CHARACTERIZING VULNERABILITIES OF DEEP REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep Reinforcement learning (DRL) has achieved great success in various applications, such as playing computer games and controlling robotic manipulation. However, recent studies show that machine learning models are vulnerable to adversarial examples, which are carefully crafted instances that aim to mislead learning models to make arbitrarily incorrect prediction, and raised severe security concerns. DRL has been attacked by adding perturbation to each observed frame. However, such observation based attacks are not quite realistic considering that it would be hard for adversaries to directly manipulate pixel values in practice. Therefore, we propose to understand the vulnerabilities of DRL from various perspectives and provide a throughout taxonomy of adversarial perturbation against DRL, and we conduct the first experiments on unexplored parts of this taxonomy. In addition to current observation based attacks against DRL, we propose attacks based on the actions and environment dynamics. Among these experiments, we introduce a novel sequence-based attack to attack a sequence of frames for real-time scenarios such as autonomous driving, and the first targeted attack that perturbs environment dynamics to let the agent fail in a specific way. We show empirically that our sequence-based attack can generate effective perturbations in a blackbox setting in real time with a small number of queries, independent of episode length. We conduct extensive experiments to compare the effectiveness of different attacks with several baseline attack methods in several game playing, robotics control, and autonomous driving environments.

1 INTRODUCTION

In recent years, deep neural networks (DNNs) have become pervasive and led a trend of fast adoption in various commercial systems performing image recognition (Krizhevsky, Sutskever, and Hinton, 2012), speech recognition (Hannun et al., 2014), and natural language processing (Sutskever, Vinyals, and Le, 2014). DNNs have also encouraged increased success in the field of deep reinforcement learning (DRL), where the goal is to train an agent to interact with the environments for maximizing a reward function. DRL systems have been evaluated on games (Ghory, 2004; Mnih et al., 2013; 2016), autonomous navigation (Dai, Li, and Rad, 2005), and robotics control (Levine et al., 2016), etc. However, it is well-known that DNNs are vulnerable to adversarial perturbations (Goodfellow, Shlens, and Szegedy, 2015; Li and Vorobeychik, 2014; 2015). DRL systems that use DNNs to perform perception and policy making also have similar vulnerabilities. For example, one of the main weaknesses of RL models in adversarial environments is their heavy dependence on the input observations. Since RL models are trained to solve sequential decision-making problems, an attacker can perturb multiple observations. In fact, the distribution of training and testing data could be different due to random noise and adversarial manipulation (Laskov and Lippmann, 2010). Therefore, the learned policy can be vulnerable in adversarial environments.

In this paper, we first present an extensive study of adversarial perturbation attacks on DRL systems. Second, we propose and evaluate 12 adversarial perturbation attacks in order to explore points in the taxonomy that have not previously been examined in the literature. We organize adversarial perturbation attacks on DRL into a taxonomy based on details of the threat model and other properties of the attack, foremost (i) what part of the system the attacker is capable of perturbing and (ii) what knowledge the attacker needs to perform the attack. We summarize these categories in Figure 1, and will discuss this taxonomy further in Section 3.

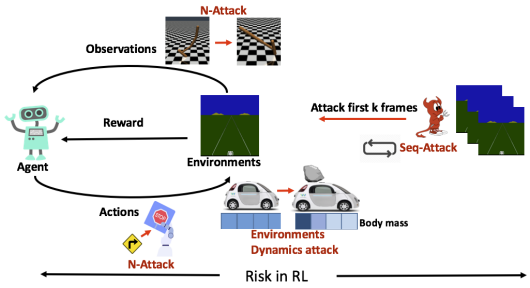

Figure 1: Taxonomy of adversarial perturbation attacks on deep reinforcement learning (DRL) systems.

