Safe Reinforcement Learning via Observation Shielding

Joe McCalmon
Computer Science Department
Wake Forest University
Winston-Salem, NC, USA
mccalmonjoe@gmail.com

Tongtong Liu
Computer Science Department
Wake Forest University
Winston-Salem, NC, USA
liut18@wfu.edu

Reid Goldsmith
Computer Science Department
Wake Forest University
Winston-Salem, NC, USA
goldrw20@wfu.edu

Andrew Cyhaniuk
Computer Science Department
Wake Forest University
Winston-Salem, NC, USA
cyhaap20@wfu.edu

Talal Halabi
Department of Computer Science
Laval University
QC, Canada
talal.halabi@ift.ulaval.ca

Sarra Alqahtani
Computer Science Department
Wake Forest University
Winston-Salem, NC, USA
alqahtas@wfu.edu

Abstract

Reinforcement Learning (RL) algorithms have shown success in scaling up to large problems. However, deploying those algorithms in real-world applications remains challenging due to their vulnerability to adversarial perturbations. Existing RL robustness methods against adversarial attacks are weak to large perturbations - a scenario that cannot be ruled out for RL adversarial threats, as is the case for deep neural networks in classification tasks. This paper proposes a method called observation-shielding RL (OSRL) to increase the robustness of RL against large perturbations using predictive models and threat detection. Instead of changing the RL algorithms with robustness regularization or retrain them with adversarial perturbations, we depart considerably from previous approaches and develop an add-on safety feature for existing RL algorithms during runtime. OSRL builds on the idea of model predictive shielding, where an observation predictive model is used to override the perturbed observations as needed to ensure safety. Extensive experiments on various MuJoco[1] environments (Ant, Hoope, InvertedPendulum, Reacher) environment demonstrate that our proposed OSRL is safer and more efficient than state-of-the-art robustness methods under large perturbations.

Keywords:

Reinforcement learning, adversarial examples, safety, robustness, shielding.

1. Introduction

Recent years have witnessed significant advances in reinforcement learning (RL), an area of machine learning that achieved great success in solving various sequential decision-making problems, especially with the development of deep neural networks (DNN) for function approximation. Most of the affected applications—from the games of Go and Poker to robotics—involve more than one agent. However, DNNs are susceptible to adversarial attacks, especially those which rely on perturbing the inputs to the network Papernot et al. (2015), Goodfellow et al. (2015), Szegedy et al. (2014). In many RL applications, the input to the agent's policy, parameterized by a DNN, can be adversarially perturbed using existing methods such as FGSM (Fast Gradient Signed Method) Goodfellow et al. (2015) and JSMA (Jacobian-based Saliency Map Attack) Wiyatno and Xu (2018). Such perturbations decrease the agent's performance Huang et al. (2017), Lin et al. (2019). With RL's major implications in Artificial Intelligence (AI) applications ranging from computer games to autonomous vehicles, the safety and security of these algorithms are critical.

In response to adversarial attacks as well as general noise to policy inputs, recent research has focused on improving the robustness of RL agents. Specifically, there are two directions in this realm: adversarial training and robustness regularization. The findings of adversarial training in supervised learning Goodfellow et al. (2015), Rajeswaran et al. (2017) have been applied to the RL field but have demonstrated unstable results in terms of performance both with and without the presence of an adversary Pattanaik et al. (2017). Recently, Zhang et al. (2021) formulated the state-adversarial Marko Decision Processes MDP (SA-MDP), which follows the thinking of worst-case to regularize RL algorithms. SA-MDP is able to recover most of the lost reward due to perturbations in RL tasks in Pong and MuJoco environments. To the best of our knowledge, SA-MDP is the current best method at developing an RL agent which is robust against small

[1] https://www.endtoend.ai/envs/gym/mujoco

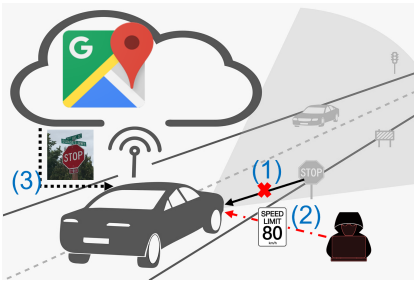

Figure 1: An example of using OSRL in self-driving cars. The following steps are depicted: (1) the car's sensor is reading the current environment state as a stop sign; (2) the adversary intercepts the sensor's reading and replaces it with a speed limit sign; and (3) OSRL's prediction model predicts the state as a stop sign using GPS maps.

perturbations (i.e., ϵ-perturbation budget Goodfellow et al. (2015)) to the policy inputs. However, as is the case with most robust RL methods, robustness is built within the algorithms' objective functions and involves re-training for each ϵ values. Furthermore, robustness in SA-MDP can only be proved for a small amount of perturbation, denoted by ϵ.

In image classification tasks, the ϵ-robustness is justified since any larger amount of perturbation could be recognized by a human observer Goodfellow et al. (2015). In RL tasks, however, a human may not have consistent access to the inputs provided to the RL agent, and even if they did, inputs in many tasks are not as recognizable as an image. Even when the inputs are images, the adversary perturbs the internal observation of the agent, not the ground-truth state, hence a human-observer would not be able to detect this easily. As a result, it is important to develop an algorithm which is robust not only to small and limited ϵ-perturbations, but stronger perturbations as well.

