Spatiotemporally Constrained Action Space Attacks on Deep Reinforcement Learning Agents

Xian Yeow Lee,[1] Sambit Ghadai,[1] Kai Liang Tan,[1] Chinmay Hegde,[2] Soumik Sarkar[1]*
[1]Department of Mechanical Engineering, Iowa State University, Ames, IA 50011
[2]Tandon School of Engineering, New York University, Brooklyn, NY 11201
{xylee, sambitg, kailiang, soumiks}@iastate.edu, chinmay.h@nyu.edu

Abstract

Robustness of Deep Reinforcement Learning (DRL) algorithms towards adversarial attacks in real world applications such as those deployed in cyber-physical systems (CPS) are of increasing concern. Numerous studies have investigated the mechanisms of attacks on the RL agent's state space. Nonetheless, attacks on the RL agent's action space (corresponding to actuators in engineering systems) are equally perverse, but such attacks are relatively less studied in the ML literature. In this work, we first frame the problem as an optimization problem of minimizing the cumulative reward of an RL agent with decoupled constraints as the budget of attack. We propose the white-box Myopic Action Space (MAS) attack algorithm that distributes the attacks across the action space dimensions. Next, we reformulate the optimization problem above with the same objective function, but with a temporally coupled constraint on the attack budget to take into account the approximated dynamics of the agent. This leads to the white-box Look-ahead Action Space (LAS) attack algorithm that distributes the attacks across the action and temporal dimensions. Our results showed that using the same amount of resources, the LAS attack deteriorates the agent's performance significantly more than the MAS attack. This reveals the possibility that with limited resource, an adversary can utilize the agent's dynamics to malevolently craft attacks that causes the agent to fail. Additionally, we leverage these attack strategies as a possible tool to gain insights on the potential vulnerabilities of DRL agents.

Introduction

The spectrum of Reinforcement Learning (RL) applications ranges from engineering design and control (Lee et al. 2019; Tan et al. 2019) to business (Hu et al. 2018) and creative design (Peng et al. 2018). As RL-based frameworks are increasingly deployed in real-world, it is imperative that the safety and reliability of these frameworks are well understood. While any adversarial infiltration of these systems can be costly, the safety of DRL systems deployed in cyber-physical systems (CPS) such as industrial robotic applications and self-driving vehicles are especially safety and life-critical.

*Corresponding Author

A root cause of these safety concerns is that in certain applications, the inputs to an RL system can be accessed and modified adversarially to cause the RL agent to take sub-optimal (or even harmful) actions. This is especially true when deep neural networks (DNNs) are used as key components (e.g., to represent policies) of RL agents. Recently, a wealth of results in the ML literature demonstrated that DNNs can be fooled to misclassify images by perturbing the input by an imperceptible amount (Goodfellow, Shlens, and Szegedy 2015) or by introducing specific natural looking attributes (Joshi et al. 2019). Such adversarial perturbations have also demonstrated the impacts of attacks on an RL agent's state space as shown by (Huang et al. 2017).

Besides perturbing the RL agent's state space, it is also important to consider adversarial attacks on the agent's *action* space, which in engineering systems, represents physically manipulable actuators. We note that (model-based) actuator attacks have been studied in the cyber-physical security community, including vulnerability of continuous systems to discrete time attacks (Kim et al. 2016); theoretical characteristics of undetectable actuator attacks (Ayas and Djouadi 2016); and "defense" schemes that re-stabilizes a system when under actuation attacks (Huang and Dong 2018). However, the issue of adversarial attacks on a RL agent's action space has relatively been ignored in the DRL literature. In this work, we present a suite of novel attack strategies on a RL agent's action space.

Our contributions:

1. We formulate a white-box Myopic Action Space (MAS) attack strategy as an optimization problem with decoupled constraints.
2. We extend the formulation above by coupling constraints to compute a non-myopic attack that is derived from the agent's state-action dynamics and develop a white-box Look-ahead Action Space (LAS) attack strategy. Empirically, we show that LAS crafts a stronger attack than MAS using the same budget.
3. We illustrate how these attack strategies can be used to understand a RL agent's vulnerabilities.
4. We present analysis to show that our proposed attack algorithms leveraging projected gradient descent on the

surrogate reward function (represented by the trained RL agent model) converges to the same effect of applying projected gradient descent on the true reward function.

Related Works

Due to the large amount of recent progress in the area of adversarial machine learning, we only focus on reviewing the most recent attack and defense mechanisms proposed for DRL models. Table 1 presents the primary landscape of this area of research to contextualize our work.

Adversarial Attacks on RL Agent

Several studies of adversarial attacks on DRL systems have been conducted recently. (Huang et al. 2017) extended the idea of FGSM attacks in deep learning to RL agent's policies to degrade the performance of a trained RL agent. Furthermore, (Behzadan and Munir 2017) showed that these attacks on the agent's state space are transferable to other agents. Additionally, (Tretschk, Oh, and Fritz 2018) proposed attaching an Adversarial Transformer Network (ATN) to the RL agent to learn perturbations that will deceive the RL agent to pursue an adversarial reward. While the attack strategies mentioned above are effective, they do not consider the dynamics of the agent. One exception is the work by (Lin et al. 2017) that proposed two attack strategies. One strategy was to attack the agent when the difference in probability/value of the best and worst action crosses a certain threshold. The other strategy was to combine a video prediction model that predicts future states and a sampling-based action planning scheme to craft adversarial inputs to lead the agent to an adversarial goal, which might not be scalable. Other studies of adversarial attacks on the specific application of DRL for path-finding have also been conducted by (Xiang et al. 2018) and (Bai et al. 2018), which results in the RL agent failing to find a path to the goal or planning a path that is more costly.

