Characterizing Attacks on Deep Reinforcement Learning

Xinlei Pan*
UC Berkeley
United States
xinleipan@berkeley.edu

Chaowei Xiao*
NVIDIA, ASU
United States
xiaocw@asu.edu

Warren He
UC Berkeley
United States
_w@eecs.berkeley.edu

Shuang Yang
Alibaba
P.R.China
shuang.yang@antfin.com

Jian Peng
UIUC
United States
jianpeng@illinois.edu

Mingjie Sun
Carnegie Mellon University
United States
mingjies@cs.cmu.edu

Mingyan Liu
University of Michigan, Ann Arbor
United States
mingyan@umich.edu

Bo Li
UIUC
United States
lbo@illinois.edu

Dawn Song
UC Berkeley
United States
dawnsong@cs.berkeley.edu

ABSTRACT

Recent studies show that Deep Reinforcement Learning (DRL) models are vulnerable to adversarial attacks, which attack DRL models by adding small perturbations to the observations. However, some attacks assume full availability of the victim model, and some require a huge amount of computation, making them less feasible for real world applications. In this work, we make further explorations of the vulnerabilities of DRL by studying other aspects of attacks on DRL using realistic and efficient attacks. First, we adapt and propose efficient black-box attacks when we do not have access to DRL model parameters. Second, to address the high computational demands of existing attacks, we introduce efficient online sequential attacks that exploit temporal consistency across consecutive steps. Third, we explore the possibility of an attacker perturbing other aspects in the DRL setting, such as the environment dynamics. Finally, to account for imperfections in how an attacker would inject perturbations in the physical world, we devise a method for generating a robust physical perturbations to be printed. The attack is evaluated on a real-world robot under various conditions. We conduct extensive experiments both in simulation such as Atari games, robotics and autonomous driving, and on real-world robotics, to compare the effectiveness of the proposed attacks with baseline approaches. To the best of our knowledge, we are the first to apply adversarial attacks on DRL systems to physical robots.

KEYWORDS

Adversarial Machine Learning; Reinforcement Learning; Robotics

ACM Reference Format:
Xinlei Pan*, Chaowei Xiao*, Warren He, Shuang Yang, Jian Peng, Mingjie Sun, Mingyan Liu, Bo Li, and Dawn Song. 2022. Characterizing Attacks on Deep Reinforcement Learning. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 13 pages.

*indicates equal contribution.

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

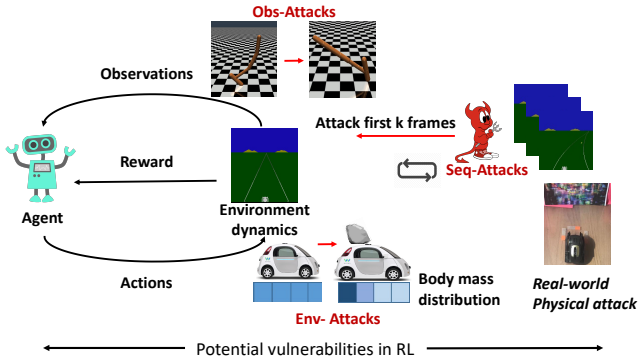


Figure 1: Analysis of adversarial attacks on DRL. RL environments are usually modeled as a Markov Decision Process (MDP), and its *observation* space, and *environment (transition) dynamics* may be attacked. We propose adversarial attacks that have improved computational efficiency by using the sequential nature of MDP, with attacks in both digital and physical environments.

1 INTRODUCTION

With recent progress of DRL in various applications, such as computer games [12, 21, 22], autonomous navigation [9, 26] and robotics [17], the safety and robustness of DRL models are becoming a major concern, especially on real world robotics tasks [17, 30, 35]. Recently, adversarial attacks that have imposed challenges to general deep neural network (DNN) models [14, 18, 33, 37–40] started to challenge the robustness of DRL models. However, the adversarial attacks on DRL models can be different from attacks on DNN models: DRL models focus on sequential decision-making problems while DNN models mostly work on individual prediction problems with no temporal continuity. Existing white-box attacks on DRL models assume almost full access to the victim policy [16]. Some black-box attacks assume partial knowledge of the victim policy [13, 16] but attack each observation individually. These black-box attacks can be computationally intensive especially on tasks with high dimensional inputs. Moreover, most of these attacks have only been evaluated in simulated environments. It remains unclear of their effects on real world DRL models.

In this work, we seek to expand our understanding of the vulnerabilities of DRL systems. To achieve this goal, we propose a set of realistic attacks with improvement on the computational efficiency. To make less assumptions on the victim model, we focus on black-box attack. Based on the components of MDP (shown in Figure 1), we categorize attacks into different types, including attacks on observation space and on environment dynamics. To improve the attack efficiency on multiple high dimensional inputs, we propose to reduce the amount of computation by utilizing the sequential nature of MDP tasks. Finally, to validate the feasibility of adversarial attacks on real world DRL tasks, we perform real world physical attack on a real robot control task. We summarize the proposed attacks on DRL models and our contribution as follows.

Advanced black-box attacks. An attacker may not have access to the DRL system's internals, making white-box attacks infeasible in this case. For DNN models, black-box attacks that take advantage of query access to the victim model have emerged [6]. Based on this progress, we adapt and improve this method for attacking DRL systems and demonstrate the effectiveness. Specifically, starting from the finite difference (FD) based attack [6], we propose an improvement, named *adaptive sampling FD (SFD)*, that reduces the amount of computation by adaptively sampling the input dimensions for gradient estimation. We provide a theoretic analysis of SFD method and prove its efficiency and estimation error bound.

High throughput perturbation. The adaptive SFD method is still inefficient since the attacker would need to generate adversarial examples on each individual frame. To make further improvement, we propose an *online sequential FD attack* based on the fact that DRL tasks have temporal continuity, where temporal consecutive frames tend to correlate with each other. We hypothesize that the attacks generated on a small group of *selected frames* can be applied globally to other similar and temporally close frames. When limiting the number of frames to be selected, it is important to find the best set of frames for generating such attacks. We observe that not all action decisions (frames) are critical within a trajectory, and hypothesize that attacks that are generated based on a small group of important and critical frames can be more effective. Based on this, we propose an *optimal frame selection* approach to select the most important frames to generate the attack. We provide mathematical analysis of this approach.

Perturbations on components other than the observation. As shown in Figure 1, the observation space is not the only component in DRL systems. Besides, environment dynamic is another important component in the DRL systems. Thus, we propose another attacks that perturb the environment transition dynamics by changing physical properties of the environment rather than changing the input observation to the victim model.

Physical attacks. While we have shown the digital adversarial examples on DRL systems, it is still unclear about the possibility to generate physical adversarial examples to attack the physical DRL system. We bridge this gap by using a printed adversarial patch and a toy robot visual navigation task. We show that the overall end-to-end attack is effective even under effects such as manufacturing inaccuracy and varied viewing angles.

We conduct extensive experiments on the above proposed attacks and compare them with existing white-box (which could be viewed as the performance upper bound) and black-box attacks both in simulation and on a real robot. We show that it is feasible to attack real-world DRL systems with our proposed approaches.

2 RELATED WORK

Adversarial attacks on machine learning models. Our attacks draw inspirations from previously proposed attacks. Goodfellow et al. [14] describes the fast gradient sign method (FGSM) of generating adversarial perturbations in a white-box setting. Carlini and Wagner [7] describe additional methods based on optimization, which results in smaller perturbations. Moosavi-Dezfooli et al. [24] demonstrates a way to generate a "universal" perturbation that is effective on multiple inputs. Xiao et al. [39] generates adversarial examples in 3D world by changing the shape and texture information respectively. Evtimov et al. [11] shows that adversarial examples can be robust to natural lighting conditions and viewing angles using real world examples. Furthermore, black-box attacks without providing victim model's training algorithms are also proposed for general machine learning models [8, 31].

Adversarial attacks on DRL. DRL methods train a policy that maps state observations to action decisions. Examples include Deep Q-Learning (DQN) [23] for discrete control, and Deep Deterministic Policy Gradient (DDPG) [19] for continuous control. Proximal Policy Optimization (PPO) [34], and Soft Actor-Critic (SAC) [15] are also proposed recently. We select DQN and DDPG as our target victim algorithms, but our attacks can apply to other RL methods.

Recently, Huang et al. demonstrates an attack that uses FGSM to perturb observation frames in a DRL setting [16]. However, the white-box setting in this work requires knowing the full victim model and the preferred action. They also propose a black-box attack method based on transferability, where the surrogate models are trained to obtain attacks and the generated attacks are then applied on the victim models. We build on FD-based attacks that do not rely on transferability. Lin et al. [20] designs an algorithm to achieve targeted attack for DRL models, and they propose a method to select optimal frames for attacks based on the preference of the policy on the best action over the worst action. We provide related theoretical proofs to demonstrate the optimality of frame selections. Behzadan and Munir [4] propose a black-box attack method that trains another DQN network to minimize the expected return using FGSM. Gleave et al. [13] proposes to train another agent to interact and modify the environment so as to indirectly attack the victim model. The black-box attacks in these related works are all evaluated on simulated environments. We provide real world experiments validating the effectiveness of our proposed attacks. For adversarial attacks on environment dynamics, Pan et al. [28] propose to use candidate inference attack to infer possible dynamics used for training a candidate policy, posing potential privacy-related risk to deep RL models.