This paper proposes Observation-Shielding Reinforcement Learning (OSRL), the first sample-efficient model-based safety method that can be used to mitigate large adversarial perturbations on the RL observation. OSRL provides a predicted observation based on the underlying environment dynamics for the agent to continue working even when targeted by strong attacks. Although the observation prediction is a difficult problem, an accurate prediction model can be developed under the assumption that system dynamics are known or can be accurately modeled. For instance, a self-driving car can build a model for the environment dynamics (i.e., traffic signs) using GPS maps such as Google Maps. Figure 1 shows an example of using OSRL in a self-driving car environment. We will study the robustness of Deep Reinforcement Learning (DRL) agents in a more challenging setting where the agent has continuous actions and its observations are strongly perturbed for a long period of time.

OSRL is a two-step algorithmic framework. In the first step, we leverage the dynamics modeling from the model-based RL field to build an action-conditioned predictive transition model that predicts the current observation based on the previous observation, action, and the learned environment dynamics. Using this model, we develop a detection method that compares the physical observation received from the environment (x_t) and the predicted observation generated by the prediction model ($\hat{x_t}$). When adversarial perturbations are induced in the current observation, the two values will be different by a certain threshold, and their distantness indicates the presence of adversarial attacks. In the second step of OSRL, once adversarial perturbations are detected, the agent's policy will be adapted to use the predicted observation instead of the real one to decide the optimal action.

To summarize, we build an agnostic, black-box approach that can be utilized on top of any RL algorithm to mitigate the risk of strong adversarial perturbations to the agent's observations. Our contributions are outlined as follows:

- We extend the Markov Decision Process (MDP) to formulate the perturbation on the agent's observations by creating the Observation-Shielding MDP (OS-MDP), and study the main conceptual differences between MDP, OS-MDP, and partially-observable MDP (POMDP).

- To solve OS-MDP, we develop a general two-step robustness algorithm called OSRL, which can be directly applied to the commonly-used threat model of observation manipulation but with higher ϵ than the state-of-the-art robustness methods. OSRL can be practically and efficiently applied as an add-on defense mechanism to any RL algorithm.

- We study the effectiveness of our proposed model-based robustness method compared to model-free robustness baselines based on adversarial training and the worst-case robustness regularization. We show that our OSRL can preserve the agent's performance in numerous MuJoco domains by up to 50% in terms of the total reward.

The remainder of this paper is organized as follows. We discuss the background and related work on RL robustness and safety in Section 2. Section 3 formulates the threat model and introduces the proposed approach. Section 4 presents our experiments and discusses the results. Final remarks and conclusions are outlined in Section 6.

2. Background and Related Work

This section presents some important concepts related to adversarial attacks on RL algorithms and discusses the literature on robustness methods, allowing to fully understand the proposed OSRL approach.

2.1. Reinforcement Learning

An RL agent learns by acting within an environment. Its interaction can be characterized as an MDP described by the tuple (S, A, P, R), where S is the set of states, A is the set of actions available to the agent, P is the transition function such that $P(s_t, a_t)$ produces a distribution over all possible next states at time t, and R is the set of all possible rewards that an agent can receive for actions taken. The goal of an agent is to learn the policy, $\pi(s_t) = a_t$, which maximizes the total discounted reward over the whole task. To help discover the optimal policy, an agent learns to approximate the value function, $v(s_i)$, which is the expected discounted reward that can be gained by being in state s_i and following the policy π. Formally, the value function is defined by:

$$v(s_i) = \mathbb{E}\left(\sum_{t=i} \gamma^t R(s_t, \pi(s_t), s_{t+1}) \right) \tag{1}$$

2.2. Adversarial Perturbation in Image Classification: FGSM

Image classifiers trained with DNNs misclassify examples that are only slightly perturbed from correctly classified examples drawn from the data distribution. For images, such perturbations are often imperceptible to the human vision system, yet they completely fool the deep learning models. One such attack used often on image classifiers is known as the Fast Gradient Sign Method (FGSM). Goodfellow et al. (2015) proved that by adding an imperceptibly small vector (noise) whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, many models will misclassify the input image using $x_{\text{adversarial}} = \epsilon * sign(\Delta_x J(\Theta, x, y))$.

To illustrate the adversarial examples through FGSM, consider the images in Figure 2, potentially consumed by an autonomous vehicle Lopez-Montiel et al. (2019) and Stallkamp et al. (2012). To the human vision system, the difference between those images are imperceptible; both show a stop sign. However, DNN models misclassify the right image as a yield sign, as described in Lopez-Montiel et al. (2019). In fact, the right image was modified by adding a precise perturbation to the left one. Attackers could potentially use the altered image to cause an autonomous vehicle to behave dangerously by maliciously modifying the sign itself, e.g., with stickers or paint Papernot et al. (2016).

2.3. Adversarial Perturbations in RL

Though the FGSM attack is focused on image perturbations, it carries over to all deep learning-based algorithms such as DRL Huang et al. (2017), Lin et al. (2019), and Russo and Proutiere (2019). Researchers in Huang et al. (2017) attack the RL algorithms using FGSM by perturbing the states every single time step in a given episode. They analyzed three types of DRL algorithms including Deep Q-networks (DQN) Mnih et al. (2013), Trust Region Policy Optimization (TRPO) Schulman et al. (2017), and A3C Mnih et al. (2016) and concluded that the reward significantly dropped with all three algorithms, proving the vulnerability of RL to adversarial attacks.