Robustification of RL Agents

As successful attack strategies are being developed for RL models, various works on training RL agents to be robust against attacks have also been conducted. (Pattanaik et al. 2018) proposed that a more severe attack can be engineered by increasing the probability of the worst action rather than decreasing the probability of the best action. They showed that the robustness of an RL agent can be improved by training the agent using these adversarial examples. More recently, (Tessler, Efroni, and Mannor 2019) presented a method to robustify RL agent's policy towards action space perturbations by formulating the problem as a zero-sum Markov game. In their formulation, a separate nominal and adversary policy are trained simultaneously with a critic network being updated over the mixture of both policies to improve both adversarial and nominal policies. Meanwhile, (Havens, Jiang, and Sarkar 2018) proposed a method to detect and mitigate attacks by employing a hierarchical learning framework with multiple sub-policies. They showed that the framework reduces agent's bias to maintain high nominal rewards in the absence of adversaries. We note that other methods to defend against adversarial attacks exist, such as studies done by (Tramèr et al. 2017) and (Sinha, Namkoong, and Duchi 2018). These works are done mainly in the context of a DNN but may be extendable to DRL agents that employs DNN as policies, however discussing these works in detail goes beyond the scope of this work.

Mathematical Formulation

Preliminaries

We focus exclusively on model-free RL approaches. Below, let s_t and a_t denote the (continuous, possibly high-dimensional) vector variables denoting *state* and *action*, respectively, at time t. We assume a state evolution function, $s_{t+1} = E(s_t, a_t)$ and let $R(s_t, a_t)$ denote the reward signal the agent receives for taking the action a_t, given s_t. The goal of the RL agent is to choose actions that maximizes the cumulative reward, $\sum_t R(s_t, a_t)$, given access to the trajectory, τ, comprising all past states and actions. In value-based methods, the RL agent determines action at each time step by finding an intermediate quantity called the *value function* that satisfies the recursive Bellman Equations. One example of such method is Q-learning (Watkins and Dayan 1992) where the agent discovers the Q-function, defined recursively as:

$$Q_t(s_t, a_t) = R(s_t, a_t) + \max_{a'} Q_{t+1}(E(s_t, a_t), a').$$

The optimal action (or "policy") at each time step is to deterministically select the action that maximizes this Q-function conditioned on the observed state, i.e.,

$$a_t^* = \arg\max_a Q(s_t, a).$$

In DRL, the Q-function in the above formulation is approximated via a parametric neural network Θ; methods to train these networks include Deep Q-networks (Mnih et al. 2015).

In policy-based methods such as policy gradients (Sutton et al. 2000), the RL agent *directly* maps trajectories to policies. In contrast with Q-learning, the selected action is sampled from the policy parameterized by a probability distribution, $\pi = \mathbb{P}(a|s, \Theta)$, such that the expected rewards (with expectations taken over π) are maximized:

$$\pi^* = \arg\max_\pi E[R(\tau)], \;\; a_t^* \sim \pi^*.$$

In DRL, the optimal policy π is the output of a parametric neural network Θ, and actions at each time step are sampled; methods to train this neural network include proximal policy optimization (PPO) (Schulman et al. 2017).

Threat Model

Our goal is to identify adversarial vulnerabilities in RL agents in a principled manner. To this end, We define a formal threat model, where we assume the adversary possesses the following capabilities:

1. **Access to RL agent's action stream**. The attacker can directly perturb the agent's nominal action adversarially (under reasonable bounds, elaborated below). The nominal agent is also assumed to be a closed-loop system and have no active defense mechanisms.

Table 1: Landscape of adversarial attack strategies on RL agents. First column denotes if the attack takes into account the dynamics of the agent. Second column shows the method of computing the attacks; O denotes an optimization-based method and M denotes a model-based method where the parameters of a model needs to be learned. Last column represents if the attacks are mounted on agent's state space (S) or action space (A).

Method	Includes Dynamics	Method	Space of Attack
FGSM on Policies (Huang et al. 2017)	X	O	S
ATN (Tretschk, Oh, and Fritz 2018)	X	M	S
Gradient based Adversarial Attack (Pattanaik et al. 2018)	X	O	S
Policy Induction Attacks (Behzadan and Munir 2017)	X	O	S
Strategically-Timed and Enchanting Attack (Lin et al. 2017)	✓	O, M	S
NR-MDP (Tessler, Efroni, and Mannor 2019)	X	M	A
Myopic Action Space (MAS)	X	O	A
Look-ahead Action Space (LAS)	✓	O	A

2. **Access to RL agent's training environment**. This is required to perform forward simulations to design an optimal sequence of perturbations (elaborated below).
3. **Knowledge of trained RL agent's DNN**. This is needed to understand how the RL agent acts under nominal conditions, and to compute gradients. In the adversarial ML literature, this assumption is commonly made under the umbrella of white-box attacks.