Robust RL via adversarial training. Safety and generalization in various robotics and autonomous driving applications have drawn significant attention for training robust models [25, 27, 32]. Knowing how DRL models can be attacked is beneficial for training robust DRL agent. Pinto et al. proposes to train a RL agent to provide adversarial attack during training so that the agent can be robust against dynamics variations [32]. However, since they manually selected the perturbations on environment dynamics, the attack

Table 1: Summary of the adversarial attacks on DRL systems, categorized based on our proposal. The name reflects the category of the attack method. For example, obs-fgsm-wb means attack on observation using fast gradient sign method based white-box attack and obs-fgsm-bb means attack on observation using fast gradient sign method based black-box attack. The attack methods we proposed are highlighted using bold text. "Arch.," "Param.," and "Query" indicate whether the attack requires knowledge of the policy network's architecture, parameters and whether it needs to query the policy network.

Attack	MDP Component	Attacker Knowledge				Real-time	Physical	Temporal Dependency
		White/Black-Box	Arch.	Param.	Query			
obs-fgsm-wb	Observation	White-box	Yes	Yes	Yes	Yes	No	Independent
obs-fgsm-bb	Observation	Black-box	No	No	No	Yes	No	Independent
obs-fd-bb	Observation	Black-box	No	No	Yes	Too slow	No	Independent
obs-sfd-bb	Observation	Black-box	No	No	Yes	slow	No	Independent
obs-seq-fgsm-wb	Observation	White-box	Yes	Yes	Yes	Yes	No	Sequential
obs-seq-fd-bb	Observation	Black-box	No	No	Yes	Yes	No	Sequential
obs-seq-sfd-bb	Observation	Black-box	No	No	Yes	Yes	No	Sequential
env-search-bb	Transition Dynamics	Black-box	No	No	Yes	N/A	Yes	N/A

provided in their work may not be able to generalize to broader RL systems. Additionally, their method relies on an accurate modeling of the environment dynamics, which may not be available for real world tasks such as robotics systems.

3 THREAT MODEL ON DRL

Our victim models are trained by interacting with environments that are Markov Decision Processes (MDPs), which include several components: the state space $\mathcal{S}$, the action space $\mathcal{A}$, the transition dynamics $\mathcal{T}$ and the reward function $\mathcal{R}$. The goal of the DRL model is to learn a policy π so as to maximize the agent's future expected return $\mathbb{E}_\pi[\sum_t \gamma^t r_t]$, where γ is a discount factor. In this work, we provide methods for adversarial attack for trained DRL policies, including discrete control and continuous control methods. We select two representative algorithms: DQN [23] for discrete control and DDPG [19] for continuous control.

We aim to attack well-trained DRL models without accessing the victim model's parameters and only querying the victim model to get model output. The goal of the attacker is to minimize the agent's future expected return. We do not assume these attackers have full control over the agent nor the robotics system. They are weaker than, for example, an attacker that could generally compromise the robot's software. To avoid trivial detection, an adversary needs to constrain the magnitude of perturbation. In this work, we bound the L_{inf} norm of the added perturbation during the evaluation of the digital attacks. For real world physical attacks, we follow the common settings in the literature [2, 11].

4 ADVERSARIAL ATTACKS ON DRL

In this section we develop several concrete attacks to improve the attack feasibility and efficiency. We first introduce some baseline attacks and then describe our new attacks in detail. Table 1 summarizes these attacks, where we categorize them based on their attack components (attack observation or transition dynamics), attacker's knowledge, the computational efficiency of the attack (real-time), whether the attack requires physically changing the environment (physical), and whether the attack is based on temporally dependency of consecutive frames (independent or sequential).

4.1 Baseline Attacks

We discuss both white-box and black-box baseline attack methods.

White-box attacks. In this setting, we assume that the attacker can access the agent's policy network $\pi(a|s)$ where a refers to the action and s refers to the state. Huang et al.[16] has previously introduced one attack in this category that applies the FGSM method to generate white-box perturbation purely on observations. We reproduce this experiment with our **obs-fgsm-wb** attack. This attack's application scenario is when we know the policy network's architecture and parameters.

Black-box attacks. There are different scenarios depending on the attacker's knowledge. In one scenario, the attacker does not have any information about the model architecture, or parameters, and it can't query the model either. In this case, the attacker can perform a "transferability" based attack by attacking a surrogate model (of which it has complete knowledge) and then transfer the perturbation to the victim model. Huang et al.[16] introduced a black-box variant of the FGSM attack using transferability, which we denote as **obs-fgsm-bb**. Their experiments assume that the attacker has access to the same environment, knows the training algorithm of the targeted agent, and uses the same algorithm to train the victim model.

4.2 Advanced Black-Box Attacks

In an alternative black-box setting, the attacker has access to query the model, obtaining the model's outputs on given inputs (while still not knowing the model architecture or parameters). This setting represents a realistic real-world deployment, for example where an executable copy of the agent is shipped to customers, or where an online service is offered. In this setting, we propose more advanced black-box attacks, which take advantage of this query access.

Black-box finite difference (FD) based attack. Baseline black-box attack **obs-fgsm-bb** requires retraining a surrogate policy. Previous work [6] applies the finite difference (FD) method in attacking classification models. We extend the FD method to DRL systems in **obs-fd-bb** which doesn't require retraining a new policy. This attack works in the setting where we don't have the policy network's architecture or parameters and the training algorithms, but can query the network. FD based attack on DRL uses FD to estimate gradient with respect to the input observations. It then generates

perturbations on the input observations by using the estimated gradient. The key step in FD is to estimate the gradient. Denote the loss function as L and state input as $\mathbf{s} \in \mathbb{R}^d$. The canonical basis vector $\mathbf{e}_i$ is defined as an d dimension vector with 1 only in the i-th component and 0 otherwise. The FD method estimates gradients via the following equation

$$\nabla_{\mathbf{s}} L(\mathbf{s}) = \mathbf{FD}(L(\mathbf{s}), \delta) = \Big[\frac{L(\mathbf{s} + \delta \mathbf{e}_1) - L(\mathbf{s} - \delta \mathbf{e}_1)}{2\delta}, \cdots, \frac{L(\mathbf{s} + \delta \mathbf{e}_d) - L(\mathbf{s} - \delta \mathbf{e}_d)}{2\delta} \Big]^\intercal, \tag{1}$$

where δ is a parameter to control estimation accuracy. The loss function L depends on the actual RL algorithm we use, and since we have the model output, we can select an action with a minimal value. Define the state-action value function as $Q(s, a)$, then given a state s, we can obtain a bad target action a_t from the model output as $a_t = \arg\min_a Q(s, a)$. Then we define L to induce the model to select that bad action as follows. Denote the actor of the RL algorithm as π; the loss function for continuous control is: $L(\mathbf{s}) = \|\pi(\mathbf{s}) - a_t\|_2^2$, and the loss function for discrete control is, $L(\mathbf{s}) = \text{CELoss}(\pi(\mathbf{s}), a_t)$, where CELoss() is the cross-entropy loss for classification. In the DQN setting, there is no actor, but we can define an action probability distribution as

$$\pi(\mathbf{s}) = \arg\max_a \frac{\exp(Q(\mathbf{s}, a))}{\sum_{a_i \in \mathcal{A}} \exp(Q(\mathbf{s}, a_i))}.$$

For d dimensional input, the finite difference method would require $2d$ queries to obtain the estimation, which is computationally intensive for high dimensional inputs such as images . Therefore, we propose a sampling technique to mitigate this computational cost.

Adaptive sampling based FD (SFD). Many deep learning models extract features from inputs patch-wise and have sparse activation map [3]. Based on the this observation, we propose a method for estimating gradients that exploits this spatial structure. In this method, we first estimate the gradient with respect to some randomly sampled pixels, then iteratively, we identify pixels where the gradient has a high magnitude and estimate the gradient with respect to surrounding pixels.

Given a function $f(\cdot; w) : \mathbb{R}^d \to \mathbb{R}^1$, where w is the model parameter (we omit this for conciseness below), our goal is to estimate the gradient of f with respect to an input $x \in \mathbb{R}^d$: $\nabla_x \hat{f}(x)$. We define the nontrivial dimension of the gradient of f at x as $\{j \in \{1, 2, \cdots, d\}; |\nabla_j f(x)| \geq \theta\}$, i.e., the dimensions with gradient absolute value greater or equal to a threshold value $\theta > 0$. To estimate nontrivial dimension of the gradient, first, we randomly sample k dimensions from $\{1, \cdots, d\}$, to get a set of dimensions $S = \{S_1, S_2, \cdots, S_k\}$, and use FD to estimate the gradients for dimensions in S. Then we select a set of dimensions $S' = \{j \in S; |\nabla_j f(X; w)| \geq \theta\}$, and use FD to estimate the gradients of the neighbors (a set S'') of dimensions in S', if these gradients haven't been estimated (for dimension i within a d dimensional vector, the neighbor dimension is defined as dimension $i + 1$ if it exists). Then we select dimensions with absolute gradients no less than θ from S'' and find their neighbors to estimate gradients. We repeat this process for multiple iterations. By exploring the sparse large gradients this way, we can adaptively sample dimensions to estimate gradients, which can significantly reduce the number of queries. We give the full attack algorithm of **obs-sfd-bb**. in our appendix. We denote **obs-s[n]fd-bb** as the attack of obs-sfd-bb with n iterations.