Following Huang et al. (2017), the work in Russo and Proutiere (2019) models the selection of attacking time steps as an MDP. By solving this MDP, an attacker could identify the optimal time steps to launch attacks and thus minimize the attack detection rate. In Lin et al. (2019), two novel types of attack on a single RL agent system were introduced: the strategically-timed attack and the enchanting attack. The goal of the former is to reduce the agent's reward by only feeding adversarial examples to the DNN of the agent for a selected small subset of timesteps in a certain episode. In the enchanting attack, the researcher exploits the fact that each action taken by the agent influences its future observations. Therefore, the adversary could plan a sequence of adversarial examples to maliciously lure the agent toward a dangerous state. A detailed survey regarding the adversarial perturbations on different RL settings can be found in Xiao et al. (2019).

2.4. Adversarial Training

Adversarial training as a defense mechanism consists in incorporating adversarial perturbations during training to improve the robustness of the agent. Different techniques have been developed for adversarial training in RL Yuan et al. (2018). In Mandlekar et al. (2017), the adversarial training has been applied using simple FGSM adversarial examples on policy gradient algorithms, which have been tested on simple RL tasks. The projected gradient descent (PGD) attack was used in Pattanaik et al. (2017) to generate more complicated adversarial examples in training RL agents on Atari games. However, the results demonstrated a trade-off between the agent's robustness against adversarial perturbations and its task performance. Moreover, training RL agents with PGD adversarial examples often incurs much higher computation cost than regular training. Other adversarial training methods have achieved mixed results Rajeswaran et al. (2017), Fischer et al. (2019).

One major limitation of adversarial training is its failure against the transferability property of adversarial

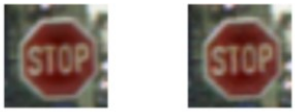

Figure 2: Example of adversarial images Papernot et al. (2016).

examples Tramer et al. (2020), that is, when adversarial examples generated for one agent would still successfully attack another agent even if the latter has been trained using adversarial training. Korkmaz (2022) studies the correlation between the direction of high sensitivity to adversarial perturbations across different training algorithms, environments, and states. It also investigates which features in the state space that are more sensitive to the adversarial perturbations and if they are the same across different algorithms and environments. The results proves the transferability of the perturbations between DRL algorithms which invalidates using adversarial training as a defense mechanism against adversarial perturbations as we stated in our paper. This paper's purpose is different from OSRL's of developing a defense/mitigation technique against the adversarial perturbations. However, the results from Korkmaz (2022) can be used to develop more generalizable defense technique shielding only the agent when the state is perturbed in the direction with high sensitivity such that the observation shield is only activated in that direction. Therefore, naive adversarial training by including adversarial examples in the training leads to unstable training and does not significantly improve robustness under strong ϵ-perturbation. To summarize, all above training methods are not very effective for increasing both robustness and performance. Moreover, they only focus on small ϵ adversarial examples and fail to consider the case of high ϵ adversarial examples attack in RL settings.

2.5. Robust Reinforcement Learning

Each element of the MDP framework for RL (observations, actions, transition dynamics, and rewards) can be a target for adversarial perturbation. Robust RL has been studied to improve the robustness of those elements against deliberate attacks and uncertainty. Recently, Zhang et al. Zhang et al. (2021) developed State-adversarial MDP (SA-MDP) to formulate the RL agent under adversarial attacks on state observations. To solve SA-MDP, they proposed a robustness policy regularizer that minimizes the KL-divergence between the optimal policy and a perturbed policy, which they trained on the worst-case perturbation.

Robust MDP (RMDP) Goyal and Grand-Clément (2021) considers the worst case perturbation from transition probabilities. In this framework, the RL agent observes the true state from the environment and acts accordingly, but the environment can choose from a set of transition probabilities that minimizes the reward. The difference between RMDP and SA-MDP is that in the former the adversary intercepts and changes only the observations, while in the latter the ground-truth states are changed, hence RMDP is more suitable for modeling environment parameter changes.

Ying et al. (2022) provides a theoretical analysis of the transition and observation disturbance in RL. It then correlates both types of disturbances using the value function range which measures the gap in the value function between the best and worst states. They then develop a safe method for RL under both disturbances called CVaR Proximal-Policy-Optimization (CPPO). The experiments in Ying et al. (2022) were conducted by introducing Gaussian noise of 0.0-0.5 to the transition and observation of the RL agent. OSRL, on the other hand, focuses on the observation perturbation using the famous adversarial attack FGSM with larger magnitude of perturbation in all environments. Moreover, Ying et al. (2022) did not test their results against the safe RL methods such as SA-DDPG. We think OSRL can be extended in future work to shield the transition function from perturbation which will make it comparable with CPPO. The focus of our paper is on studying the robustness of RL algorithms under adversarial attacks on state observations by utilizing SA-MDP to implement our adversaries. Furthermore, our solution to SA-MDP consists in dealing with the intrinsic weakness of RL policies by learning the environment dynamics during training time, rather than directly regularizing function approximators.

3. Observation-Shielding Reinforcement Learning (OSRL)

This section first describes our problem setup and threat model, then presents our two-step OSRL robustness method that can be used to ensure the safety of any DRL algorithm. The overall architecture of OSRL is shown in Figure 3.