In the context of the above assumptions, the goal of the attacker is to choose a (bounded) action space perturbation that minimizes long-term discounted rewards. Based on how the attacker chooses to perturb actions, we define and construct two types of optimization-based attacks. We note that alternative approaches, such as training another RL agent to learn a sequence of attacks, is also plausible. However, an optimization-based approach is computationally more tractable to generate on-the-fly attacks for a target agent compared to training another RL agent (especially for high-dimensional continuous action spaces considered here) to generate attacks. Therefore, we restrict our focus on optimization-based approaches in this paper.

Myopic Action-Space (MAS) Attack Model

We first consider the case where the attacker is *myopic*, i.e., at each time step, they design perturbations in a greedy manner without regards to future considerations. Formally, let δ_t be the action space perturbation (to be determined) and b be a *budget* constraint on the magnitude of each δ_t [1]. At each time step t, the attacker designs δ_t such that the anticipated future reward is minimized

$$\begin{aligned}\min_{\delta_t} R_{\text{adv}}(\delta_t) &= R(s_t, a_t+\delta_t) + \sum_{j=t+1}^{T} R(s_j, a_j)\\ \text{subject to : } &\|\delta_t\|_p \le b,\\ &s_{j+1} = E(s_j, a_j),\\ &a_j = \Theta(s_j) \text{ (for } j = t,\dots,T),\end{aligned} \tag{1}$$

where $\|\cdot\|_p$ denotes the ℓ_p-norm for some $p \ge 1$. Observe that while the attacker ostensibly solves separate (decoupled) problems at each time, the states themselves are not independent since given any trajectory, $s_{j+1} = E(s_j, a_j)$, where $E(s_j, a_j)$ is the transition of the environment based on s_j and a_j. Therefore, $\mathbf{R}$ is implicitly coupled through time since it depends heavily on the evolution of state trajectories rather than individual state visitations. Hence, the adversary perturbations solved above are strictly myopic and we consider this a static attack on the agent's action space.

[1]Physically, the budget may reflect a real physical constraint, such as the energy requirements to influence an actuation, or it may be a reflection on the degree of imperceptibility of the attack.

Look-ahead Action Space (LAS) Attack Model

Next, we consider the case where the attacker is able to look ahead and chooses a designed *sequence* of future perturbations. Using the same notation as above, let $\sum_{j=t}^{t+H} R(s_j, a_j+\delta_j)$ denote the sum of rewards until a horizon parameter H, and let $\sum_{j=t+H+1}^{T} R(s_j, a_j)$ be the future sum of rewards from time $j = t + H + 1$. Additionally, we consider the (concatenated) matrix $\Delta = [\delta_t, \delta_{t+1} \dots \delta_{t+H}]$ and B denote a budget parameter. The attacker solves the optimization problem:

$$\begin{aligned}\min_{\Delta} R_{adv}(\Delta) &= \sum_{j=t}^{t+H} R(s_j, a_j+\delta_j) + \sum_{j=t+H+1}^{T} R(s_j, a_j)\\ \text{subject to : } &\|\Delta\|_{p,q} \le B, \Delta = [\delta_t, \delta_{t+1}, \dots, \delta_H],\\ &s_{j+1} = E(s_j, a_j),\\ &a_j = \Theta(s_j)\end{aligned} \tag{2}$$

where $\|\cdot\|_{p,q}$ denotes the $\ell_{p,q}$-norm (Boyd and Vandenberghe 2004). By coupling the objective function and constraints through the temporal dimension, the solution to the optimization problem above is then forced to take the dynamics of the agent into account in an explicit manner.

Proposed Algorithms

In this section, we present two attack algorithms based on the optimization formulations presented in previous section.

Algorithm for Mounting MAS Attacks

We observe that (1) is a nonlinear constrained optimization problem; therefore, an immediate approach to solve it is via projected gradient descent (PGD). Specifically, let $\mathcal{S}$ denote the ℓ_p ball of radius b in the action space. We compute the gradient of the adversarial reward, ∇R_{adv} w.r.t. the action space variables and obtain the *unconstrained* adversarial action $\hat{a}_{t+\frac{1}{2}}$ using gradient descent with step size η. Next, we calculate the *unconstrained* perturbation δ_t and project in onto $\mathcal{S}$ to get δ'_t :

$$\begin{aligned} \hat{a}_{t+\frac{1}{2}} &= a_t - \eta \nabla R_{adv}(s_t, \hat{a}_t), \\ \delta_t &= \hat{a}_{t+\frac{1}{2}} - a_t, \\ \delta'_t &= P_{\mathcal{S}}(\delta_t). \end{aligned} \tag{3}$$

Here, a_t represents the nominal action. We note that this approach resembles the fast gradient-sign method (FGSM) (Goodfellow, Shlens, and Szegedy 2015), although we compute standard gradients here. As a variation, we can compute multiple steps of gradient descent w.r.t the action variable prior to projection; this is analogous to the basic iterative method (or iterative FGSM) (Kurakin, Goodfellow, and Bengio 2016). The MAS attack algorithm is shown in the supplementary material.