Previous work on attacking DRL systems have focused on a narrow part of our proposed taxonomy: perturbations of the observation, and mostly with whitebox access to the agent. We outline these existing attacks in Section 2. We explore several other potential attacks in our taxonomy by devising 12 novel attacks and evaluating them on several DRL environments. On the other hand, existing attacks that perturb the observation operate independently on each frame, which requires high computation power to run in real time. We propose two novel strategies for *quickly* creating adversarial perturbations to use in real-time attacks. The first strategy, *N-attack*, trains a neural network to generate a perturbation, reducing the computation to a single forward pass over this network. The second strategy, *online sequential attack*, generates a perturbation using information from a few frames and then applies the generated perturbation to later steps. We include our experiments with these strategies as part of our exploration of attacks that perturb the observation. We describe our attacks in detail in Section 4, and we present our evaluation in Section 5.

To summarize, our contributions are: (1) We systematically organize adversarial perturbation attacks on DRL systems into a taxonomy; (2) We propose two practical strategies, N-attack and online sequential attack, for performing real-time attacks on DRL systems; (3) We introduce the first attack that adversarially perturbs a DRL system's environment dynamics, and the first attack that perturbs a DRL system's actions; (4) We devise and evaluate 12 new attacks on several DRL environments.

2 Related work

Adversarial attacks on machine learning models Our attacks draw some of their techniques from previously proposed attacks. Goodfellow, Shlens, and Szegedy (2015) describe the fast gradient sign method (FGSM) of generating adversarial perturbations in a whitebox setting. Carlini and Wagner (2017) describe additional methods based on optimization, which result in smaller perturbations. Moosavi-Dezfooli et al. (2017) demonstrate a way to generate a "universal" perturbation that is effective on a set of multiple inputs. Evtimov et al. (2018) show that adversarial examples can be robust to natural lighting conditions and viewing angles. In our blackbox attacks, we apply techniques that have been proposed for adapting whitebox methods to blackbox scenarios Papernot et al. (2017); Chen et al. (2017). Recently, Huang et al. (2017) demonstrate an attack that uses FGSM to perturb observation frames in a DRL setting. They experiment with a whitebox attack and a blackbox attack based on transferability. In this work, we propose novel blackbox attacks without depending on transferability and we propose several ways to reduce the computational complexity of attack. Lin et al. (2017) designed an algorithm to achieve targeted attack for deep RL models. However, their work only considers targeted attack and requires training a generative model to predict future states, which is already a very computational intensive task. Behzadan and Munir (2017) propose a blackbox attack method that trains another DQN network to minimize the reward while still using FGSM as the attack method.

Robust RL via Adversarial Training. Saftey in various robotics and autonomous driving applications has drawn lots of attention for training robust models. Knowing how RL models can be attacked is beneficial for training robust RL agent. Pinto et al. (2017) proposed to train an RL agent to provide adversarial attack during training so that the agent can be robust against dynamics variations. However, since they manually selected the perturbation on environment dynamics, the attack 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 and autonomous driving systems.

3 TAXONOMY OF VULNERABILITIES IN DRL

Existing work on attacking DRL systems with adversarial perturbations focuses on perturbing an agent's observations. This is the most appealing place to start, with seminal results already suggesting that recognition systems are vulnerable to adversarial examples (Goodfellow, Shlens, and Szegedy, 2015; Moosavi-Dezfooli et al., 2017). It naturally follows that we should ask whether perturbations introduced in other places in a RL system can cause the agent to misbehave, and in what scenarios, taking into account (i) can the attacker perform the attack with limited knowledge about the agent, (ii) can the attacker perform the attack in real time, and (iii) can the attacker introduce the perturbation physically. In order to help systematize the exploration of this open-ended question, we propose a taxonomy of possible adversarial perturbation attacks.

Attacker capability: In the first layer of our proposed taxonomy, we divide attacks into categories based on what components in an MDP the attacker chooses to perturb in a RL system: the agent's *observation*, the agent's *action*, and *environment dynamics*. This categorization affects where these attacks will be feasible, depending on the integrity of these components of real systems.