3.1. Threat Model

We consider adversaries that strongly perturb the observation of the state provided to the agent at any given time instant using FGSM Goodfellow et al. (2015). Let $x_t \in \mathbb{R}^N$ and $a_t \in \mathbb{R}^M$ be the observation vector and action vector at time t, respectively. Let $\pi : \mathbb{R}^N \times \mathbb{R}^M$ be the agent's policy. Let $f : \mathbb{R}^N \times \mathbb{R}^M \rightarrow \mathbb{R}^N$ be the dynamics of the environment, which take the state-action pair (x_t, a_t) as inputs and output the next state s_{t+1}. The adversary's goal is to perturb the state observation x_t into $adv(x_t)$ while keeping the underlying true environment state x unchanged to mislead the agent into incurring a sub-optimal return, less than the return of the optimal policy $R(\pi(adv(x)) < R(\pi(x))$. Formally, we formulate this goal as an optimization problem.

Definition 1. Observation Perturbation: The adversary is allowed to perturb the observation x_t that the agent perceived within an ϵ-budget:

$$\|\Delta x_t\|_\infty, L_x \leq x_t + \Delta x_t \leq U_x \tag{2}$$

where $\Delta x_t \in \mathbb{R}^N$ is the adversarial perturbation to the agent's observation and $L_x \in \mathbb{R}^N$, $U_x \in \mathbb{R}^N$ are the

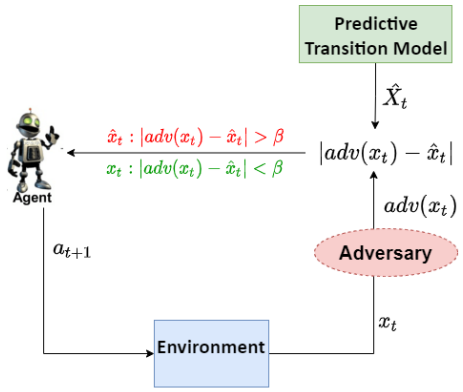

Figure 3: The overall architecture of the proposed method OSRL.

observation lower and upper bounds, respectively.

The adversary perturbs only the agent's state observations, such that the action is taken as $\pi(a|adv(x))$ instead of $\pi(a|x)$. However, the environment still transits from the true state x rather than $adv(x)$ to the next state. Since $adv(x)$ is different from x, the agent's action from $\pi(a|adv(x))$ may be sub-optimal (i.e., $\pi(a|adv(x)) \neq \pi(a|x)$), and thus the adversary is able to reduce the reward, i.e., $R(\pi(adv(x)) < R(\pi(x))$.

3.2. OS-MDP

The observation perturbation in Definition 1 is aligned with many adversarial attacks on state observations Huang et al. (2017) and Lin et al. (2019) but cannot be characterized by existing frameworks such as SA-MDP Zhang et al. (2021) or POMDP Kaelbling et al. (1996). SA-MDP is a framework that characterizes the decision-making problem under adversarial attacks on state observations by extending MDP into (S, A, B, R, p, γ), where $B(s)$ is the perturbation set, to restrict the adversary to perturb a state s only to a predefined set of states $adv(s) \in B(s)$ Zhang et al. (2021). $B(s)$ is an ϵ-bounded set of task-specific "neighboring" states of s, which makes the observation meaningful yet not accurate. Then, the same paper Zhang et al. (2021) extended several DRL algorithms (DQN, DDPG, and PPO) to increase their robustness against the observation perturbations by bounding the performance gap between the state value function of SA-MDP and the state value function of the regular MDP. This bounding is only possible if the difference between the action distribution of π in MDP and π in SA-MDP with and without the perturbation is not too large (Theorem 5 in Zhang et al. (2021)). In other words, the ϵ-perturbation must be bounded to restrict the power of the adversary. Therefore, SA-MDP restricts the adversary to perturb a state s only to a predefined set of states $B(s)$ by ϵ and the RL algorithms must be retrained every time ϵ changes. Our paper utilizes State-Adversarial Deep Deterministic Policy Gradient (SA-DDPG) from Zhang et al. (2021) as a baseline model.

Since our perturbation is not limited within certain set $B(s)$ as in SA-MDP, we propose a novel extension to MDP called Observation-shielding MDP (OS-MDP) to efficiently mitigate large input perturbations without modifying or retraining the underlying RL algorithms.

Definition2. OS-MDP: It extends MDP into $(S, A, \mathbb{I}(X, \hat{X}), R, p, \gamma)$ where the symbol $\mathbb{I}$ indicates the selection of the predicted observation $\hat{X}$ or real observation X, and A, R, γ are defined as in MDP. S in OS-MDP is the hidden state representation of the environment that the agent does not have a direct access to during the attack.

OS-MDP is similar to POMDP Kaelbling et al. (1996) in that the agent cannot directly observe the environment's state and must make decisions under uncertainty about the true environment state. In POMDP, the agent deals with this uncertainty by building its belief in the true state and updating the probability distribution of the current state. However, in OS-MDP the agent cannot trust its belief distribution since the adversary can change the agent's perception about the environment dynamics by manipulating its observations. To act effectively even with perturbed observations, the agent needs to anticipate the consequences of its actions within the environment for an extended period of time into the future. OS-MDP agents can be equipped with this ability by having access to models that can simulate how the environments change in response to their actions.