We note that in DRL approaches, only a *noisy proxy* of the true reward function is available: In value-based methods, we utilize the learned Q-function (for example, a DQN) as an approximate of the true reward function, while in policy-iteration methods, we use the probability density function returned by the optimal policy as a proxy of the reward, under the assumption that actions with high probability induces a high expected reward. Since DQN selects the action based on the argmax of Q-values and policy iteration samples the action with highest probability, the Q-values/action-probability remains a useful proxy for the reward in our attack formulation. Therefore, our proposed MAS attack is technically a version of *noisy projected gradient descent* on the policy evaluation of the nominal agent. We elaborate on this further in the theoretical analysis section.

Algorithm for Mounting LAS Attacks

The previous algorithm is myopic and can be interpreted as a purely *spatial* attack. In this section, we propose a *spatiotemporal* attack algorithm by solving Eq. (2) over a given time window H. Due to the coupling of constraints in time, this approach is more involved. We initialize a copy of both the nominal agent and environment, called the adversary and adversarial environment respectively. At time t, we sample a virtual roll-out trajectory up until a certain horizon $t + H$ using the pair of adversarial agent and environment. At each time step k of the virtual roll-out, we compute action space perturbations $\delta_{t,k}$ by taking (possibly multiple) gradient updates. Next, we compute the norms of each $\delta_{t,k}$ and project the sequence of norms back onto an ℓ_q-ball of radius B. The resulting projected norms at each time point now represents the individual budgets, b_k, of the spatial dimension at each time step. Finally, we project the original $\delta_{t,k}$ obtained in the previous step onto the ℓ_p-balls of radii b_k, respectively to get the final perturbations $\delta'_{t,k}$.[2] We note that to perform virtual roll-outs at every time step t, the state of the E_{adv} has to be the same as the state of E_{nom} at t. To accomplish this, we saved the history of all previous actions to re-compute the state of the E_{adv} at time t from $t = 0$. While this current implementation may be time-consuming, we believe that this problem can be avoided by giving the adversary direct access to the current state of the nominal agent through platform API-level modifications; or explicit observations (in real-life problems).

[2]Intuitively, these steps represent the allocation of overall budget B across different time steps.

Algorithm 1: Look-ahead Action Space (LAS) Attack

```
1  Initialize nominal and adversary environments $E_{nom}$,
     $E_{adv}$ with same random seed
2  Initialize nominal agent $\pi_{nom}$ weights, $\theta$
3  Initialize budget $B$, adversary action buffer $A_{adv}$,
     horizon $H$
4  while $t \leq T$ do
5      Reset $A_{adv}$
6      if $H$ = 0 then
7          Reset $H$ and $B$
8      while $k \leq H$ do
9          Compute gradient of surrogate reward $\nabla R_{adv}$
10         Compute adversarial action $\hat{a}_{t+\frac{1}{2},k}$ using
             $\nabla R_{adv}$
11         Compute $\delta_{t,k} = \hat{a}_{t+\frac{1}{2},k} - a_{t,k}$
12         Append $\delta_{t,k}$ to $A_{adv}$
13         Step through $E_{adv}$ with $a_{t,k}$ to get next state
14     Compute $||\delta_{t,k}||_{\ell_p}$ for each element in $A_{adv}$
15     Project sequence of $||\delta_{t,k}||_{\ell_p}$ in $A_{adv}$ on to ball of
         size $B$ to obtain look-ahead sequence of budgets
         $[b_{t,k}, b_{t,k+1} \dots b_{t,k+H}]$
16     Project each $\delta_{t,k}$ in $A_{adv}$ on to look-ahead sequence
         of budgets computed in the previous step to get
         sequence $[\delta'_{t,k}\ \delta'_{t,k+1} \dots \delta'_{t,k+H}]$
17     Compute projected adversarial action $\hat{a}_t = a_t + \delta'_{t,k}$
18     Step through $E_{nom}$ with $\hat{a}_t$
19     $B \leftarrow max(0, B - \delta'_{t,k})$
20     $H \leftarrow H - 1$
```

In subsequent time steps, the procedure above is repeated with a reduced budget of $B - \sum_{t=0}^{t} \delta'_t$ and reduced horizon $H - t$ until H reaches zero. The horizon H is then reset again for planning a new spatiotemporal attack. An alternative formulation could also be shifting the window without reducing its length until the adversary decides to stop the attack. However, we consider the first formulation such that we can compare the performance of LAS with MAS for an equal overall budget. This technique of re-planning the δ'_t at every step while shifting the window of H is similar to the concept of *receding horizons* regularly used in optimal control (Mayne and Michalska 1990). It is evident that using this form of dynamic re-planning mitigates the planning error that occurs when the actual and simulated state trajectories diverge due to error accumulation (Qin and Badgwell 2003). Hence, we perform this re-planning at every t to account for this deviation. The pseudocode is provided in Alg. 1.