Attacker knowledge: In the second layer of our proposed taxonomy, we categorize attacks based on what information the attacker needs access to in order to perform the attack. This divides attacks into *whitebox* attacks, where the attacker has full access to the agent, and *blackbox* attacks, where the attacker has less or no access to the agent. **Whitebox:** In whitebox attacks, the agent has access to the architecture and weights of the policy network, and knowledge of the data used to train the agent. **Blackbox:** In blackbox attacks, the attacker has limited or no knowledge of the agent. Previous work in adversarial examples targeting other machine learning tasks has introduced some general techniques for adapting ideas from whitebox attacks to work under blackbox settings. We identify the use of these techniques as part of our taxonomy.

Further categorization: We consider these additional properties of attacks. **Real-time:** While some attacks require more computation than an attacker can perform in the time range of a single step, some are fast enough to run in real time. Still other attacks perform a precomputation phase and then are able to generate perturbations quickly for each step. We identify this pragmatic property as part of our taxonomy. **Physical:** For RL tasks that take place in the real world, this property concerns the feasibility of physically applying the perturbation. We distinguish attacks that make this easy as part of our taxonomy. **Temporal dependency:** In this part of our taxonomy, we distinguish between attacks that generate a perturbation in each frame *independently* from other frames and *sequential* attacks that attacks that, in contrast, consider a step in the context of previous steps to generate perturbations.

4 ADVERSARIAL ATTACKS ON REINFORCEMENT LEARNING POLICIES

In order to study the previously unexplored parts of our proposed taxonomy from Section 3, we develop several concrete attacks. In this section, we describe these attacks in detail. Table 1 summarizes these attacks. In our attacks, the attacker's goal is modeled as a different reward function $\tilde{r}$. As a concrete example, we experiment an attacker that tries to minimize the agent's reward r, with $\tilde{r} = -r$.

4.1 ATTACKS ON STATE OBSERVATIONS

We now describe attacks that perturb an agent's state observations. In this category of attacks, the attacker changes the input state observation s to $\tilde{s} = s + h(s; w)$, where the attacker generates perturbation $h(s; w)$ from the original observation s and some learned parameters w. In order to ensure that perturbations are small, we require that $||h(s; w)||_\infty \leq \epsilon$, which we can enforce by choosing h to be of the form $\epsilon \tanh(\cdot)$. Following our taxonomy of attacks, we present both whitebox attacks and blackbox attacks in this category.

Name	Attacker capability	Attacker knowledge	Real-time	Physical	Temporal dependency
obs-fgsm-wb	Observation	Whitebox	Yes	No	Independent
obs-opt-wb	Observation	Whitebox	Too slow	No	Independent
obs-n-wb	Observation	Whitebox	Yes	No	Independent
obs-fgsm-bb	Observation	Transferability	Yes	No	Independent
obs-opt-bb	Observation	Transferability	Too slow	No	Independent
obs-imi-bb	Observation	Query access	Yes	No	Independent
obs-fd-bb	Observation	Query access	Too slow	No	Independent
obs-fdadap-bb	Observation	Query access	Too slow	No	Independent
obs-seq1-fd-bb	Observation	Query access	Yes	No	Sequential
obs-seq10-fd-bb	Observation	Query access	Yes	No	Sequential
obs-seqopt-fd-bb	Observation	Query access	Yes	No	Sequential
act-opt-wb	Action	Whitebox	Yes	No	Independent
env-ran-bb	Env. dynamics	No information	N/A	Yes	N/A
env-search-bb	Env. dynamics	Query access	N/A	Yes	N/A

Table 1: Summary of our adversarial perturbation attacks on DRL systems in this paper. Categorized based on our proposed taxonomy. The name reflects the category of the attack method. For example, obs-n-wb means attack on observation using neural network based whitebox attack.