Given an initial observation x_0 and a pre-trained policy π, our objective is to solve OS-MDP by minimizing the threat impact. We specifically aim to minimize the total distance between each real observation x_t and predicted observation $\hat{x_t}$ while under the threat for T time steps in a row. This can be defined by the optimization problem:

$$\begin{aligned} &\min_{\forall t \in T} d(x_t, \hat{x_t}) \text{ s.t. } a_t = \pi(x_t) = \pi(\hat{x_t}), \\ &p(x_{t+1}|x_t, a_t) = f(\hat{x_t}, a_t) \end{aligned} \tag{3}$$

A common choice of $d(x_t, \hat{x_t})$ is the squared $\mathcal{L}_2$. Here, f is the learned dynamics model of the system (explained next) and T is the attack length which can start from 0 to the maximum length of the episode.

3.3. Predictive Transition Model f

If the environment dynamics f represents the exact dynamics model of the environment, and assuming we have an optimization transition model to solve Eq. 3, then OSRL will always take the optimal actions even under strong ϵ-perturbations. Therefore, learning a good dynamics model (i.e., transition model) can help in developing a strong defense mechanism against the threat model defined in Eq. 2. Depending on the environment, different forms of f can be utilized Weng et al. (2020). For instance, if the environment

Algorithm 1 Predictive Transition Model f

Input:

Pre-trained policy π, environment v, and trainable hyperparameters θ

Output:

Learned transition model $f(x, a, \theta)$

procedure LEARN_DYNAMICS

  $U_{agent} = collect\_trajectories(\pi, v)$

  $U_{random} = collect\_trajectories(\pi_{random}, v)$

  f(x,a,θ) $\leftarrow$ supervised_learning $[U_{agent} \cup U_{random}, \theta]$

return f(x,a,θ)

can be represented by a linear system, then we could design $f(x, a) = \alpha s + \beta a$, where α and β are unknown matrices to be learned from the sample trajectories (x_t, a_t, x_{t+1}) pairs.

For a more complex environment, we could utilize a non-linear model such as neural networks, which usually has better prediction power but may require more training samples. For either case, the model parameters α and β or neural network parameters can be learned via standard supervised learning with the sample trajectories pairs (x_t, a_t, x_{t+1}). The training algorithm of the predictive transition model is given in Algorithm 1. The model is trained using trajectories generated from the agent's policy and from another random policy. Training with a random policy helps the model to explore more the state space and not focus only on the agent's policy state space. Hence, the model is able to generalize to unseen states during training.

3.4. OSRL Algorithm

After learning the transition model f, we now solve the optimization problem in Eq. 3 to detect the adversarial perturbations and use the predicted observations instead. When the attack length is $T > 0$, Eq. 3 usually cannot be directly solved by off-the shelf convex optimization toolbox since the DRL policy π is usually a non-linear and non-convex neural network. Hence, we incorporate Eq 3 into the objective function of RL as a simple condition statement.

To detect the perturbed states, the OSRL agent compares the difference between the predicted observation $\hat{x}$ and the observation x received from the environment, with an environment-specific threshold β (line 8). If $\mid \hat{x} - x \mid > \beta$, then the agent chooses to use $\hat{x}$ as its policy input. Otherwise, the agent uses x as the policy input (Figure 1). The choice of β is arbitrary. In Section 4, we set $\beta = \epsilon/2$ so that any time the observation is perturbed, $\hat{x}$ is used as policy input. Our proposed algorithm is summarized in Algorithm 2.

4. Experiments and Results

We test OSRL against two baselines in 5 environments. The first baseline is vanilla DDPG Lillicrap et al. (2019), an actor-critic method often used in continuous control tasks. The second baseline is SA-DDPG Zhang et al. (2021) which is built using the SA-MDP framework.

Algorithm 2 Observation-Shielding RL

Input:

1: Pre-trained policy π, $f(x, a, \theta)$, ϵ, and episode length T
2: **procedure** OSRL
3: **for all** episodes **do**
4: X = []
5: x_0 = initial state
6: **while** $t \leq T$ **do**
7: $\hat{x}_t = f(x_{t-1}, a_{t-1})$
8: **if** $\mid x_t - \hat{x}_t \mid > \beta$ **then**
9: $a_t = \pi(\hat{x}_t)$
10: **else**
11: $a_t = \pi(x_t)$
12: **end if**
13: **end while**
14: **end for**

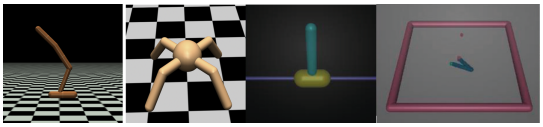

Figure 4: From left to right; Hopper, Ant, InvertedPendulum, and Reacher environments we use to test our approach.

