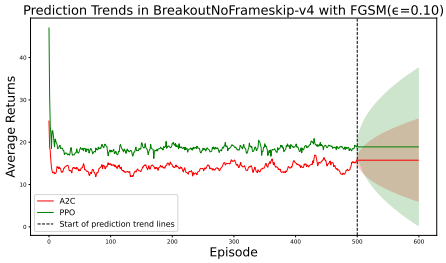

Prediction Trends in BeamRiderNoFrameskip-v4 with FGSM(ϵ=0.10)
Average Returns
Episode
A2C
PPO
Start of prediction trend lines

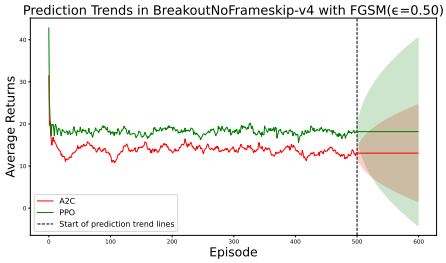

Prediction Trends in BeamRiderNoFrameskip-v4 with FGSM(ϵ=0.50)
Average Returns
Episode
A2C
PPO
Start of prediction trend lines

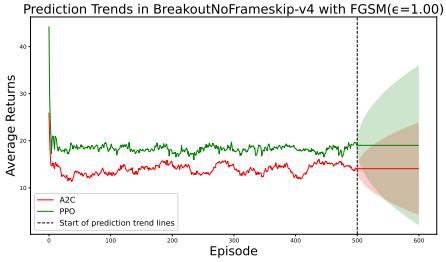

Prediction Trends in BeamRiderNoFrameskip-v4 with FGSM(ϵ=1.00)
Average Returns
Episode
A2C
PPO
Start of prediction trend lines

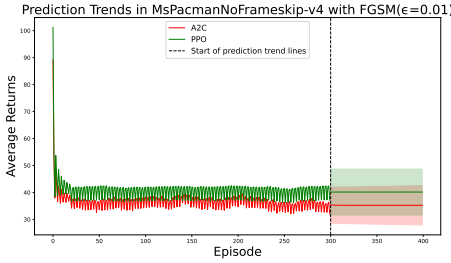

Prediction Trends in BreakoutNoFrameskip-v4 with FGSM(ϵ=0.01)
Average Returns
Episode
A2C
PPO
Start of prediction trend lines

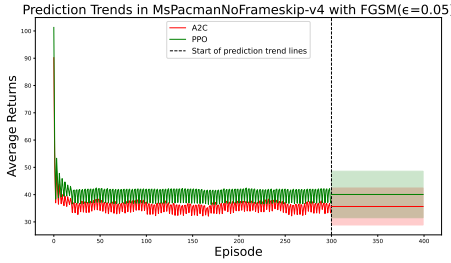

Prediction Trends in BreakoutNoFrameskip-v4 with FGSM(ϵ=0.05)
Average Returns
Episode
A2C
PPO
Start of prediction trend lines

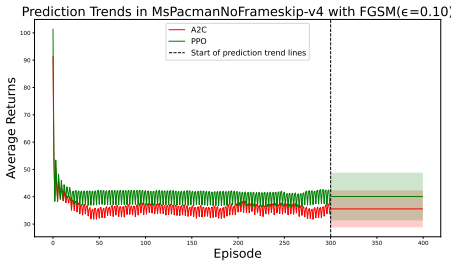

Prediction Trends in BreakoutNoFrameskip-v4 with FGSM(ϵ=0.10)
Average Returns
Episode
A2C
PPO
Start of prediction trend lines

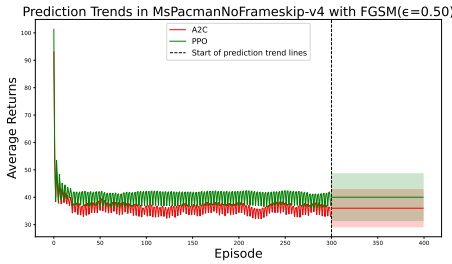

Prediction Trends in BreakoutNoFrameskip-v4 with FGSM(ϵ=0.50)
Average Returns
Episode
A2C
PPO
Start of prediction trend lines

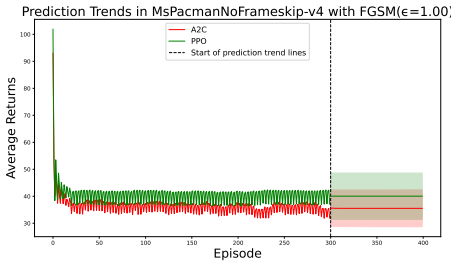

Prediction Trends in BreakoutNoFrameskip-v4 with FGSM(ϵ=1.00)
Average Returns
Episode
A2C
PPO
Start of prediction trend lines