Whitebox 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 during attack. Huang et al. (2017) has previously introduced one attack in this category that applies the gradient of the reward with respect to the input observation, using FGSM. In this case, the perturbation is computed purely from the observation s, and w is unused. We reproduce this experiment with our obs-fgsm-wb attack, and we go on to propose new attacks in this category. First, we propose a variant of Huang et al.'s attack that replaces FGSM with an optimization based method in obs-opt-wb. Second, we propose a new attack obs-n-wb where the perturbation $h(s, w)$ is a deep neural network. We train the parameters w of this network based on the policy π such that the policy receives low reward in expectation when the perturbations are applied: $w = \arg\max_w \mathbb{E}_{\pi(a|\tilde{s})}[\sum_t \gamma^t \tilde{r}_t] = \arg\max_w \mathbb{E}_{\pi(a|s+h(s,w))}[-\sum_t \gamma^t r_t]$.

With a fixed victim policy π, this attack is similar to training a policy. For example, in DQN, our goal is to perform gradient update on w based on the following loss function: $L(w) = (Q(s+h(s,w),a) - (\tilde{r} + \gamma \max_{a'} Q(s'+h(s',w),a')))^2$, where Q is the model under attack, s' is the next state relative to current state s. In continuous control using DDPG, our goal is to perform gradient update on w based on the following loss function $L(w) = (\tilde{r} + \gamma Q(s'+h(s',w), \mu(s'+h(s',w))) - Q(s+h(s,w),a))^2$, where Q is the value function and μ is the actor function.

Blackbox attacks: In this setting, we propose attacks that have no knowledge of the policy $\pi(a|s)$. Huang et al. (2017) include a blackbox variant of their FGSM attack using transferability, which we reproduce in obs-fgsm-bb and an optimization based variation obs-opt-bb. We propose new attacks not based on transferability, where the attacker can query the victim policy during simulation.

Imitation learning based black-box attack: This attack obs-imi-bb is inspired by Rusu et al.'s work on policy distillation (2015). The attacker trains a surrogate policy $\hat{\pi}(a|s,\theta)$ such that it imitates the policy π on states from a training set and and the victim policy's outputs from. Then the attacker uses a whitebox method on the surrogate policy to generate a perturbation and uses that perturbation on the victim policy. Formally, in a Deep Q learning case, given a blackbox policy π^T with access to its policy outputs, we collect some dataset $\mathcal{D}^T = \{(s_i, \mathbf{q}_i)\}_{i=0}^N$, where each data sample consists of a short observation sequence s_i and a vector $\mathbf{q}_i$ which is the unnormalized Q-values, and one value corresponds to one action. We will perform imitation learning to learn a new policy $\pi^S(\cdot|s,\theta)$ such that we minimize the following loss function by taking gradient update with respect to network parameters θ: $L(\mathcal{D}^T, \theta) = \sum_{i=1}^{|\mathcal{D}|} \text{softmax}(\frac{\mathbf{q}_i^T}{\tau}) \ln \frac{\text{softmax}(\frac{\mathbf{q}_i^T}{\tau})}{\text{softmax}(\mathbf{q}_i^S)}$, where T is the victim policy, S is our surrogate policy, and τ is a temperature factor.

Finite difference (FD) based blackbox attack: Bhagoji et al.'s previous work (2017) applies the finite difference method in attacking classification models. We extend the finite difference method to DRL systems in obs-fd-bb. We denote the loss function as L and state input as $\mathbf{s} \in \mathbb{R}^n$. Then the canonical basis vector $\mathbf{e}_i$ is defined as an n dimension vector with 1 only in the i-th component and 0 otherwise. Finite difference method estimates gradients via the following equation $\mathbf{FD}(L(\mathbf{s}), \delta) = \left[\frac{L(\mathbf{s}+\delta\mathbf{e}_1) - L(\mathbf{s}-\delta\mathbf{e}_1)}{2\delta}, \cdots, \frac{L(\mathbf{s}+\delta\mathbf{e}_n) - L(\mathbf{s}-\delta\mathbf{e}_n)}{2\delta}\right]^\intercal$, where δ is a parameter to control estimation accuracy. However, for n dimensional input, the finite difference method would require

$2n$ queries to obtain the estimation, which is computationally intensive for high dimensional inputs such as images. We use propose a sampling technique to lower this computational cost.