4.1. Experimental Setup

The test environments are InvertedPendulum-v0, Ant, Hopper, Reacher environments from MuJoco as shown in Figure 4:

- **InvertedPendulum.** It is the simplest control task with four continuous observation features: 3-dimension real values, cosine of pendulum angle $\cos\varphi$, sine of pendulum angle $\sin\varphi$, and pendulum angular velocity $\varphi\cdot$. The initial states are uniformly randomized. The action space is continuous and 1-dimensional. The reward is calculated using the following equation:
  $r_t = -(\varphi_t^2 + 0.1 * \varphi_t^{\cdot 2} + 0.001 * ||a_t||_2^2)$
- **Ant.** The state space is 111-dimension, position and velocity of each joint, and contact forces. The initial states are uniformly randomized. The action is an 8-dimensional continuous space. The reward is calculated using the following equation:
  $r_t = x_t^{\cdot} - 0.5 * ||a_t||_2^2 - 0.0005 * ||s_t^{contact}||_2^2 + 1$
- **Hopper.** The state space is 11-dimension, position, and velocity of each joint. The initial states are uniformly randomized. The action is a 3-dimensional

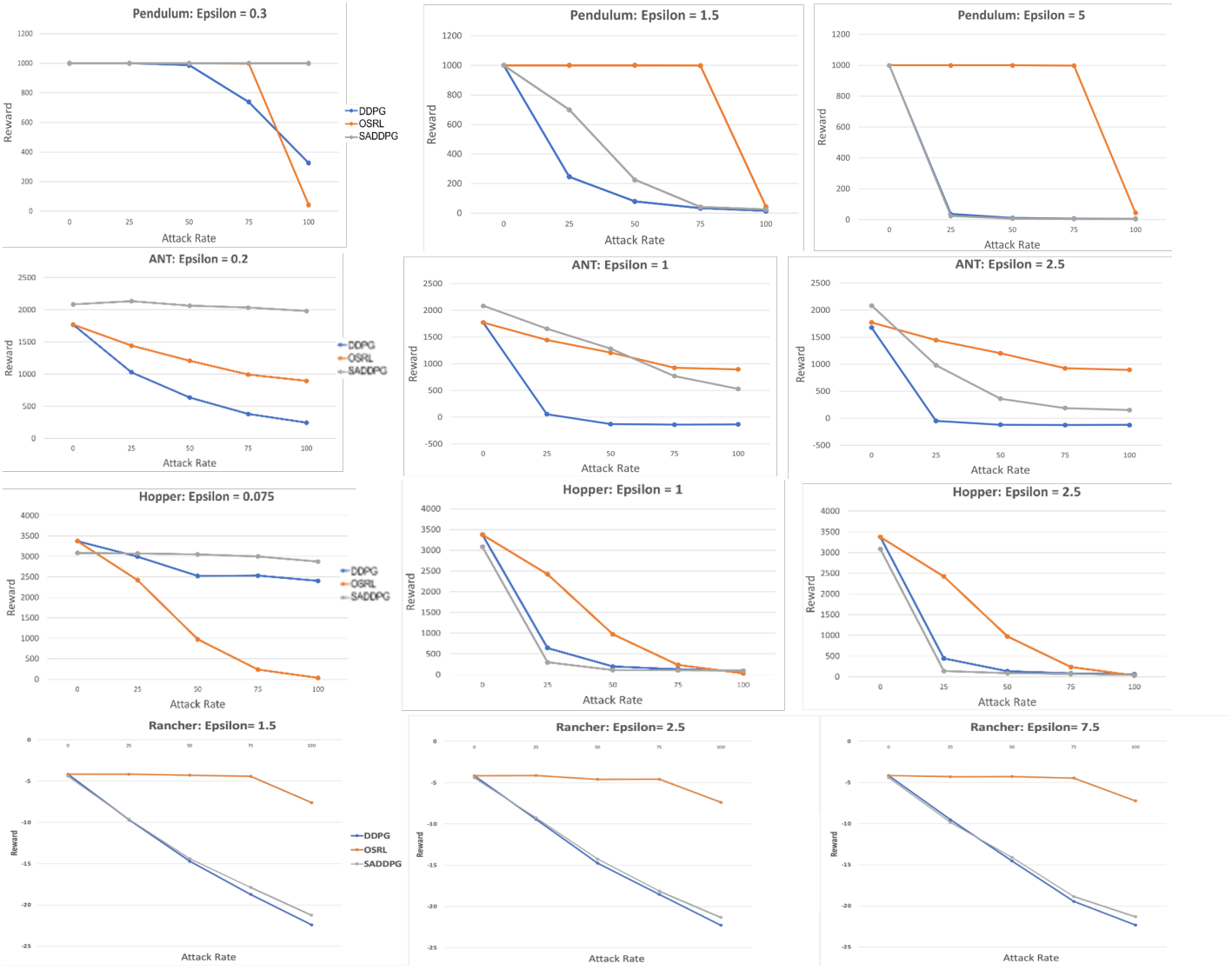

Figure 5: Results obtained by the proposed OSRL in three environments: InvertedPendulum, Ant, Hopper, and Reacher and three levels of ϵ-budget. We tested against two baselines: vanilla DDPG, and SADDPG agents. We also tested at four levels of attack rate: 25%, 50%, 75%, and 100%.

continuous space. This environment is terminated when the agent falls down. The reward is calculated using the following equation:
$r_t = x_{\dot{t}} - 0.01 * ||a_t||_2^2 + 1$

- **Reacher.** The state space is 11-dimension, position and velocity of each joint. The initial states are uniformly randomized. The action is a 2-dimensional continuous space. The reward is calculated by the following equation:
$r_t = [x_t - x_g \leq \epsilon]$

Each observation feature for every environment can range from negative infinity to infinity, so even ϵ-budgets as large as 5 are not very large when compared to the range of values the observations could potentially take.

We test each baseline in each environment with three different levels of ϵ, whose choices are environment-specific. The highest level demonstrates a level of perturbation under which baseline methods fail to act optimally. We also test each baseline against 4 different rates of perturbation: 100%, 75%, 50%, and 25% of the episode. The 0% perturbation indicates an episode without any perturbation, and each baseline achieves optimal reward in that scenario.