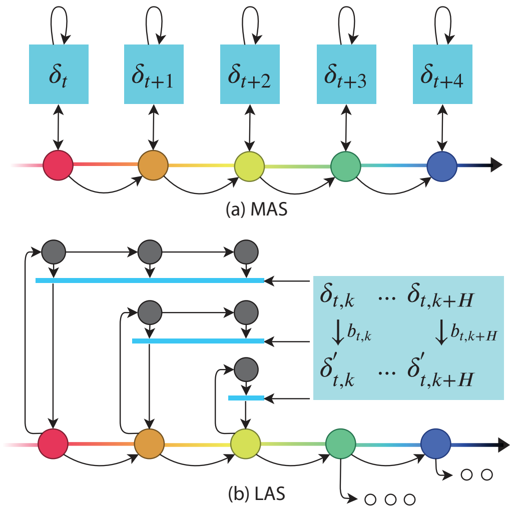

Figure 1: Visual comparison of MAS and LAS. In MAS, each δ_t is computed via multi-step gradient descent w.r.t. expected rewards for the current step. In LAS, each $\delta'_{t,k}$ is computed w.r.t. the dynamics of the agent with receding horizon. An adversarial agent & environment is used to compute LAS for each step. Projection is applied to each δ_t in the temporal domain. The final perturbed action is obtained by adding the first $\delta'_{t,k}$ to the nominal action. This is done until the end of the attack window, i.e., $H - t = 0$.

Theoretical Analysis

We can show that projected gradient descent on the surrogate reward function (modeled by the RL agent network) to generate both MAS and LAS attacks provably converges; this can be accomplished since gradient descent on a surrogate function is akin to a noisy gradient descent on the true adversarial reward.

As described in previous sections, our MAS/LAS algorithms are motivated in terms of the adversarial reward R_{adv}. However, if we use either DQN or policy gradient networks, we do not have direct access to the reward function, but only its noisy *proxy*, defined via a neural network. Therefore, we need to argue that performing (projected) gradient descent using this proxy loss function is a sound procedure. To do this, we appeal to a recent result by (Ge et al. 2015), who prove convergence of noisy gradient descent approximately converges to a local minimum. More precisely, consider a general constrained nonlinear optimization problem:

$$\min f(x)$$
$$\text{s.t. } c(x) = 0,$$

where c is an arbitrary (differentiable, possibly vector-valued) function encoding the constraints. Define $S = \{x | c(x) = 0\}$ define the constraint set. We attempt to minimize the objective function via noisy (projected) gradient updates:

$$x_{t+1/2} = x_t - \eta \nabla f(x_t) + \xi_t\ ,$$
$$x_{t+1} = P_S(x_{t+1/2}).$$

Theorem 1. *(Convergence of noisy projected gradients.) Assume that the noise terms $\{\xi_t\}$ are i.i.d., satisfying $E[\xi] = 0, E[\xi\xi^T] = \sigma^2 Id$, $\|\xi\| \leq O(1)$ almost surely. Assume that both the constraint function $c()$ and the objective function $f(\cdot)$ is β-smooth, L-Lipschitz, and possesses ρ_i-Lipschitz Hessian. Assume further that the objective function f is B-bounded. Then, there exists a learning rate $\eta = O(1)$ such that with high probability, in polylog$(1/\eta^2)$ iterations, noisy projected gradient descent converges to a point $\hat{x}$ that is polylog$(\sqrt{\eta})$-close to some local minimum of f.*

In our case, f and ξ depends on the structure of the RL agent's neural network. (Smoothness assumptions of f can perhaps be justified by limiting the architecture of the network, but the iid-ness assumption on ξ is hard to verify). As such, it is difficult to ascertain whether the assumptions of the above theorem are satisfied in specific cases. Nonetheless, an interesting future theoretical direction is to understand Lipschitz-ness properties of specific families of DQN/policy gradient agents.

We defer further analysis of the double projection step onto mixed-norm balls used in our proposed LAS algorithms to the supplementary material.

Experimental Results & Discussion

To demonstrate the effectiveness and versatility of our methods, we implemented them on RL agents with continuous action environments from OpenAI's gym (Brockman et al. 2016) as they reflect the type of action space in most practical applications [3]. For policy-based methods, we trained a nominal agent using the PPO algorithm and a DoubleDQN (DDQN) agent (Van Hasselt, Guez, and Silver 2016) for value-based methods[4]. Additionally, we utilize Normalized Advantage Functions (Gu et al. 2016) to convert the discrete nature of DDQN's output to continuous action space. For succinctness, we present the results of the attack strategies only on PPO agent for the Lunar-Lander environment. Additional results of DDQN agent in Lunar Lander and Bipedal-Walker environments and PPO agent in Bipedal-Walker, Mujoco Hopper, Half-Cheetah and Walker environments are provided in the supplementary materials. As a baseline, we implemented a random action space attack, where a random perturbation bounded by the same budget b is applied to the agent's action space at every step. For MAS attacks, we implemented two different spatial projection schemes, ℓ_1 projection based on (Condat 2016) that represents a sparser distribution and ℓ_2 projection that represents a denser distribution of attacks. For LAS attacks, all combinations of spatial and temporal projection for ℓ_1 and ℓ_2 were implemented.

[3]Codes and links to supplementary are available at https://github.com/xylee95/Spatiotemporal-Attack-On-Deep-RL-Agents

[4]The only difference in implementation of policy vs value-based methods is that in policy methods, we take analytical gradients of a distribution to compute the attacks (e.g., in line 10 of Algorithm 1) while for value-based methods, we randomly sample adversarial actions to compute numerical gradients.