Adaptive sampling based FD (SFD): Considering the fact that many deep learning models extract features from inputs patch wise and have sparse activation map (Bau et al., 2017), we propose to iteratively estimate the gradients of high dimensional inputs and apply adaptive sampling to sample candidate dimensions for querying. We give a brief overview of this attack obs-fdadap-bb, with a more detailed algorithm and analysis in Appendix B. First, we randomly sample certain dimensions and use finite difference to estimate the corresponding gradients for these dimensions. Then we sample the neighbors of these dimensions and select the ones whose estimated gradients are larger than some threshold value θ. We repeat this process for certain iterations. By exploring the sparsity of gradients, we can adaptively sample dimensions to estimate gradients. In this way, we can significantly reduce the amount of queries.

Considering real-world cases, for example, an autonomous vehicle 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*.

Universal attack based approach: In obs-seq$k$-fd-bb, we propose to generate universal perturbation from the first k states, for $k \in 1, 10$. Note that in a DRL setting, consecutive observations do not follow an i.i.d distribution, which means the data instances themselves are highly related (next state depends on previous one). Therefore, we propose to compute gradients for some states and use those gradients to generate a perturbation for all subsequent states. When generating perturbation from the previous states, both whitebox and blackbox attacks can be used. We experiment specifically with blackbox attacks based on finite differences.

Optimal selection of initial frames: We improve upon the above attack by finding the optimal set of states from which to estimate gradients. In obs-seqopt-fdbb, we propose to select a subset of states within the first k based on the variance of their Q values. Then, in all subsequent frames, the attack applies a perturbation generated from the averaged gradient. Intuitively, the Q value function's variance is roughly a measure of how certain the policy is about performing the current action. If the variance of the Q value is small, it means most actions' value are fairly close, which in turn means the current state may be less important for the overall success. Therefore, attacking states with small variance of Q value is less effective than attacking states with high variance of Q value function. We give an informal proof for why attacking these important frames is more effective, and we give the full proof in Appendix C.

Corollary 1. *Let the state and state-action value be $V(s)$ and $Q(s,a)$ respectively, and let the state with higher variance of Q value be state s_{t_1} and the state with smaller variance of Q value be s_{t_2}. Let the current policy be π. We have*

$$\mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t r_t | do(s_{t_1} = \hat{s}_{t_1})] \leq \mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t r_t | do(s_{t_2} = \hat{s}_{t_2})], \tag{1}$$

where $do(s_{t_1} = \hat{s}_{t_1})$ means the observation to the policy is changed from s_{t_1} to $\hat{s}_{t_1}$ while the underlying true state is not changed.

4.2 Attacks on action selection

Our second category of attacks is to directly attack action output and minimize the expected reward. We experiment with one attack in this category, under a whitebox scenario, act-opt-wb. Here we train another policy network that takes in the state s and outputs a perturbation on the Q function: $Q'(s,a,w)$, the goal is also to minimize the reward. For example, in DQN, the loss is chosen to be $L(w) = (Q(s,a) + Q'(s,a,w) - \tilde{r} - \gamma \max_{a'}(Q(s',a') + Q'(s',a',w)))^2$. For DDPG, the loss is chosen to be $L(w) = (Q(s, a = \mu(s)) + Q'(s, a = \mu(s), w) - \tilde{r} - \gamma(Q(s', a' = \mu(s')) + Q'(s', a' = \mu(s'), w)))^2$, where $\tilde{r} = -r$ is reward that captures the attacker's goal of minimizing the victim agent's reward.