4.2. Implementing Predictive Transition

We develop our predictive transition model using a fully-connected neural network for all tested environments.

As a dataset for training, validation, and testing, we use 100 episodes worth of ground-truth transitions sampled from the optimal policy, and another 100 episodes sampled from a random policy. We use the random policy to guarantee that the training data for the transition model covers as much transitions in the environment as possible, avoiding the compound error phenomena. This data can be collected after the weights of the policy have been trained, but before deployment, so that the neural network is not vulnerable to attacks at this stage. Using this dataset, we produce a transition model that takes the previous observation x_{t-1} and action a_{t-1} as input, and predicts the next observation $\hat{x_t}$. The details of this process is given in Algorithm 1.

We found that a two layer, fully connected network works the best for predicting the next states in our environments when compared to an LSTM network. We hypothesize that this is because an LSTM network requires more than one time step of past information to make predictions, which increases the compounding error phenomenon. This phenomenon occurs when multiple predictions are required in a row, which can lead to straying from the ground-truth.

5. Results Analysis

In Figure 5, we see that as the perturbation budget ϵ increases, the performance of OSRL remains constant, whereas the performance of the baselines worsen. Since OSRL does not rely on the observation when an attack is detected, its performance is independent of the strength of the perturbation. For baseline methods, given a large enough perturbation to the observations, the RL agent eventually takes a string of sub-optimal actions and fails the task. We also notice that for each environment, the perturbation level required to defeat the baselines are relatively low when compared to the range of the state space. In each environment, especially at low rates of attack, OSRL is able to successfully complete the task even when faced with attacks that cause baselines to fail quickly. Given even larger perturbations, baselines perform worse, but OSRL's performance is constant and predictable.

We also find that OSRL often underperforms compared to baselines in the case of high rates of small perturbations. We attribute this behavior to the compounding error problem seen often in model-based RL Moerland et al. (2021). We see in each case that as attack rates decrease, the performance of the OSRL agent increases, since less predictions are made based on faulty predictions from past time steps.

RL robustness approaches such as SA-MDP need to modify the RL algorithms to include an ϵ-robustness regularizer while restricting the adversary's power to the same ϵ. As shown in Figure 5, these regularization approaches fail severely when we increase the perturbation ϵ. OSRL, on the other hand, can be directly used on top of any RL algorithm without any changes made to the algorithm or its learned policy, which makes OSRL truly agnostic toward RL algorithms and adversarial models. The only required pre-condition about the adversarial models is to only perturb the state observations instead of perturbing the real state by changing the environment dynamics (Definition 1).

Moreover, the proposed OSRL does not require the RL algorithms to be trained with the same ϵ that will be used for testing as in robust RL algorithms. In other words, our solution is a mitigation approach that would work with larger ϵ perturbation that the RL algorithms never been trained (i.e., regularized) with. Hence, predicting the states during the adversarial attacks helps to mitigate the catastrophic impacts of taking sub-optimal actions using the perturbed observation $\pi(a_t, adv(s))$. It is worth mentioning that the success of our approach is heavily influenced by the long-term prediction accuracy of the predictive transition model.

6. Conclusion

We propose a method for increasing robustness and safety against large perturbations in RL using predictive models and threat detection. Our results show that existing RL robustness methods are weak to large perturbations. While our method is weak when the OSRL agent is attacked for many time steps in a row, it can recover most of the optimal reward when it has the opportunity to compare predictions with the ground-truth state. If this method is used in conjunction with existing RL robustness methods, RL agents can achieve robustness against existing adversarial threat models with no limit on their ϵ-budget. The success of OSRL, however, is based heavily on the long-term accuracy of its prediction model. In the future, we will attempt to improve the long-term prediction accuracy of the model by adapting state-of-the-art techniques used in model-based RL.

References

Fischer, M., Mirman, M., Stalder, S., & Vechev, M. (2019). Online robustness training for deep reinforcement learning.

Goodfellow, I., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. *International Conference on Learning Representations*. http://arxiv.org/abs/1412.6572

Goyal, V., & Grand-Clément, J. (2021). Robust markov decision process: Beyond rectangularity.

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., & Abbeel, P. (2017). Adversarial attacks on neural network policies.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1996). *Planning and acting in partially observable*

Table 1: The tabular results of the experiments.