To better understand the benefits of SFD, here we provide an analysis of this algorithm and estimate the amount of nontrivial dimension of the gradient that can be estimated using our method in Lemma 1. The basic idea of this lemma is to prove that by using SFD, we can sample more of the nontrivial dimensions of the gradient than by using random sampling. We also provide an error bound for the estimated gradient with SFD in Theorem 2.

Definition 4.1 (Neighbor Dimension's Gradient). *$\forall i, j \in \{1, 2, \cdots, d\}$ and $j = i + 1$, we define the neighbor dimension's gradient as $\nabla_i f(x)^N = \nabla_{j=i+1} f(x)$. Note that $j = i + 1$ is equivalent to $j = i - 1$, and to be general we choose the first one to obtain the definition.*

Definition 4.2 (Non-trivial Gradient Dimension). *Given a positive gradient threshold θ, an input data instance $x \in \mathbb{R}^d$, and a loss function $f : \mathbb{R}^d \to \mathbb{R}^1$, for any dimension $i \in \{1, \cdots, d\}$, if $|\nabla_i f(x)| \geq \theta$, then we define this gradient as non-trivial gradient and the corresponding dimension i as non-trivial gradient dimension. On the other hand, if $|\nabla_i f(x)| < \theta$, then we define this gradient as trivial gradient and the corresponding dimension i as trivial gradient dimension.*

Definition 4.3 (Gradient Sampling Probability). *Given a selected threshold $\theta > 0$ in Algorithm SFD, for any $x \in \mathbb{R}^d$, define the non-trivial gradient sampling probability as,*

$$P_A(\theta) = \frac{1}{|D_A|} \sum_{i \in D_A} 1(|\nabla_i f(x)| \geq \theta), \tag{2}$$

where D_A represents the set of dimensions selected by algorithm A. Therefore, the gradient sampling probability of SFD and random sampling are $P_{SFD}(\theta)$ and $P_{random}(\theta)$ respectively. Some further definitions on neighbor gradient distribution probability are as following:

- *If $|\nabla_i f(x)| \geq \beta + \theta$, then define*

$$\begin{aligned} p(|\nabla_i f(x)^N| \in [\theta, \beta + \theta]) &= q \\ p(|\nabla_i f(x)^N| \in [\beta + \theta, \infty)) &= 1 - q. \end{aligned} \tag{3}$$

- *If $|\nabla_i f(x)| \in [\theta, \beta + \theta]$, then define*

$$\begin{aligned} p(|\nabla_i f(x)^N| \in [0, \theta]) &= p_1 \\ p(|\nabla_i f(x)^N| \in [\theta, \beta + \theta]) &= p_2 \\ p(|\nabla_i f(x)^N| \in [\beta + \theta, \infty)) &= p_3. \end{aligned} \tag{4}$$

Based on the above assumption that these distribution p_1, p_2, p_3 and q are defined over all possible dimensions in one image (over i) and these distribution works throughout the entire gradient estimation iteration process, we have the following lemma.

Lemma 1. *We make the following assumptions on f: $\exists \beta > 0$, s.t. $|\nabla_i f(x) - \nabla_i f(x)^N| \leq \beta, \forall i \in \{1, \cdots, d-1\}, \forall x \in \mathbb{R}^d$. For dimension i whose gradient $|\nabla_i f(x)| \in [\theta, \beta + \theta]$, the probability that the gradient magnitude of its neighborhood pixel is in $[0, \theta]$ is p_1. We conclude, as long as $p_1 < 1 - P_{random}(\theta)$, we have $P_{SFD}(\theta) > P_{random}(\theta)$.*

The intuitive understanding of this lemma is that when nontrivial gradients (of magnitude no less than θ) dimensions are spatially concentrated (p_1 is small, then $p_1 < 1 - P_{random}(\theta)$), our method will be more sample efficient than random sample method. We provide the proof of this Lemma in our appendix. Next we give another theorem about the upper bound for the gradient estimation error and include the proof for this theorem in our appendix.

Theorem 2. *Suppose we sample all nontrivial dimensions of the gradient and estimate the gradient with perturbation strength δ, the estimation error of the gradients is upper bounded by the following inequality,*

$$\|\nabla \hat{f}(x) - \nabla f(x)\|_1 \leq S_\theta C\delta^2 + (d - S_\theta)\theta, \tag{5}$$

for constant $C > 0$, $S_\theta = \sum_{i=1}^{d} 1(|\nabla_i f(x)| \geq \theta)$, and $\nabla \hat{f}(x)$ is the estimated gradient of f with respect to x.

4.3 High Throughput Attacks

A DRL system operates on a sequence of consecutive frames. To develop a useful attack method against real-time DRL systems, we must consider the computational costs. Therefore, in this section, we propose a method for generating perturbations efficiently: an online sequential attack.

In a DRL setting, consecutive observations are not i.i.d.—instead, they are highly correlated and sometimes the consecutive observations do not change too much. It's then possible to perform an attack with less computation than performing the attack independently on each state. Considering real-world cases, for example, an autonomous robot would take a real-time video as input to help make decisions, an attacker is motivated to generate perturbations only based on previous states and apply it to future states, which we refer to as an *online sequential attack*. We hypothesize that a perturbation generated this way is effective on subsequent states.

Therefore, we propose online sequential attacks **obs-seq-fgsm-wb** in whitebox setting and **obs-seq-fd-bb**, **obs-seq-sfd-bb** in blackbox setting. In these attacks, we first collect k observation frames and generate a single perturbation using the averaged gradient on these frames (or estimated gradients using FD or SFD, in the case of **obs-seq-fd-bb** and **obs-seq-sfd-bb**). Then, we apply that perturbation to all subsequent frames. We denote them as **obs-seq[Fk]-fgsm-wb, obs-seq[Fk]-fd-wb,obs-seq[Fk]-sfd-wb**.

Next, instead of using all the k frames, we further improve the above attack by finding the the set of frames that are important and using the gradients from those frames to perform attack. With this, we hope to maintain attack effectiveness while reducing the number of queries needed. We propose to select a subset of frames within the first k frames based on the variance of their Q values. Then, in all subsequent frames, the attack applies a perturbation generated from the averaged gradient. We select an optimal set of important frames with highest value variance to generate the perturbations. We denote them as **obs-seq[Lk]-fgsm-wb, obs-seq[Lk]-sfd-wb, obs-seq[Lk]-sfd-wb**. We give a proof in Corollary 3 below for why attacking these important frames is more effective to reduce the overall expected return. We include the proof in our appendix.

Corollary 3. *Let the state and state-action value be $V(s)$ and $Q(s,a)$ respectively for a policy π with time horizon H. We conclude that $\forall t_1, t_2 \in 1, 2, \cdots, H$, if $Var(Q(s_{t_1}, \cdot)) \geq Var(Q(s_{t_2}, \cdot))$, then $\mathbb{E}_\pi \left[\sum_{t=0}^{H} \gamma^t r_t | do(s_{t_1} = \hat{s}_{t_1})\right] \leq \mathbb{E}_\pi \left[\sum_{t=0}^{H} \gamma^t r_t | do(s_{t_2} = \hat{s}_{t_2})\right]$, where $do(s_{t_1} = \hat{s}_{t_1})$ means the observation at time t_1 is changed from s_{t_1} to $\hat{s}_{t_1}$.*

4.4 Attacks on Components Other than the Observation

Besides attacking the observation space, the attacker can have access to the testing environments of the victim model. Therefore, potentially the attacker can modify the environment such as changing the physical properties of the environment to perform the attack. In this case, the observations of the victim model are not perturbed but the environment transition dynamics will be perturbed.

RL based attacks on environment dynamics. In this case, the attacker will perturb the environment transition model (dynamics). Such transition model is usually non-differentiable with respect to the policy. Therefore, we propose a novel reinforcement learning based method to attack environment dynamics. We describe a targeted attack (in which the agent will fail in a specific way, e.g. a Hopper turn over, or a self driving car drive off the road and hit obstacles.) where the attacker changes the environment dynamics (e.g. by changing the mass of the car). The algorithm is as follows.

Define the environment dynamics as $\mathcal{M}$, the agent's policy as π, the agent's state at step t following the current policy under current dynamics as s_t, and define a mapping from $\pi, \mathcal{M}$ to $s_t{:}s_t \sim f(s_t|\pi, \mathcal{M}, s_0)$, which outputs the state at time step t: s_t given initial state s_0, policy π, and environment dynamics $\mathcal{M}$. The task of attacking environment dynamics is to find another dynamics $\mathcal{M}'$ such that the agent will reach a target state s'_t at step t: $\mathcal{M}' = \arg\min_{\mathcal{M}} \|s'_t - \mathbb{E}_{s_t \sim f(s_t|\pi, \mathcal{M}, s_0)}[s_t]\|$.