4.3 Attacks on environment dynamics

In this third category, attacks perturb the environment transition model. In our case, we aim to achieve targeted attack, which means we want to change the dynamics such that the agent will fail

in a specific way. 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: $\mathcal{M}' = \arg\min_{\mathcal{M}} \|s'_t - \mathbb{E}_{s_t \sim f(s_t|\pi,\mathcal{M},s_0)}[s_t]\|$. **Random dymamics search:** A naive way to find the target dynamics, which we demonstrate in env-rand-Bb, is to use random search. At every time step, we randomly propose a new dynamics and see whether, under this new dynamics, the agent will reach s'_t. **Adversarial dynamics search:** We design a more systematic algorithm based on RL to search for a dynamics to attack. At each time step, an attacker agent 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 (Lillicrap et al., 2016) to train the attacker agent. 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.

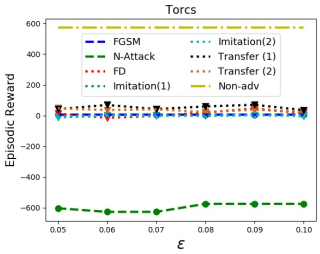

Figure 2: Episodic Reward

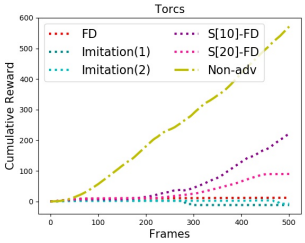

Figure 3: Cumulative Reward (ϵ=0.05)

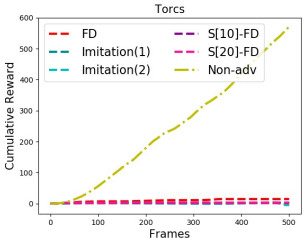

Figure 4: Cumulative Reward (ϵ=0.10)

Figure 5: Episodic reward under different attack methods and cumulative reward of different blackbox attacks on TORCS.

5 Experiments

5.1 Experiment Setup

We first train RL models on Pong, Enduro, HalfCheetah, Hopper and TORCS environments. We train DQN (Mnih et al., 2015) on Pong, Enduro and TORCS, and we train DDPG (Lillicrap et al., 2016) on HalfCheetah and Hopper environments. The state pre-processing and neural network architecture for DQN come from Mnih et al. (2015). The network architecture for continuous control using DDPG comes from Dhariwal et al. (2017). For each game, we train the above agents with different random seeds and different architectures in order to test the transferability and imitation learning based blackbox attack. Details of network structure and the performance for each game are included in Appendix A.

5.2 Experimental Design

Now we describe the details of attacks on state observations, actions and environment dynamics.

State Attack-Whitebox Attack. We implement the neural network based attack method to attack models trained in Atari game Bellemare et al. (2013), MuJoCo Todorov, Erez, and Tassa (2012), and TORCS Pan et al. (2017) autonomous driving environment. We train a DNN based policy to achieve attack by applying perturbation on the input state. We vary the amount of perturbations by varying the L_∞ bound ϵ to evaluate the effectiveness of our attack method. We also test the performance for different L_∞ bounds.

State Attack-Blackbox Attack 1: Imitation Learning Based Attack. We consider two cases of black box imitation learning attack. In the first case, we assume we know the architecture of the original model but can't perform back-propagation on the original model. We train another policy

network with the same architecture to imitate the original policy in the same training environment with different random seed. Then we use FGSM to attack the imitated policy and also the original policy. We denote this method as `imitation(1)`. In the second case, we assume we don't know the network architecture, and we use a different policy network architecture compared with the original model and train the policy network to imitate the original policy. Then we apply our FGSM attack on the imitated policy, and use the same attack to attack the blackbox policy. We denote this method as `imitation(2)`. Both `imitation(1)` and `imitation(2)` are evaluated on all 5 games.