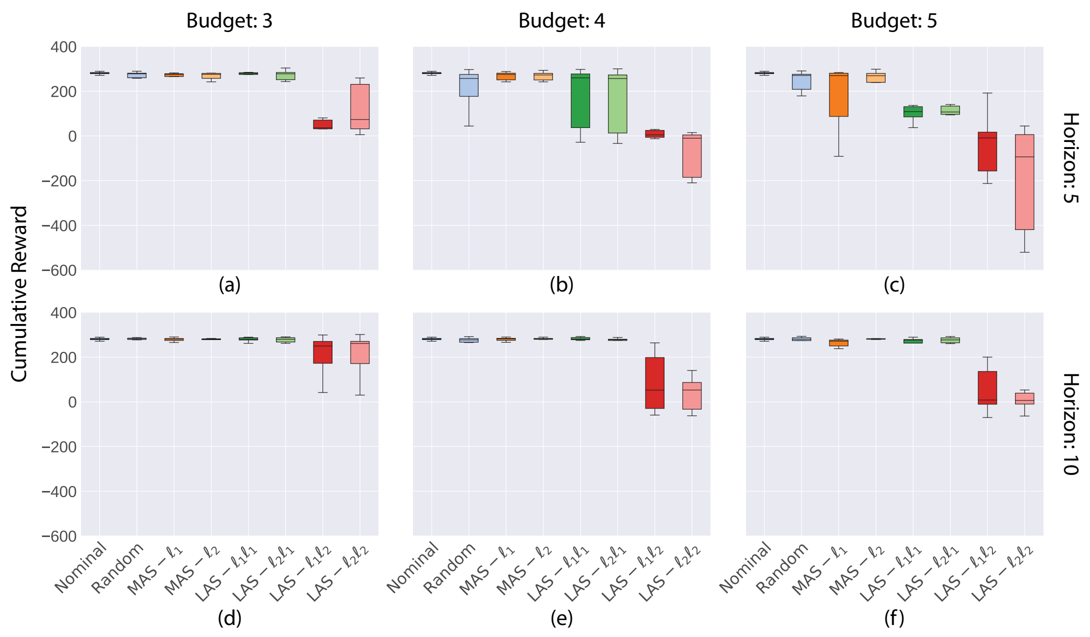

Figure 2: Box plots of PPO Lunar Lander showing average cumulative reward across 10 episodes for each attack methods. The top Figs. (a-c) have H=5 with B= 3, 4, and 5 respectively. For a direct comparison, corresponding MAS budgets are taken as $b = B/H$. Similarly, Figs.(d-f) have the same B values but with H=10. An obvious trend is that as B increases, the effectiveness of LAS over MAS becomes more evident as seen in the decreasing trend of the reward.

Comparison of MAS and LAS Attacks

Fig. 2 shows distributions of cumulative rewards obtained by the PPO agent across ten episodes in a Lunar Lander environment, with each subplot representing different combinations of budget, B and horizon, H. Top three subplots show experiments with a H value of 5 time steps and b value of 3, 4, and 5 from left to right respectively. Bottom row of figures show a similar set of experiments but with a H value of 10. For a direct comparison between MAS and LAS attacks with equivalent budgets across time, we have assigned the corresponding MAS budget values as $b = B/H$. This assumes that the total budget B is allocated uniformly across every time step for a given H, while LAS has the flexibility to allocate the attack budget non-uniformly in the same interval, conditioned on the dynamics of the agent.

We note that keeping H constant while increasing B provides both MAS and LAS with a higher budget to inject δ_t to the nominal actions. We observe that with a low budget of 3 (Fig. 2a), only LAS is successful in attacking the RL agent, as seen by the corresponding decrease in rewards. With a higher budget of 5 (Fig. 2c), MAS has a more apparent effect on the performance of the RL agent while LAS reduces the performance of the agent severely.

With B constant, increasing H allows the allocated B to be distributed along the increased time horizon. In other words, LAS virtually looks-ahead further into the future. In the most naive case, a longer horizon dilutes the severity of each δ_t in compared to shorter horizons. By comparing similar budget values of different horizons (i.e horizons 5 and 10 for budget 3, Fig. 2a and Fig. 2d respectively), attacks for $H = 10$ are generally less severe than their $H = 5$ counterparts. For all B and H combinations, we observe that MAS attacks are generally less effective compared to LAS. We note that this is a critical result of the study as most literature on static attacks have shown that the attacks can be ineffective below a certain budget. Here, we demonstrate that while MAS attacks can seemingly look ineffective for a given budget, a stronger and more effective attack can essentially be crafted using LAS with the same budget.

In the following sections, we further study the difference between MAS and LAS as well as demonstrate how the attacks can be utilized to understand the vulnerabilities of the agent in different environments.

Action Dimension Decomposition of LAS Attacks

Fig. 3 shows action dimension decomposition of LAS attacks. The example shown in Fig. 3 is the result of ℓ_2 projection in action space with ℓ_2 projection in time. From Fig. 3a, we observe that through all the episodes of LAS attacks, one of the action dimension (i.e., Up - Down direction of lunar lander) is consistently perturbed more, i.e., accumulates more attack, than Left-Right direction.