We consider the following two algorithms for generating this attack: First, **Random dynamics search**. A naive way to find the target dynamics, which we demonstrate in **env-rand-bb**, is to use random search. Specifically, we randomly propose a new dynamics and see whether, under this dynamics, the agent will reach s'_t. This method works in the setting where we don't need to have access to the policy network's architecture and parameters, but just need to query the network. Second, **RL based adversarial dynamics search**. We design a more systematic algorithm based on RL to search for a dynamics to attack and call this method **env-search-bb**. The algorithm is included in appendix. At each time step, an attacker proposes a change to the current environment dynamics with some perturbation $\Delta\mathcal{M}$, where $\|\Delta\mathcal{M}/\mathcal{M}\|$ is bounded by some constant ϵ, and we find the new state $s_{t,\mathcal{M}'}$ at time step t following the current policy under dynamics $\mathcal{M}' = \mathcal{M} + \Delta\mathcal{M}$, then the attacker agent will get reward $\tilde{r} = 1/\|s_{t,\mathcal{M}'} - s'_t\|$. We demonstrate this in env-search-bb using DDPG [19] to train the attacker. In order to show that this method works better than random search, we also compare with the random dynamics search method, and keep the bound of maximum perturbation $\|\Delta\mathcal{M}/\mathcal{M}\|$ the same.

4.5 Physical Attacks

We discuss how to apply previous proposed FD attack algorithms to generate adversarial perturbations on real world DRL models. We choose visual navigation robots as our experiment platform. There are several challenges to perform real-world attacks in this task. (1) Imperfect camera. Images observed by the robot usually

are captured by cameras, and the computed perturbation may not be directly applicable on real world objects due to camera sensor noise and color shift. (2) Imperfect fabrication process. Using a printer to make the adversarial image patch limits the attacker to printable colors, and there exists a color gap between input image to the printer and output paper from the printer. (3) Imperfect alignment of the patch. Mounting the adversarial image patch at an exact position with a particular orientation is hard. Thus, the attack should be robust to variations of relative position/orientation of the image patch to the robot position. In order to overcome the above challenges, we adapt our algorithm to generate an adversarial patch that is robust against these imperfections. We select a discrete control task for visual navigation as our real world victim model (see details in experiment section).

In order to improve the robustness of the generated patch against various mounting positions, we randomly sample multiple binary masks $\{K_i\}_{i=1}^{n_1}$ and apply the masks to the state (an image) I, such that any part of the patch can be used for attack. One mask is consisted of a rectangular region with value 1 and all other regions with value 0. Denote the generated perturbed image as I', then the masked frame I_i^{masked} with mask K_i could be defined as: $I_i^{masked} = I \odot (1 - K_i) + I' \odot K_i$, where $\odot$ is the element-wise product. To further improve robustness against imperfect printer and variability of mounting orientation, we randomly generate a set of transformations $T = \{T_j\}_{j=1}^{n_2}$ including contrast, brightness adjustments, random rotation adjustments on I^{masked} and get the final image to be optimized: $I_{final} = T(I^{masked})$. Define the function of generating the final image I_{final} as u: $I_{ij}^{final} = T(I_i^{masked})_j = u(I'; I, K_i, T_j)$. Given a trained policy π with parameters θ, the pristine optimal action output is $a^* = \pi_\theta(I)$. We select a target action a', which should have smaller return than a^*. The objective function is shown as follows: $I' = \arg\min_{I'} \sum_{i=1,j=1}^{i=n_1,j=n_2} CE(\pi_\theta(u(I'; I, K_i, T_j)), a')$, where CE denotes the cross entropy loss. We then apply online-sequential method and use sampling based FD-based method obs-seq-sfd-bb to estimate the perturbation.

5 EXPERIMENTS

We design our experiments to answer the following questions: (1) Can our proposed black-box method achieve similar or better performance compared with existing white-box and black-box methods? (2) How does the adaptive SFD method perform compared with FD? (3) How much improvement of sample efficiency does online sequential attack obtain? (4) Does attacking the most important frame selection work better than attacking other frames? (5) Does the RL based environment dynamics attack perform better than random search? (6) Does the real robot attack work in the visual navigation task? To answer these questions, we first introduce the environments we use, and then introduce our baselines and evaluations on all methods.

Experiment environments and victim RL models. We attack several agents trained for five different simulated RL environments: Atari games including Pong and Enduro [5], HalfCheetah and Hopper in MuJoCo [36], and the driving simulation TORCS [29]. We train DQN [23] on Pong, Enduro and TORCS, and train DDPG [19] on HalfCheetah and Hopper. We report the cummulative reward on the first 500 frames. The reward function for TORCS comes from [29] and DQN network architecture comes from [23]. The network for continuous control using DDPG comes from [10].

Baselines. We compare the agents' performance under all attacks with their performance under no attack, denoted as non-adv. We test the white-box attacks with FGSM [14] (obs-fgsm-wb) and blackbox attacks with obs-fgsm-bb [16] which the attacker leverages the transferability to perform attacks by training a surrogate model to generate adversarial examples. We test the attacks on observation under L_∞ perturbation bounds of $\epsilon = 0.005$ and $\epsilon = 0.01$ on the Atari games and MuJoCo simulations and $\epsilon = 0.05$ and $\epsilon = 0.1$ on TORCS. (Observation values are normalized to [0,1].)

Evaluating FD methods. We evaluate the finite difference method obs-fd-bb and test obs-s[n]fd-bb under different numbers n of SFD iterations, for $n \in \{10, 20\}$. Here we denote the attack that uses n iterations as obs-s[n]fd-bb. For online sequential attacks, we test under conditions obs-seq[Fk]-fgsm-wb and obs-seq[Fk]-fd-bb (F for "first"), where we use *all* of the first k frames to compute the gradient for generating a perturbation and then apply the perturbation to the subsequent frames. We report the cumulative rewards of the first k frames and the final cumulative reward among the first 500 frames by applying the perturbation after the first k frames. We implement a baseline method obs-seq[Fk]-rand-bb by using random noise as the perturbation to evaluate the effectiveness of our algorithms. To increase the attack efficiency, we also evaluate the obs-seq[Lk]-fd-bb (L for "largest"), in which we select 20% of the first k frames that have the *largest* Q value variance to generate the universal perturbation, and obs-seq[Sk]-fd-bb (S for "smallest"), in which we select 20% of the first k the frames that have the *smallest* Q value variance to generate the universal perturbation, as baseline.

Perturbations on components other than the observations. For attacks on environment dynamics, we test env-rand-bb and env-search-bb on MuJoCo and TORCS. In the tests on MuJoCo, we perturb the body mass and body inertia vector, which are in $\mathbb{R}^{32}$ for HalfCheetah and $\mathbb{R}^{20}$ for Hopper. In the tests on TORCS, we perturb the road friction coefficient and bump size, which is in $\mathbb{R}^{10}$. The perturbation strength is within 10% of the original magnitude of the dynamics being perturbed.

Real robot experiment. We conduct physical attack experiments on an Anki Vector robot [1]. For training the DRL policy, we design a discrete control task for the robot in a closed playground. The robot has a discrete action space of going forward, turning left, and turning right. It receives a positive reward of 3 for moving forward (in any direction) and a reward of -10 for colliding with anything. The task ends if the robot collides with the wall. We train DQN policy until convergence. We set the maximum episode length to be 200 steps to shorten the training time. We use the attack method in Section 4's real robot attack method to generate perturbation patches that are robust to the imperfections throughout the attack. For the target action a' we choose the worst action under the original input I for attack. We print out the perturbed image I' and crop a random patch from I' and mount it in the robot's current field of view. In order to test the robustness of the attack algorithm, we mount the patch at different positions and put the robot at different viewing angles towards the patch.