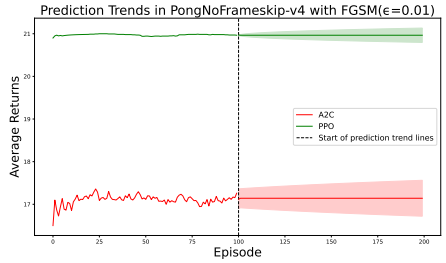

Prediction Trends in MsPacmanNoFrameskip-v4 with FGSM(ϵ=0.01)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

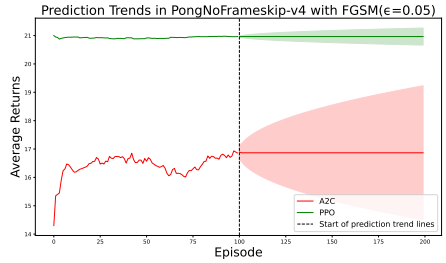

Prediction Trends in MsPacmanNoFrameskip-v4 with FGSM(ϵ=0.05)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

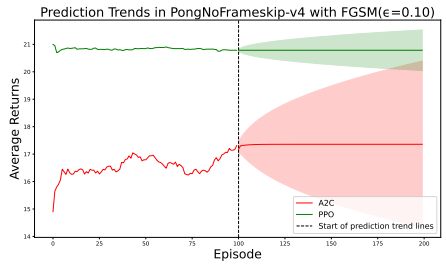

Prediction Trends in MsPacmanNoFrameskip-v4 with FGSM(ϵ=0.10)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

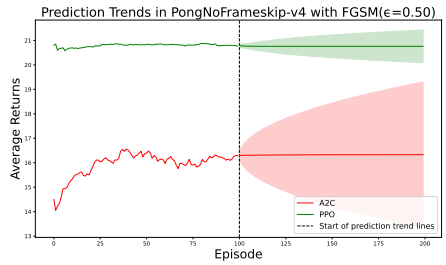

Prediction Trends in MsPacmanNoFrameskip-v4 with FGSM(ϵ=0.50)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

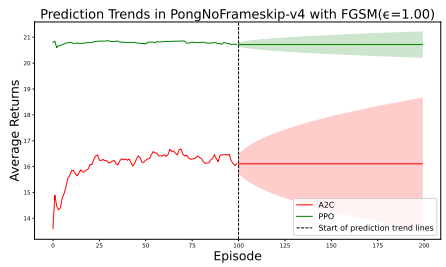

Prediction Trends in MsPacmanNoFrameskip-v4 with FGSM(ϵ=1.00)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

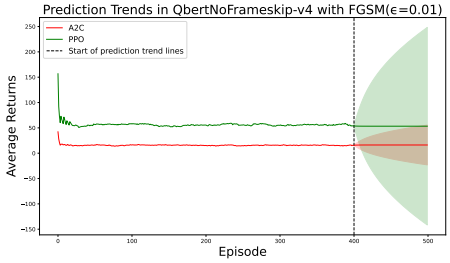

Prediction Trends in PongNoFrameskip-v4 with FGSM(ϵ=0.01)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

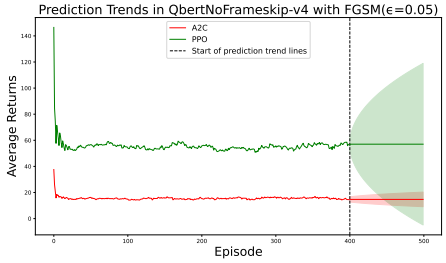

Prediction Trends in PongNoFrameskip-v4 with FGSM(ϵ=0.05)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

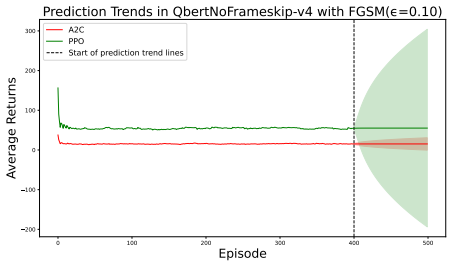

Prediction Trends in PongNoFrameskip-v4 with FGSM(ϵ=0.10)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

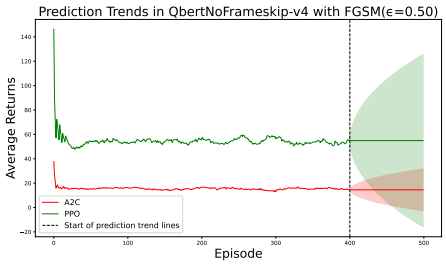

Prediction Trends in PongNoFrameskip-v4 with FGSM(ϵ=0.50)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