State Attack-Blackbox Attack 2: Transferability Based Attack . For testing transferability with different random seeds, two policy networks with the same architecture are trained with different random seeds and the attack obtained using FGSM from one model is applied on another model. We denote this attack as `transfer(1)`. For testing transferability with different policy network architecture, we train two models with different policy networks but the same random seed, then we apply attack obtained using FGSM on one model onto another model to test the transferability across models. We denote this kind of attack as `transfer(2)`.

State Attack-Blackbox Attack3 : Finite Difference Based Attack . We evaluate the efficiency of Finite Difference(FD) and adaptive sampling based Finite Difference(SFD) methods. We use `S[i]FD` to represent the number of iterations in SFD. We show that the number of queries is significantly reduced and we provide the actual number of queries in the appendix.

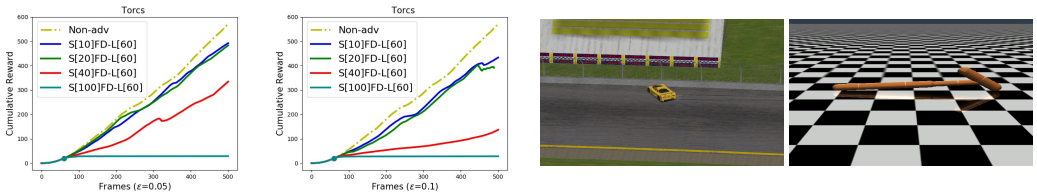

Figure 6: Performance of universal perturbation generated based on different numbers of query iterations with SFD (the two images on the left); final state of environment dynamics attack on TORCS and Hopper, where the result for TORCS shows the top view of the driving environment. (The two images on the right).

State Attack-Seq-Attack : Universal Finite Difference Attack. We evaluate the following different forms of finite difference (FD) based sequential attack. In the first case, we use finite difference to estimate gradients from the first k frames in one episode to generate an universal perturbation, then add this perturbation on all subsequent frames (except the first k frames). We denote this method as `FD[k]`. In order to further reduce the time complexity, we test the effect of optimal frame based selection algorithm. To show that attacking states with large Q value variance is better than attacking states with small Q value variance, we select 20 percent of the states from the first k frames with the largest Q value variance and also 20 percent of the states from the first k frames with the smallest Q value variance, and apply finite difference attack on all subsequent frames. We denote these two cases by `FD-L[k]` and `FD-S[k]`, respectively. Finally, we apply our adaptive sampling based approach using finite difference. Everything is the same as in `FD-L[k]` approach but the gradients are estimated using our adaptive sampling finite difference method. We denote this method as `S[i]FD-L[k]`, where i is the iteration number for estimating gradients and k is the number of first k frames used to estimate universal gradients using finite difference. Additionally, we also apply random perturbation bounded by their L_∞ norm on states to compare, and to show that random perturbation can not achieve the desired attack effect. We denote this method as `Random`. For all the above finite difference methods, we evaluate two perturbation cases of two L_∞ norm bound, and the values are 0.05 and 0.1 on torcs and 0.005 and 0.01 on atari and mujoco.

Attack Action Selection. We use the action selection attack method proposed in our method section to perform the attack on several Atari game and TORCS environments.

Attack Environment Dynamics. We attack two MuJoCo environments and the TORCS environment. In the MuJoCo environment, we first find a specific state where we want to attack and record its observation. Then we run our algorithms to find a dynamics under which the agent can reach the

predefined target state following its policy. On the two MuJoCo environments, we change the body mass and body inertia vector; on the TORCS environment, we change the road friction coefficient and bump size.

5.3 EXPERIMENT RESULTS

The results of whitebox attack (attack on states), blackbox attack leveraging imitation learning (`Imitation(1)`, `Imitation(2)`), blackbox attack using transferability (`Transfer(1)` and `Transfer(2)`) and blackbox finite difference attack (FD) for TORCS are included in figure 2. This figure also contains the result for testing the original policy without any attack (`Non-adv`). The results of whitebox and blackbox on atari and mujoco games are shown in appendix and the results of whitebox attack on action for all games and TORCS environment are included in appendix.