Table 2: Cumulative reward of the first 500 frames among different attack methods on Torcs

non-adv	ϵ	obs-fgsm-wb	obs-fgsm-bb	**obs-fd-bb**	**obs-s[n]fd-bb**		online sequential attack										**obs-seq[L60]-s[n]fd-bb**	
							obs-seq[k]-rand-bb		obs-seq[Fk]-fgsm-wb		**obs-seq[Fk]-fd-bb**		obs-seq[Sk]-fd-bb		**obs-seq[Lk]-fd-bb**			
571.4	0.05	5.8	45.2	11.9	n=10	220.6	k=10	581.74	k=10	8.57	k=10	9.40	k=10	571.63	k=10	400.82	n=10	492.2
					n=20	89.8											n=20	483.0
					n=40	**43.2**	k=60	604.18	k=60	22.62	k=60	27.41	k=60	302.52	**k=60**	**30.27**	n=40	334.4
																	n=100	**28.7**
	0.10	5.9	18.6	14.9	n=10	22.3	k=10	583.59	k=10	8.58	k=10	8.47	k=10	525.53	k=10	386.61	n=10	433.2
					n=20	23.6											n=20	391.2
					n=40	**18.9**	k=60	603.43	k=60	17.88	k=60	24.60	k=60	271.73	**k=60**	**22.62**	n=40	136.7
																	n=100	**28.2**

5.1 Experimental Results

The first 6 columns of Table 2 shows the results of the attacks on TORCS, including baseline attacks: obs-fgsm-wb, and obs-fgsm-bb, and finite difference based methods (obs-fd-bb, obs-s[n]fd-bb) on black-box settings. It shows that obs-fd-bb could achieve similar performance compared with whitebox attack (obs-fgsm-wb) and slightly better than the baseline blackbox attack (obs-fgsm-bb). Moreover, for obs-s[i]fd-bb, it shows the effectiveness and with n increase, it will increase the computation cost but the attack effectiveness increases as well. For the results, we could observe that when $n = 40$, it could achieve comparable attack effectiveness compared to obs-fd-bb, therefore, in the following experiments, we select $n = 40$ when we report obs-sfd-bb. In Table 3, we show the number of queries for obs-s[n]fd-bb with different iteration parameter n and the number of queries for obs-fd-bb. The results show that obs-sfd-bb uses significantly fewer queries (around 1000 to 6000) than obs-fd-bb (around 14,000) but achieves similar attack performance.

Table 3: Number of queries for SFD on each image among different settings. (14112 would be needed for FD.)

ϵ	10 iter.	20 iter.	40 iter.	100 iter.
0.05	1233 ± 50	2042 ± 77	3513 ± 107	5926 ± 715
0.10	1234 ± 41	2028 ± 87	3555 ± 87	6093 ± 399

Besides those, we evaluate the performance of online sequential attacks (obs-seq[Fk]-fgsm-wb, obs-seq[Fk]-fd-bb, obs-seq[Lk]-fd-bb, and baselines(obs-seq[Sk]-fd-bb, obs-seq[Fk]-rand-bb) in the columns 8-17 of Table 2 with different L_∞ norm bound ($\epsilon = 0.05$ and 0.1).

We could observe that the baseline obs-seq[Fk]-rand-bb is not effective, while obs-seq[Fk]-fd-bb achieves attack performance close to its white-box counterpart obs-seq[Fk]-fgsm-wb and to non-online sequential attack obs-fd-bb. Even when $k = 10$, the performance is still good. The 14-17 columns of Table 2 shows the results of optimal frame selection. We could observe that when we select a set of states with the largest Q value variance (obs-seq[Lk]-fd-bb) to estimate the gradient, the attack is more effective than selecting states with the smallest Q value variance (obs-seq[Sk]-fd-bb). It also empirically verifies corollary 1. When k is very small ($k = 10$), the estimated universal perturbation may be not strong enough to apply to the following frames while when $k = 60$, the attack performance is reasonably good and similar to obs-seq[F60]-fd-bb. Therefore, in the following settings, we select obs-seq[L60]-fd-bb as our default setting.

Finally, we combine online-sequential attack and sampling based finite difference together to evaluate the performance. We show the results of obs-seq[L60]-s[i]fd-bb by selecting the 20% of frames with the largest Q value variance within the first 60 frames to estimate the gradient and using SFD with i iterations. From Table 2, we could observe that it is clear that with more iterations; we are able to get more accurate estimation of the gradients and thus achieve better attack performance. When $i = 100$, it could achieve comparable attack effectiveness. By looking at Table 3, we could find that when $i = 100$, the number of queries for SFD is around 6k, which is still significantly smaller than needed for FD, which takes 14k queries to estimate the gradient on an image of size 84×84 ($14112 = 84 \times 84 \times 2$).

Table 4: Results of different attacks on other environments. We report the cumulative reward within first 500 frames.

Env	ϵ	non-adv	obs-fgsm-wb	obs-fgsm-bb	**obs-fd-bb**	**obs-seq[L60]-s[100]fd-bb**
Pong	0.005	6.0	-14.0	-14.0	-13.0	-9.0
	0.01		-14.0	-14.0	-13.0	-9.0
Enduro	0.005	43.0	2.0	5.0	2.0	27.0
	0.01		2.0	5.0	2.0	11.0
HalfCheetah	0.005	8257.1	3447.6	2149.7	6173.8	8263.6
	0.01		980.1	1021.5	1273.6	1498.4
Hopper	0.005	3061.4	1703.0	1736.8	1731.2	1843.5
	0.01		1687.2	1694.4	1711.3	1718.8

We provide the results of attack applied on observation space in other environments in Table 4. These environments include Atari games Pong and Enduro, and MuJoCo robotics simulation environments HalfCheetah and Hopper. The results further instantiates the effectiveness of the set of proposed FD methods.

Perturbations on components other than the observations. **Attacks on environment dynamics**. In Table 5, we show our results for performing targeted adversarial environment dynamics attack. The results are the L_2 distance to the target state (the smaller the better). The results show that random search method performs worse than RL based search method in terms of reaching a specific state after certain steps. The quality of the attack can be qualitatively evaluated by observing the sequence of states when the agent is being attacked and see whether the target state has been achieved. The results are shown in figure 3. We could observe that the final stages of our RL based method (env-search-bb) are similar to the targeted state among different games. We include example trajectories of agents under dynamics attack in our appendix.

Real robot policy attack To evaluate the performance of the adversarial patch in the physical world, we print it out and mount it at five different positions (front, left, right, up and below of the robot position) and put the robot at 50 different viewing angles from the adversarial patch (varied from -60° to 60°). We select a target action for attack, which results in suboptimal behavior such as a collision. The physical experiment settings are included in Figure 2. The results of attack success rate are shown in Table 6. The results show that by mounting the patch just in front of the robot results in the best attack success rate both in white-box attack and black-box attack. When there are inaccuracies of patch mounting, such as

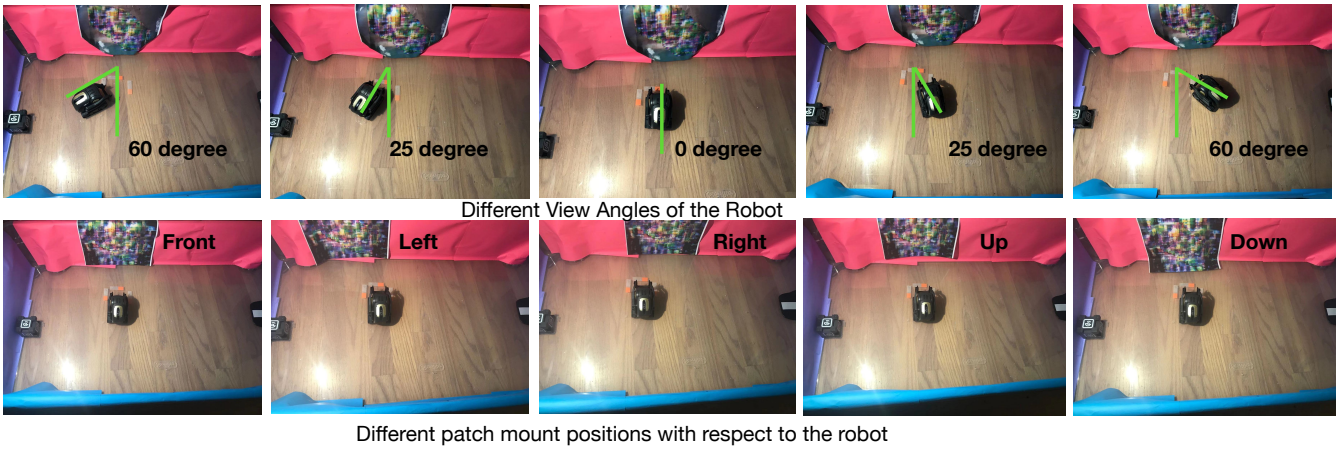


Figure 2: First row: example varied robot states with different viewing angles of the robot towards the patch. (The patch is generated using white-box method) Second row: example varied patching mounting positions with respect to the robot position. (The patch is generated using black-box method) We varied 50 different view angles from the left to the right to evaluate the robustness of the attack.

Table 5: Results of environment dynamics based attacks showing our proposed env-search-bb outperforms baseline env-rand-bb. Shown are the L2 distance to the target state, the smaller the better.

Environment	env-rand-bb	env-search-bb
HalfCheetah	7.91	**5.76**
Hopper	1.89	**0.0017**
TORCS	25.02	**22.75**

(a) Agent's behavior under normal dynamics

(b) Agent's behavior under abnormal dynamics

(c) Agent's behavior under attacked dynamics using RL

(d) Agent's behavior under attacked dynamics using random search

Figure 3: Results for Dynamics Attack on TORCS

mounting the patch to the slight left, right, up or below of the robot field of view, the attack success rate goes down.

Table 6: Attack success rate over all viewing angles by mounting the adversarial image patch with different selected positions.