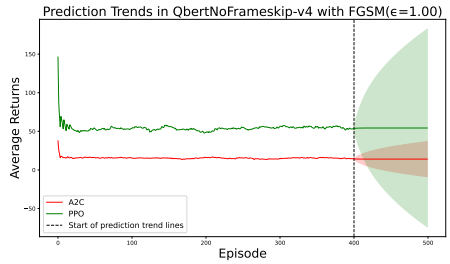

Prediction Trends in PongNoFrameskip-v4 with FGSM(ϵ=1.00)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

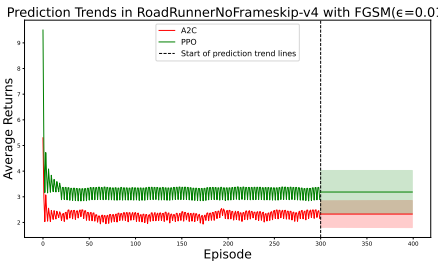

Prediction Trends in QbertNoFrameskip-v4 with FGSM(ϵ=0.01)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

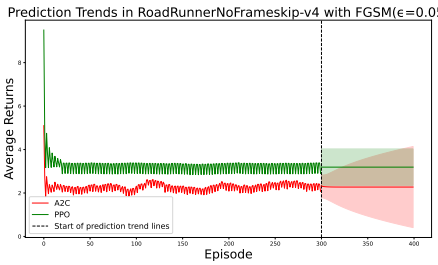

Prediction Trends in QbertNoFrameskip-v4 with FGSM(ϵ=0.05)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

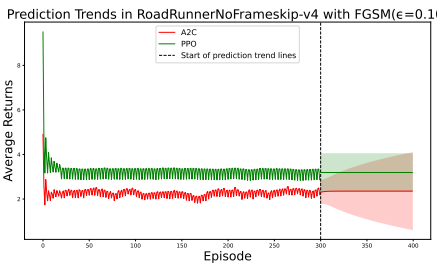

Prediction Trends in QbertNoFrameskip-v4 with FGSM(ϵ=0.10)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

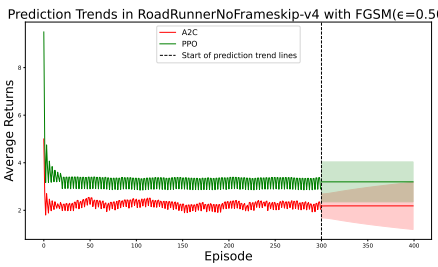

Prediction Trends in QbertNoFrameskip-v4 with FGSM(ϵ=0.50)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

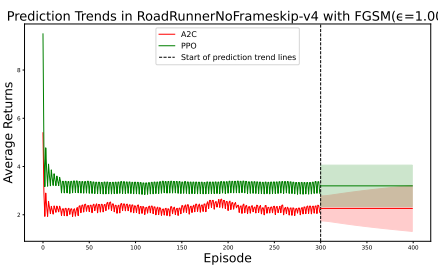

Prediction Trends in QbertNoFrameskip-v4 with FGSM(ϵ=1.00)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

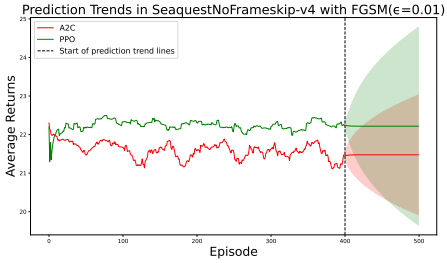

Prediction Trends in RoadRunnerNoFrameskip-v4 with FGSM(ϵ=0.01)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

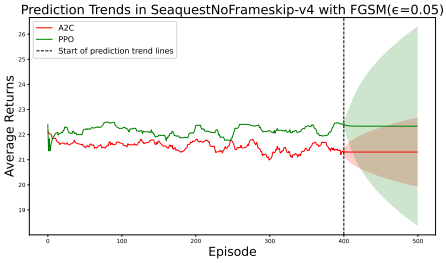

Prediction Trends in RoadRunnerNoFrameskip-v4 with FGSM(ϵ=0.05)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

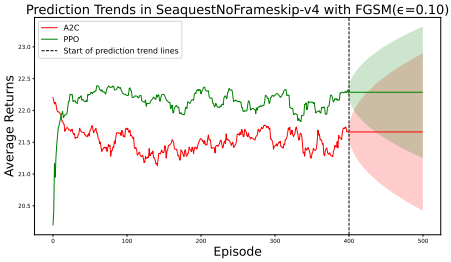

Prediction Trends in RoadRunnerNoFrameskip-v4 with FGSM(ϵ=0.10)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