Pendulum									
	ϵ **=0.3**			ϵ **=1.5**			ϵ **=5**		
Attack Rate	DDPG	SADDPG	OSRL	DDPG	SADDPG	OSRL	DDPG	SADDPG	OSRL
0%	1000	1000	1000	1000	1000	1000	1000	1000	1000
25%	1000	1000	1000	245.98	1000	700.25	36.72	24.14	1000
50%	987.29	1000	1000	79.05	1000	226.10	11.67	6.71	1000
75%	738.09	1000	997.84	33.07	41.8	997.84	6.55	4.51	997.84
100%	325.47	1000	43.64	15.41	24.76	43.64	5.54	4.02	43.64
Ant									
	ϵ **=0.2**			ϵ **=1**			ϵ **=2.5**		
Attack Rate	DDPG	SADDPG	OSRL	DDPG	SADDPG	OSRL	DDPG	SADDPG	OSRL
0%	1771.33	2084.60	1771.33	1771.33	2084.60	1771.33	1771.33	2084.60	1771.33
25%	1029.94	2134.82	1444.07	56.57	1655.71	1444.07	-47.21	981.04	1444.07
50%	637.32	2064.38	1207.89	-127.54	1283.70	1207.89	-120.86	360.72	1201.99
75%	381.20	2036.21	993.83	-140.04	769.51	923.29	-123.64	187.15	923.29
100%	246.98	1980.50	894.57	-134.61	528.46	894.57	-122.21	151.94	894.57
Hopper									
	ϵ **=0.075**			ϵ **=1**			ϵ **=2.5**		
Attack Rate	DDPG	SADDPG	OSRL	DDPG	SADDPG	OSRL	DDPG	SADDPG	OSRL
0%	3375.36	3083.95	3375.16	3375.36	3083.95	3375.16	3375.36	3083.95	3375.16
25%	2995.66	3071.65	2423.49	644.38	300.04	2423.49	446.09	138.30	2423.49
50%	2523.30	3048.95	974.06	197.87	113.72	974.06	132.62	88.18	974.06
75%	2535.07	3002.06	234.50	130.91	105.11	234.50	84.074	68.42	234.50
100%	2403.02	2874.59	31.34	93.63	100.54	31.34	62.41	44.60	31.34
Rancher									
	ϵ **=1.5**			ϵ **=2.5**			ϵ **=7.5**		
Attack Rate	DDPG	SADDPG	OSRL	DDPG	SADDPG	OSRL	DDPG	SADDPG	OSRL
0%	-4.17	-4.41	-4.17	-4.17	-4.41	-4.17	-4.17	-4.41	-4.17
25%	-9.69	-9.62	-4.17	-9.44	-9.25	-4.12	-9.52	-9.86	-4.29
50%	-14.71	-14.40	-4.29	-14.76	-14.25	-4.59	-14.55	-14.11	-4.28
75%	-18.72	-17.88	-4.42	-18.54	-18.14	-4.58	-19.46	-18.87	-4.46
100%	-22.41	-21.26	-7.60	-22.27	-21.32	-7.38	-22.34	-21.32	-7.24

stochastic domains (tech. rep.). USA, Brown University.

Korkmaz, E. (2022). Deep reinforcement learning policies learn shared adversarial features across mdps. *Proceedings of the AAAI Conference on Artificial Intelligence*, *36*(7), 7229–7238. https://doi.org/10.1609/aaai.v36i7.20684

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2019). Continuous control with deep reinforcement learning.

Lin, Y.-C., Hong, Z.-W., Liao, Y.-H., Shih, M.-L., Liu, M.-Y., & Sun, M. (2019). Tactics of adversarial attack on deep reinforcement learning agents.

Lopez-Montiel, M., Rubio, Y., Sánchez-Adame, M., & Orozco-Rosas, U. (2019). Evaluation of algorithms for traffic sign detection. In K. M. Iftekharuddin, A. A. S. Awwal, V. H. Diaz-Ramirez, & A. Márquez (Eds.), *Optics and photonics for information processing xiii* (pp. 135–151). SPIE. https://doi.org/10.1117/12.2529709

Mandlekar, A., Zhu, Y., Garg, A., Fei-Fei, L., & Savarese, S. (2017). Adversarially robust policy learning: Active construction of physically-plausible perturbations, 3932–3939. https://doi.org/10.1109/IROS.2017.8206245

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning.

Moerland, T. M., Broekens, J., & Jonker, C. M. (2021). Model-based reinforcement learning: A survey.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., & Swami, A. (2016). Practical black-box attacks against machine learning.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., & Swami, A. (2015). The limitations of deep learning in adversarial settings.

Pattanaik, A., Tang, Z., Liu, S., Bommannan, G., & Chowdhary, G. (2017). Robust deep reinforcement learning with adversarial attacks.

Rajeswaran, A., Ghotra, S., Ravindran, B., & Levine, S. (2017). Epopt: Learning robust neural network policies using model ensembles.

Russo, A., & Proutiere, A. (2019). Optimal attacks on reinforcement learning policies.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., & Abbeel, P. (2017). Trust region policy optimization.

Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2012). Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks : the official journal of the International Neural Network Society*, *32*, 323–32.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks.

Tramer, F., Carlini, N., Brendel, W., & Madry, A. (2020). On adaptive attacks to adversarial example defenses.

Weng, T.-W., Dvijotham, K., Uesato, J., Xiao, K. Y., Gowal, S., Stanforth, R., & Kohli, P. (2020). Toward evaluating robustness of deep reinforcement learning with continuous control. *ICLR*.

Wiyatno, R., & Xu, A. (2018). Maximal jacobian-based saliency map attack.

Xiao, C., Pan, X., He, W., Peng, J., Sun, M., Yi, J., Liu, M., Li, B., & Song, D. (2019). Characterizing attacks on deep reinforcement learning.

Ying, C., Zhou, X., Su, H., Yan, D., Chen, N., & Zhu, J. (2022). Towards safe reinforcement learning via constraining conditional value-at-risk. https://doi.org/10.48550/ARXIV.2206.04436

Yuan, X., He, P., Zhu, Q., & Li, X. (2018). Adversarial examples: Attacks and defenses for deep learning.

Zhang, H., Chen, H., Xiao, C., Li, B., Liu, M., Boning, D., & Hsieh, C.-J. (2021). Robust deep reinforcement learning against adversarial perturbations on state observations.