Method	Positions				
	Front	Left	Right	Up	Below
obs-seq-fgsm-wb	55%	33%	37%	42%	48%
obs-seq-sfd-bb	50%	29%	31%	33%	38%

6 DISCUSSION AND CONCLUSIONS

We propose and evaluate a set of black-box adversarial attacks on DRL models using finite-difference methods. The sample efficiency of FD based methods are further improved by using the adaptive sampling method, online sequential attack and attack frame selection. Most importantly, we provide the first example of adversarial attacks on real world visual navigation robot. Studying the adversarial attack on real robot can help to find potential defense approaches to improve the robustness of DRL models. From the perspective of training robust RL policies, it is important to know the severeness of the risk related with the proposed attacks. Among the proposed attacks, the environment dynamics attack can be a more realistic potential risk to consider than the attacks based on observations. The reason is that this attack does not require access to modify the policy network software system and only requires access to modify environment dynamics, and by modifying the environment dynamics parameters such as changing road condition in autonomous driving, we see from our experiments that the agent tends to fail with the original policy. The observation space based attack, especially the black-box attacks, are also important to defend against, since an attacker can definitely query the network and may have access to change the observations. The real world physical attack experiment by modifying the environment poses a real concern for DRL policies that may be potentially affected. We hope our exploratory work and the analysis of attacks help form a more complete view for what threats should be considered in ongoing research in robust reinforcement learning.

ACKNOWLEDGMENTS

This work is partially supported by the NSF grant No.1910100, NSF CNS 20-46726 CAR, Berkeley Deep Drive (BDD), Berkeley Artificial Intelligence Research, Open Philanthropy and Amazon Research Award. We thank for discussions from the Biomimetic Millisystems Group at UC Berkeley.

REFERENCES

[1] Anki. 2019. Anki | We create robots that move you. https://anki.com/en-us.html.

[2] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. 2017. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397* (2017).

[3] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Network dissection: Quantifying interpretability of deep visual representations. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 3319–3327.

[4] Vahid Behzadan and Arslan Munir. 2017. Vulnerability of Deep Reinforcement Learning to Policy Induction Attacks. *arXiv preprint arXiv:1701.04143* (2017).

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

[6] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. 2017. Exploring the Space of Black-box Attacks on Deep Neural Networks. *arXiv preprint arXiv:1712.09491* (2017).

[7] Nicholas Carlini and David Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *IEEE Symposium on Security and Privacy, 2017*.

[8] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. 2017. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *ACM Workshop on Artificial Intelligence and Security*. 15–26.

[9] Xiaohui Dai, Chi-Kwong Li, and Ahmad B Rad. 2005. An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control. *IEEE Transactions on Intelligent Transportation Systems* 6, 3 (2005), 285–293.

[10] Prafulla Dhariwal, Christopher Hesse, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. 2017. Openai baselines.

[11] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. 2018. Robust Physical-World Attacks on Machine Learning Models. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE.

[12] Imran Ghory. 2004. Reinforcement learning in board games. *Department of Computer Science, University of Bristol, Tech. Rep* (2004).

[13] Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. 2019. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615* (2019).

[14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

[15] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290* (2018).

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

[17] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17, 1 (2016), 1334–1373.

[18] Bo Li and Yevgeniy Vorobeychik. 2014. Feature cross-substitution in adversarial classification. In *Advances in Neural Information Processing Systems*. 2087–2095.

[19] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*.

[20] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. 2017. Tactics of Adversarial Attack on Deep Reinforcement Learning Agents. In *26th International Joint Conference on Artificial Intelligence*.

[21] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*. 1928–1937.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[24] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 1765–1773.

[25] Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. 2018. Assessing Generalization in Deep Reinforcement Learning. *arXiv preprint arXiv:1810.12282* (2018).

[26] Xinlei Pan, Xiangyu Chen, Qizhi Cai, John Canny, and Fisher Yu. 2019. Semantic Predictive Control for Explainable and Efficient Policy Learning. *IEEE International Conference on Robotics and Automation (ICRA)* (2019).

[27] Xinlei Pan, Daniel Seita, Yang Gao, and John Canny. 2019. Risk averse robust adversarial reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 8522–8528.

[28] Xinlei Pan, Weiyao Wang, Xiaoshuai Zhang, Bo Li, Jinfeng Yi, and Dawn Song. 2019. How You Act Tells a Lot: Privacy-Leakage Attack on Deep Reinforcement Learning. *arXiv preprint arXiv:1904.11082* (2019).

[29] Xinlei Pan, Yurong You, Ziyan Wang, and Cewu Lu. 2017. Virtual to real reinforcement learning for autonomous driving. In *British Machine Vision Conference (BMVC)*.

[30] Xinlei Pan, Tingnan Zhang, Brian Ichter, Aleksandra Faust, Jie Tan, and Sehoon Ha. 2020. Zero-shot imitation learning from demonstrations for legged robot visual navigation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 679–685.

[31] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical Black-Box Attacks against Machine Learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM, 506–519.

[32] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. 2017. Robust Adversarial Reinforcement Learning. In *ICML (Proceedings of Machine Learning Research, Vol. 70)*. PMLR, 2817–2826.

[33] Haonan Qiu, Chaowei Xiao, Lei Yang, Xinchen Yan, Honglak Lee, and Bo Li. 2019. SemanticAdv: Generating Adversarial Examples via Attribute-conditional Image Editing. *arXiv preprint arXiv:1906.07927* (2019).

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

[35] Jie Tan, Sehoon Ha, Tingnan Zhang, Xinlei Pan, Brian Andrew Ichter, and Aleksandra Faust. 2021. Systems and Methods for Training a Machine Learned Model for Agent Navigation. US Patent App. 16/717,471.

[36] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 5026–5033.

[37] Chaowei Xiao, Ruizhi Deng, Bo Li, Fisher Yu, Mingyan Liu, and Dawn Song. 2018. Characterizing adversarial examples based on spatial consistency information for semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 217–234.

[38] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. 2018. Generating adversarial examples with adversarial networks. In *IJCAI*.

[39] Chaowei Xiao, Dawei Yang, Bo Li, Jia Deng, and Mingyan Liu. 2019. Meshadv: Adversarial meshes for visual recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6898–6907.

[40] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. 2018. Spatially transformed adversarial examples. *ICLR 2019* (2018).

A ADAPTIVE SAMPLING-BASED FD ALGORITHM

We give the SFD algorithm description in the following Algorithm 1.

Algorithm 1 Adaptive sampling based finite difference (SFD)

Input:
$\mathbf{s} \in \mathbb{R}^d$: state vector
$f(w)$: loss function with parameters w
k: # of item to estimate gradient at each step
n: # of iteration
θ: the gradient threshold
δ: finite difference perturbation value
Output: estimated gradient $\nabla_{\mathbf{s}} \hat{f}(\mathbf{s}; w)$
Initialization:
$\nabla_{\mathbf{s}} \hat{f}(\mathbf{s}; w) \leftarrow 0$, randomly select k dimensions in $\{1, 2, \cdots, d\}$ to form an index set P.
For $t = 0$ **to** n
  For $j \in P$
    If $\nabla_j \hat{f}(\mathbf{s}; w)$hasn't been estimated
      Get $\mathbf{v} \in \mathbb{R}^d$ such that $\mathbf{v}_j = 1$ and $\mathbf{v}_i = 0, \forall i \neq j$
      obtain $\nabla_j \hat{f}(\mathbf{s}; w) = \frac{f(\mathbf{s}+\delta\mathbf{v}; w) - f(\mathbf{s}-\delta\mathbf{v}; w)}{2\delta}$
    end
  end
  $P' = \{k; k \in P, |\nabla_k \hat{f}(\mathbf{s}; w)| \geq \theta\}$
  P = indexes of neighbors of indexes in P'
end

B DYNAMICS ATTACK ALGORITHM

Algorithm 2 Dynamics Attack Algorithm

Input:
π: well trained policy
env: the environment used to train the original policy
T: total training steps
s_{target}: target attack state
n: maximum episodic steps
Initialization:
π_{attack}: RL policies for generating dynamics perturbations
B: replay buffer
$d_{current}$ = env.dynamics
For $t = 0$ **to** T
  $d_{new} = \pi_{attack}$.sample_action()
  env.change_dynamics(d_{new})
  reached_state = evaluate_policy(env, π)
  reward = 1/||reached_state-s_{target}||
  done = episode steps reach n
  action = d_{new}
  B.append($d_{current}$, d_{new}, action, reward, done)
  train_policy(π_{attack}, B)
  $d_{current} = d_{new}$

Details of dynamics attack on Torcs environment. In the Torcs self-driving car environment, we modified the road friction parameter and road bump size to generate novel challenging environments. Specifically, parameters within the XML files that defines these road properties will be modified and recompiled each time a new environment is generated.