Fig. 3b shows a detailed view of action dimension attacks for an episode (Episode 1). It is evident from the figure that the Up-Down actions of the lunar lander are more prone to attacks throughout the episode than Left-Right actions. Additionally, Left-Right action attacks are restricted after certain time steps and only the Up-Down actions are attacked further. Fig. 3c further corroborates the observation in the Lunar Lander environment. As the episode progresses

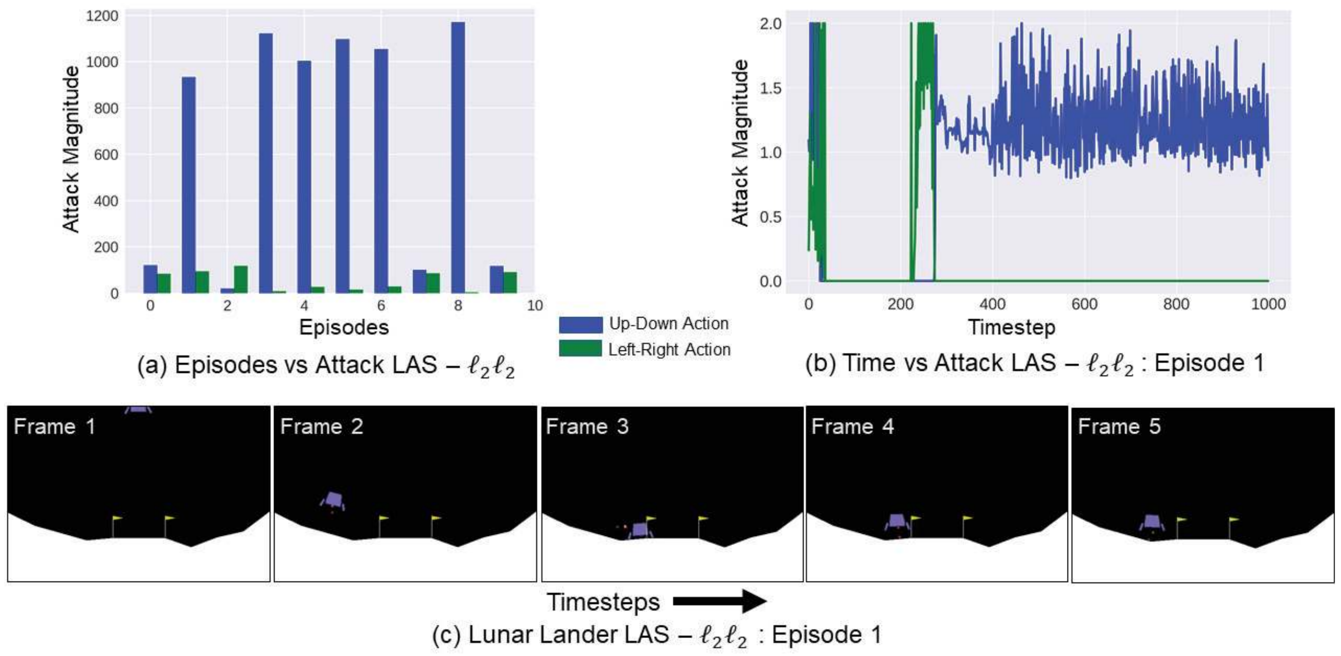

Figure 3: Time vs Attack magnitude along action dimension for LAS attacks with $B = 4, H = 5$ in Lunar Lander environment with PPO RL agent. (a) Variation of attack magnitude along Up-Down and Left-Right action dimensions through different episodes. In all episodes except episode 2, Up-Down action is more heavily attacked than Left-Right. (b) Variation of attack magnitude through time for episode 1 of (a). After 270 steps, the agent is not attacked in the Left-Right dimension, but heavily attacked in Up-Down directions. (c) Actual rendering of Lunar Lander environment for episode 1 of (a) corresponding to (b). Frame 1-5 are strictly increasing time steps showing trajectory of the RL agent controlling the lunar lander.

in Fig. 3c, the lunar lander initially lands on the ground in frame 3, but lifts up and hovers until the episode ends in frame 5. This observation supports the fact that the proposed attacks are effective in perturbing the action dimensions in an optimal manner; as in this case, perturbing the lunar lander in the horizontal direction will not further decrease rewards. On the other hand, hovering the lunar lander will cause the agent to use more fuel, which consequently decreases the total reward. From these studies, it can be concluded that LAS attacks (correlated with projections of actions in time) can clearly isolate vulnerable action dimension(s) of the RL agent to mount a successful attack.

Ablation Study of Horizon and Budget

Lastly, we performed multiple ablation studies to compare the effectiveness of LAS and MAS attacks. While we have observed that LAS attacks are generally stronger than MAS, we hypothesize that there will be an upper limit to LAS's advantage as the allowable budget increases. We take the difference of each attack's reduction in rewards (i.e. attack - nominal) and visualize how much rewards LAS reduces as compared to MAS under different conditions of B and H. In the case of PPO in Lunar Lander, we observe that the reduction in rewards of LAS vs MAS becomes less drastic as budget increases, hence showing that LAS has diminishing returns as both MAS and LAS saturates at higher budgets. We defer detailed discussions and additional figures of the ablation study to the supplementary materials.