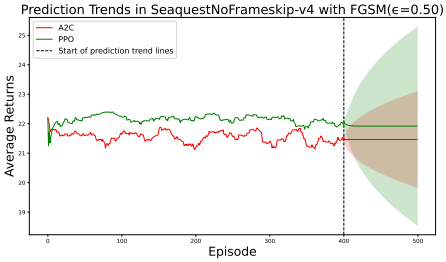

Prediction Trends in RoadRunnerNoFrameskip-v4 with FGSM(ϵ=0.50)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

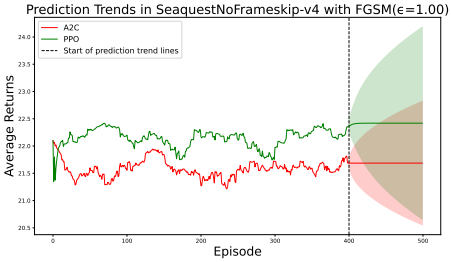

Prediction Trends in RoadRunnerNoFrameskip-v4 with FGSM(ϵ=1.00)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

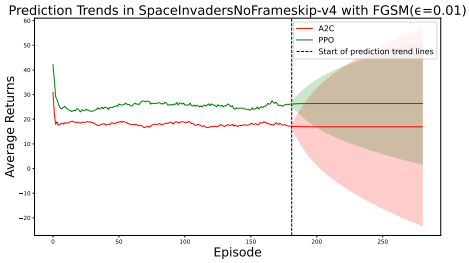

Prediction Trends in SeaquestNoFrameskip-v4 with FGSM(ϵ=0.01)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

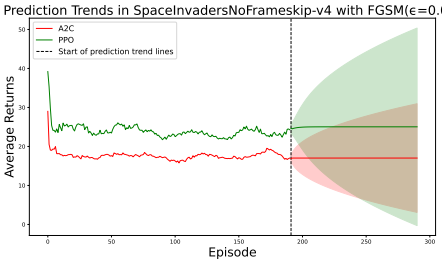

Prediction Trends in SeaquestNoFrameskip-v4 with FGSM(ϵ=0.05)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

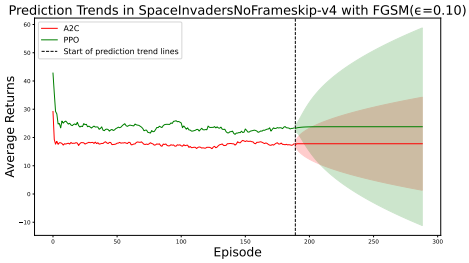

Prediction Trends in SeaquestNoFrameskip-v4 with FGSM(ϵ=0.10)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

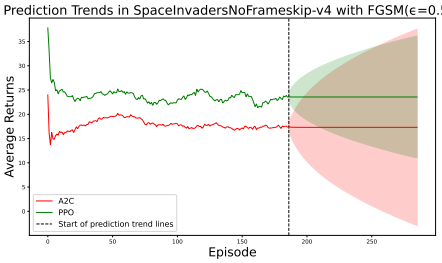

Prediction Trends in SeaquestNoFrameskip-v4 with FGSM(ϵ=0.50)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode

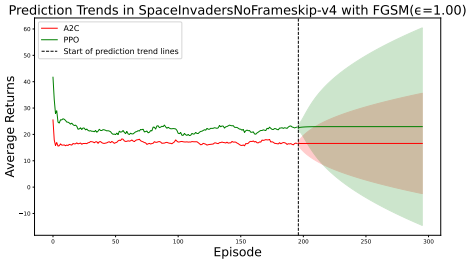

Prediction Trends in SeaquestNoFrameskip-v4 with FGSM(ϵ=1.00)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode


Prediction Trends in SpaceInvadersNoFrameskip-v4 with FGSM(ϵ=0.01)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode


Prediction Trends in SpaceInvadersNoFrameskip-v4 with FGSM(ϵ=0.05)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode


Prediction Trends in SpaceInvadersNoFrameskip-v4 with FGSM(ϵ=0.10)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode


Prediction Trends in SpaceInvadersNoFrameskip-v4 with FGSM(ϵ=0.50)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode


Prediction Trends in SpaceInvadersNoFrameskip-v4 with FGSM(ϵ=1.00)
A2C
PPO
Start of prediction trend lines
Average Returns
Episode