C PROOF OF LEMMA 1

Proof. Following the notation in Lemma 1, define

$$P_{random}(\beta + \theta) = \frac{1}{|S_A|} \sum_{i \in S_A} 1(|\nabla_i f(x)| \geq \beta + \theta)$$
$$P_{random}(\theta) = \frac{1}{|S_A|} \sum_{i \in S_A} 1(|\nabla_i f(x)| \geq \theta) \tag{6}$$

Note that the randomness is with respect to the dimensions. These probabilities are, when we randomly choose k dimensions in a gradient vector of d dimensions, the chance of sampling some dimension with gradient absolute value no less than $\beta + \theta$ or θ, respectively. For $\nabla_i f(x)$, its neighbor gradient's absolute value follows the following distribution:

Define $A_0 = P_{random}(\theta+\beta)k, B_0 = [P_{random}(\theta) - P_{random}(\theta+\beta)]k$ as the number of nontrivial gradients estimated in iteration 0, and define

$$A_t = (1-q) * A_{t-1} + p_3 * B_{t-1}$$
$$B_t = q * A_{t-1} + p_2 * B_{t-1}, \tag{7}$$

as the number of nontrivial gradients estimated in iteration t, then the ratio

$$R_t = \frac{A_t + B_t}{A_{t-1} + B_{t-1}}$$

characterizes in every iteration, the ratio of nontrivial gradient estimation over the total number of gradient estimation for $t \geq 1$, while $P_{random}(\theta)$ characterizes nontrivial gradient estimation ratio if using random sampling method. For $t = 0$, $R_0 = P_{random}(\theta)$. Since R_t is the ratio of the number non-trivial gradients over the total number of gradients estimated in the t-th iteration, if we randomly select the dimensions, R_t will be the same as $P_{random}(\theta)$. Using our SFD method, if for every $t > 0$, we have $R_t \geq P_{random}(\theta)$, then overall speaking, the ratio of the number of non-trivial gradients estimated over the total number of gradients estimated, $P_{SFD}(\theta)$, will be greater than $P_{random}(\theta)$. The definition of $P_{SFD}(\theta)$ is

$$P_{SFD}(\theta) = \frac{\sum_{i=1}^{n} A_t + B_t}{\sum_{i=0}^{n-1} A_t + B_t}, \tag{8}$$

where n is the number of iterations.

We now prove that if $p_1 \leq 1 - P_{random}(\theta)$, then for $t > 0$,

$$R_t = \frac{A_t + B_t}{A_{t-1} + B_{t-1}} \geq P_{random}(\theta). \tag{9}$$

More specifically, for $t = 0$, since we perform uniform random sampling, and we sample k dimensions, we have

$$R_0 = \frac{P_{random}(\theta)k}{k} = P_{random}(\theta). \tag{10}$$

Then if we can prove that for $t \geq 1$, $R_t \geq P_{random}(\theta)$, and for some t, we have $R_t > P_{random}(\theta)$, it in turn proves $P_{SFD}(\theta) > P_{random}(\theta)$. The reason is that if for every step in our SFD algorithm the sample efficiency is at least the same as random sampling, and for some iterations our SFD is more efficient, then overall speaking our SFD based gradient estimation is more efficient than random sampling based gradient estimation.

Now from the above definition, we have

$$\begin{aligned} R_t &= \frac{A_t + B_t}{A_{t-1} + B_{t-1}} \\ &= \frac{A_{t-1}(1-q) + p_3 B_{t-1} + q A_{t-1} + p_2 B_{t-1}}{A_{t-1} + B_{t-1}} \\ &= \frac{A_{t-1} + (1-p_1) B_{t-1}}{A_{t-1} + B_{t-1}} \\ &= 1 - \frac{p_1 B_{t-1}}{A_{t-1} + B_{t-1}} \\ &> 1 - p_1 \end{aligned} \tag{11}$$

Therefore, when $p_1 < 1 - P_{random}(\theta)$, $R_t > P_{random}(\theta)$, our sampling algorithm is more efficient than random sampling. □

D PROOF OF THEOREM 2

Proof. when $x \in \mathbb{R}^1$, assume function f is C^∞, by Taylor's series we have

$$\begin{aligned} f(x+\delta) &= f(x) + f'(x)\delta + \frac{\delta^2}{2} f''(x) + \frac{h^3}{3!} f^{(3)}(x) + \cdots \\ f(x-\delta) &= f(x) - f'(x)\delta + \frac{\delta^2}{2} f''(x) - \frac{\delta^3}{3!} f^{(3)}(x) + \cdots . \end{aligned} \tag{12}$$

Combine the two equations we get

$$\frac{f(x+\delta) - f(x-\delta)}{2\delta} - f'(x) = \sum_{i=1}^{\infty} \frac{\delta^{2i}}{(2i+1)!} f^{(2i+1)}(x), \tag{13}$$

which means the truncation error is bounded by $O(\delta^2)$. Moreover, we have

$$\left| \frac{f(x+\delta) - f(x-\delta)}{2h} - f'(x) \right| \le C\delta^2, \tag{14}$$

where $C = \sup_{t \in [x-\delta_0, x+\delta_0]} \frac{f^{(3)}(t)}{6}$, and $0 < \delta \le \delta_0$.

We can regard each dimension as a single variable function $f(x_i)$, then we have

$$\left| \frac{f(x_i+\delta) - f(x_i-\delta)}{2\delta} - f'(x_i) \right| \le C\delta^2. \tag{15}$$

Then $\forall i \in \{1, 2, \cdots, d\}$, assume we are able to sample all nontrivial gradients with absolute gradient value no less than θ, then we have

$$\begin{aligned} \sum_{i \in \{1,2,\cdots,d\}, \nabla_i f(x) \ge \theta} \|\nabla_i \hat{f}(x) - \nabla_i f(x)\|_1 &\le S_\theta C\delta^2 \\ \sum_{i \in \{1,2,\cdots,d\}, \nabla_i f(x) < \theta} \|\nabla_i \hat{f}(x) - \nabla_i f(x)\|_1 &\le (d - S_\theta)\theta. \end{aligned} \tag{16}$$

Therefore, the truncation error of gradients estimation is upper bounded by the following inequality.

$$\|\nabla \hat{f}(x) - \nabla f(x)\|_1 \le S_\theta C\delta^2 + (d - S_\theta)\theta, \tag{17}$$

for some $C > 0$, and $\nabla \hat{f}(x)$ is the estimated gradient of f with respect to x. □

E PROOF OF COROLLARY 3

Proof. Recall the definition of Q value is

$$Q(s_\tau, a_\tau) = \mathbb{E}_\pi[\sum_{t=\tau}^{H-1} \gamma^{t-\tau} r_t | s_\tau, a_\tau]. \tag{18}$$

The variance of Q value at a state s is defined as

$$Var(Q(s)) = \frac{1}{|\mathcal{A}| - 1} \sum_{i=1}^{|\mathcal{A}|} \big(Q(s, a_i) - \frac{1}{|\mathcal{A}|} \sum_{j=1}^{|\mathcal{A}|} Q(s, a_j)\big)^2, \tag{19}$$

where $\mathcal{A}$ is the action space of the MDP, and $|\mathcal{A}|$ denotes the number of actions. Suppose we are to attack state s_m and state s_n where the Q value variance of this two states are $Var(Q(s_m), \cdot)$ and $Var(Q(s_n), \cdot)$, and assume $m < n$.

Denote the state-action pair Q values after attack are $Q(s_m, \hat{a}_m)$ and $Q(s_n, \hat{a}_n)$, respectively. During the attack, state s_m is modified to $\hat{s}_m$, and state s_n is modified to $\hat{s}_n$, and their action's Q-value also change, so we use $\hat{a}_m$ and $\hat{a}_n$ to denote the actions after the attack. By using s_m and s_n instead of $\hat{s}_m$ and $\hat{s}_n$, we mean that though the observed states are modified by the attack algorithm, but the true states do not change. By using a different action notation, we mean that since the observed states have been modified, the optimal actions at the modified states can be different from the optimal actions at the original observed states. Then the total discounted expected return for the entire episode can be expressed as (assume all actions are optimal actions)

$$\begin{aligned} Q' &= Q(s_0, a_0) - \gamma^m Q(s_m, a_m) + \gamma^m Q(s_m, \hat{a}_m), \\ Q'' &= Q(s_0, a_0) - \gamma^n Q(s_n, a_n) + \gamma^n Q(s_n, \hat{a}_n). \end{aligned} \tag{20}$$

Since $m < n$, Q'' can also be expressed as

$$\begin{aligned} Q'' = Q(s_0, a_0) &- \gamma^m Q(s_m, a_m) + \gamma^m Q(s_m, a_m) \\ &- \gamma^n Q(s_n, a_n) + \gamma^n Q(s_n, \hat{a}_n). \end{aligned} \tag{21}$$

Subtract Q' by Q'' we get

$$\begin{aligned} Q' - Q'' &= \gamma^m (Q(s_m, \hat{a}_m) - Q(s_m, a_m)) \\ &\quad + \gamma^n Q(s_n, a_n) - \gamma^n Q(s_n, \hat{a}_n) \\ &= -\gamma^m [Q(s_m, a_m) - Q(s_m, \hat{a}_m) \\ &\quad - \gamma^{n-m} (Q(s_n, a_n) - Q(s_n, \hat{a}_n))]. \end{aligned} \tag{22}$$