Conclusion & Future Work

In this study, we present two novel attack strategies on an RL agent's action space; a myopic attack (MAS) and a non-myopic attack (LAS). The results show that LAS attacks, that were crafted with explicit use of the agent's dynamics information, are more powerful than MAS attacks. Additionally, we observed that applying LAS attacks on RL agents reveals the possible vulnerable actuators of an agent, as seen by the non-uniform distribution of attacks on certain action dimensions. This can be leveraged as a tool to identify the vulnerabilities and plan a mitigation strategy under similar attacks. Possible future works include extending the concept of LAS attacks to state space attacks where the agent's observations are perturbed instead of the agent's actions while taking into account the dynamics of the agent. Additionally, while we did not focus on the imperceptibility and deployment aspects of the proposed attacks in this study, defining a proper metric in terms of detectability in action space and optimizing the budget to remain undetected for different environments will be a future research direction.

Acknowledgement

This work was supported in part by NSF grants CNS-1845969 and CCF-2005804, and AFOSR YIP Grant FA9550-17-1-0220.

References

Ayas, M. S., and Djouadi, S. M. 2016. Undetectable sensor and actuator attacks for observer based controlled cyber-physical systems. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1–7.

Bai, X.; Niu, W.; Liu, J.; Gao, X.; Xiang, Y.; and Liu, J. 2018. Adversarial examples construction towards white-box q table variation in dqn pathfinding training. In *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, 781–787. IEEE.

Behzadan, V., and Munir, A. 2017. Vulnerability of deep reinforcement learning to policy induction attacks. In Perner, P., ed., *Machine Learning and Data Mining in Pattern Recognition*, 262–275. Cham: Springer International Publishing.

Boyd, S., and Vandenberghe, L. 2004. *Convex optimization*. Cambridge university press.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Condat, L. 2016. Fast projection onto the simplex and the l1 ball. *Mathematical Programming* 158(1-2):575–585.

Ge, R.; Huang, F.; Jin, C.; and Yuan, Y. 2015. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, 797–842.

Goodfellow, I.; Shlens, J.; and Szegedy, C. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Gu, S.; Lillicrap, T.; Sutskever, I.; and Levine, S. 2016. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, 2829–2838.

Havens, A.; Jiang, Z.; and Sarkar, S. 2018. Online robust policy learning in the presence of unknown adversaries. In *Advances in Neural Information Processing Systems*, 9916–9926.

Hu, Z.; Liang, Y.; Zhang, J.; Li, Z.; and Liu, Y. 2018. Inference aided reinforcement learning for incentive mechanism design in crowdsourcing. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc. 5507–5517.

Huang, X., and Dong, J. 2018. Reliable control policy of cyber-physical systems against a class of frequency-constrained sensor and actuator attacks. *IEEE Transactions on Cybernetics* 48(12):3432–3439.

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

Joshi, A.; Mukherjee, A.; Sarkar, S.; and Hegde, C. 2019. Semantic adversarial attacks: Parametric transformations that fool deep classifiers. In *The IEEE International Conference on Computer Vision (ICCV)*.

Kim, J.; Park, G.; Shim, H.; and Eun, Y. 2016. Zero-stealthy attack for sampled-data control systems: The case of faster actuation than sensing. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, 5956–5961.

Kurakin, A.; Goodfellow, I.; and Bengio, S. 2016. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*.

Lee, X. Y.; Balu, A.; Stoecklein, D.; Ganapathysubramanian, B.; and Sarkar, S. 2019. A case study of deep reinforcement learning for engineering design: Application to microfluidic devices for flow sculpting. *Journal of Mechanical Design* 141(11).

Lin, Y.-C.; Hong, Z.-W.; Liao, Y.-H.; Shih, M.-L.; Liu, M.-Y.; and Sun, M. 2017. Tactics of adversarial attack on deep reinforcement learning agents. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 3756–3762. AAAI Press.

Mayne, D. Q., and Michalska, H. 1990. Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control* 35(7):814–824.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Pattanaik, A.; Tang, Z.; Liu, S.; Bommannan, G.; and Chowdhary, G. 2018. Robust deep reinforcement learning with adversarial attacks. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2040–2042. International Foundation for Autonomous Agents and Multiagent Systems.

Peng, X. B.; Abbeel, P.; Levine, S.; and van de Panne, M. 2018. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)* 37(4):143.

Qin, S. J., and Badgwell, T. A. 2003. A survey of industrial model predictive control technology. *Control engineering practice* 11(7):733–764.

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

Sinha, A.; Namkoong, H.; and Duchi, J. 2018. Certifiable distributional robustness with principled adversarial training. In *International Conference on Learning Representations*.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063.

Tan, K. L.; Poddar, S.; Sharma, A.; and Sarkar, S. 2019. Deep reinforcement learning for adaptive traffic signal control. *arXiv preprint arXiv:1911.06294*.

Tessler, C.; Efroni, Y.; and Mannor, S. 2019. Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*, 6215–6224.

Tramèr, F.; Kurakin, A.; Papernot, N.; Goodfellow, I.; Boneh, D.; and McDaniel, P. 2017. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*.

Tretschk, E.; Oh, S. J.; and Fritz, M. 2018. Sequential attacks on agents for long-term adversarial goals. In *2. ACM Computer Science in Cars Symposium*.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Watkins, C. J. C. H., and Dayan, P. 1992. Q-learning. *Machine Learning* 8(3):279–292.

Xiang, Y.; Niu, W.; Liu, J.; Chen, T.; and Han, Z. 2018. A pca-based model to predict adversarial examples on q-learning of path finding. In *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, 773–780. IEEE.