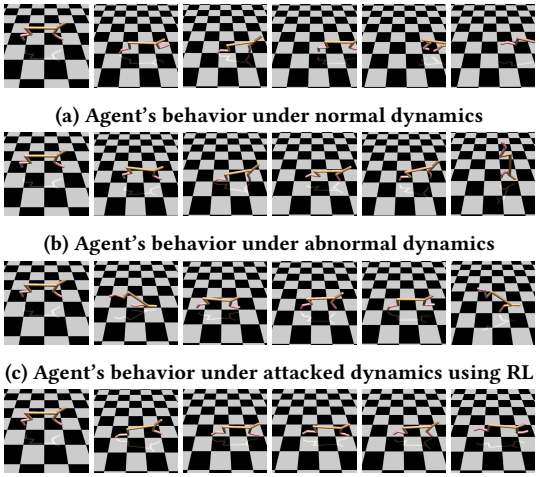
According to our claim that states where the variance of Q value function is small will get better attack effect, suppose $Var(Q(s_m)) > Var(Q(s_n))$, and assume the range of Q value at step m is larger than step n, then we have

$$\begin{aligned} Q(s_m, a_m) - Q(s_m, \hat{a}_m) &> Q(s_n, a_n) - Q(s_n, \hat{a}_n) \\ &> \gamma^{n-m} [Q(s_n, a_n) - Q(s_n, \hat{a}_n)]. \end{aligned} \tag{23}$$

Therefore $Q' - Q'' < 0$ which means attack state m the agent will get less return in expectation. If $Var(Q(s_m)) < Var(Q(s_n))$, assume the range of Q value at step m is smaller than step n, then we have

$$Q(s_m, a_m) - Q(s_m, \hat{a}_m) < Q(s_n, a_n) - Q(s_n, \hat{a}_n). \tag{24}$$

If $n - m$ is very small or $Q(s_n, a_n) - Q(s_n, \hat{a}_n)$ is large enough such that $Q(s_m, a_m) - Q(s_m, \hat{a}_m) < \gamma^{n-m} [Q(s_n, a_n) - Q(s_n, \hat{a}_n)]$, then we have $Q' - Q'' > 0$ which means attacking state m the agent will get more reward in expectation than attacking state n. □



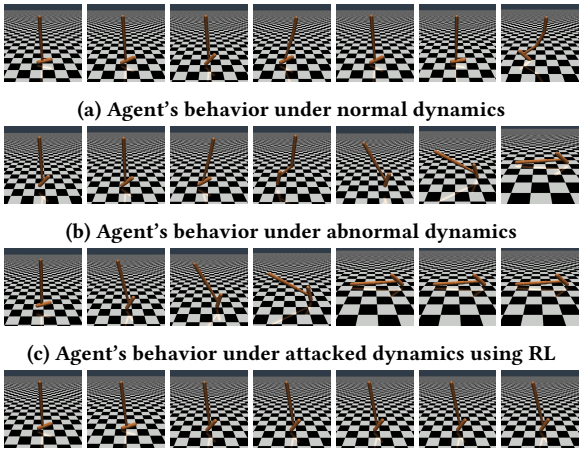
(a) Agent's behavior under normal dynamics

(b) Agent's behavior under abnormal dynamics

(c) Agent's behavior under attacked dynamics using RL

(d) Agent's behavior under attacked dynamics using random search

Figure 4: Results for Dynamics Attack on HalfCheetah



(a) Agent's behavior under normal dynamics

(b) Agent's behavior under abnormal dynamics

(c) Agent's behavior under attacked dynamics using RL

(d) Agent's behavior under attacked dynamics using random search

Figure 5: Results for Dynamics Attack on Hopper

F EXPERIMENTAL SETUP

We trained DQN models on Pong, Enduro, and TORCS, and trained DDPG models on HalfCheetah and Hopper. The DQN model for training Pong and Enduro consists of 3 convolutional layers and 2 fully connected layers. The two network architectures differ in their number of filters. Specifically, the first network structure is $C(4, 32, 8, 4)$–$C(32, 64, 4, 2)$–$C(64, 64, 3, 1)$–$F(3136, 512)$–$F(512, na)$, where $C(c_1, c_2, k, s)$ denotes a convolutional layer of input channel number c_1, output channel number c_2, kernel size k, and stride s. $F(h_1, h_2)$ denotes a fully connected layer with input dimension h_1 and output dimension h_2, and na is the number of actions in the environment. The DQN model for training TORCS consists of 3 convultional layers and 2 or 3 fully connected layers. The convultional layers' structure is $C(12, 32, 8, 4) - C(32, 64, 4, 2) - C(64, 64, 3, 1)$, and the fully connected layer structure is $F(3136, 512) - F(512, 9)$ for one model and $F(3136, 512) - F(512, 128) - F(128, 9)$ for the other model.

The DDPG model for training HalfCheetah and Hopper consists of several fully connected layers. We trained two different policy network structures on all MuJoCo environments. The first model's actor is a network of size $F(dim_{in}, 64) - F(64, 64) - F(64, na)$ and the critic is a network of size $F(dim_{in}, 64) - F(64, 64) - F(64, 1)$. The second model's actor is a network of size $F(dim_{in}, 64) - F(64, 64) - F(64, 64) - F(64, 64) - F(64, na)$, and the critic is a network of size $F(dim_{in}, 64) - F(64, 64) - F(64, 64) - F(64, 64) - F(64, 1)$. For both models, we added ReLU activation layers between these fully connected layers.

For the network structure of obs-nn-wb attack, we use C(12,8,7,1)-C(8,16, 3, 2)-C(16, 32, 3, 2) -R(32,32,3,1)- R(32,32,3,1)-R(32,32,3,1)- R(32,32,3,1)-R(32,32,3,1)-R(32,32,3,1)-C(32,16,3,2)–C(16,8,3,2)-C(8,12, 7,7) for torcs and C(4,8,7,1)-C(8,16, 3, 2)-C(16, 32, 3, 2) -R(32,32,3,1)- R(32,32,3,1)-R(32,32,3,1)- R(32,32,3,1)-R(32,32,3,1)-R(32,32,3,1)-C(32,16,3,2)–C(16,8,3,2)-C(8,4, 7,7) for atari games. R(c1,c2,k,s) denotes a residual block layer of input channl c1, output channel c2, kernel size k, and stride s.

The TORCS autonomous driving environment is a discrete action space control environment with 9 actions, they are turn left, turn right, keep going, turn left and accelerate, turn right and accelerate, accelerate, turn left and decelerate, turn right and decelerate and decelerate. The other 4 games, Pong, Enduro, HalfCheetah, and Hopper are standard OpenAI gym environment.

The trained model's performance when tested without any attack is included in the following Table 7.

Table 7: Model performance among different environments

	Torcs	Enduro	Pong	HalfCheetah	Hopper
Reward	1720.8	1308	21	8257	3061
# of steps	1351	16634	1654	1000	1000

The DDPG neural network used for env-search-bb is the same as the first model (3-layer fully connected network) used for training the policy for HalfCheetah, except that the input dimension dim_{in} is of the perturbation parameters' dimension, and output dimension is also of the perturbation parameters' dimension. For HalfCheetah, Hopper and TORCS, these input and output dimensions are 32, 20, and 10, respectively.

G ADDITIONAL EXPERIMENT RESULTS

Example Trajectory of Agents Under Dynamics Attack. In Figure 4, Figure 5, we show the sequences of states when the agents are under attack with the random search or reinforcement learning based search method. The last image in each sequence denotes the state at same step t. The last image in each abnormal dynamics rollout sequence corresponds to the target state, the last image in the attacked dynamics using RL search denotes the attacked results using env-search-bb, and the last image in the attacked dynamics using random search denotes the attacked results using env-rand-bb. It can be seen from these figures that env-search-bb method is very effective at achieving targeted attack while using random search, it is relatively harder to achieve this.

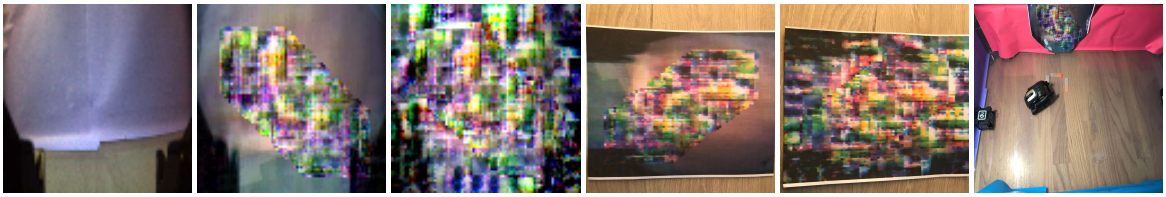


Figure 6: Original robot observation (first), optimized perturbed image with white-box approach (second), optimized perturbed image with black-box approach (third), printed perturbation patch with white-box approach (forth), printed perturbation patch with black-box approach (fifth) physical environment with the playground, adversarial patch and the robot (sixth).

Real world attack additional results. We include in Figure 6 the original robot observation, the optimized perturbed image, the actual printed perturbed image patch and the bird-view of the robot environment after the patch has been mounted.