Figure 2 shows that our whitebox attack method achieves better attack results on TORCS than the FGSM method. The proposed Imitation(1)(2) and FD can achieve better performance than transferability based algorithm on blackbox setting.

Figure 3 and 4 compare the cumulative rewards among different blackbox attack methods and Non-adversarial setting. It is shown that the policy is vulnerable to all of the blackbox methods. Further, it also shows that `S[i]FD` based on different number of iterations can achieve similar performance with FD under different bounds of perturbation ϵ.

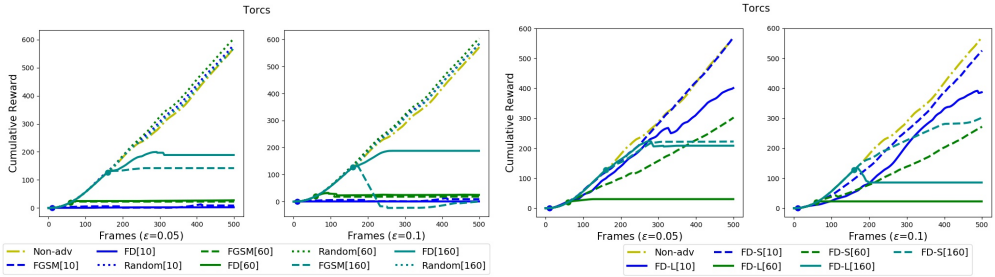

Figure 7: Performance of sequential attack approach (left two images) and achieved dynamics attacked state in TORCS and Hopper environment (right two images). **Left:** Cumulative reward after adding vanilla universal perturbation generated by FGSM, FD, and Random perturbation. **Right:** Cumulative reward after adding optimal state based universal perturbation.

The results for comparing finite difference method `FD[k]`, `FGSM`, and random perturbation is shown in Figure 7 (left) for perturbation of different L-infinity norm bound (ϵ =0.05 and 0.1). The two figures shows the cumulative reward for one episode when the states are under attack. Comparing our proposed finite difference method with FGSM (whitebox) and random perturbation method, it shows that our method can achieve close attack performance compared with FGSM and is significantly better than random perturbation method.

Figure 7 (right) shows that when we select an optimal set of frames to estimate the gradient, the attack effect is better than selecting states with the smallest Q value variance, which indicates that selecting frames based on their Q value variance (selecting states with large Q value variance) is more effective. The two figures show the results for attacking effect when the perturbation's L-infinity norm is bounded by 0.1 and 0.05 respectively.

In Figure 24 (left), we show the results of `S[i]FD-L[k]` by varying the number of iterations i. It is clear that with more iterations, we are able to get more accurate estimation of the gradients and thus achieve better attack performance, while the total number of queries is still significant reduced. We conclude in supplementary material that when the $i = 100$, the number of queries for SFD is around 6k, which is significantly smaller than the vanilla FD, which takes 14k number of queries to estimate the gradient on an image of size 84×84 (14112 = $84 \times 84 \times 2$). In Figure 24 (right), we show the final state of TORCS visually, where the attacked car finally hits the wall.

In Table 2, we show our results for performing targeted adversarial environment dynamics attack. Our goal is to attack the environment dynamics such that the original well trained agent will fail in a

pre-specified way, for example, for a Hopper to turn over and for a self driving car to drive off road and hit obstacles. The results show that random search method performs worse than RL based search method in terms of reaching a specific state after certain steps.

Table 2: Environment Dynamics Attack Result: L2 distance to the target state.

Environment	Random Search	Reinforcement Learning	Perturbed Dynamics
HalfCheetah-v1	7.91	5.76	Body Mass & Body Inertia $\in \mathbb{R}^{32}$
Hopper-v1	1.89	0.0017	Body Mass & Body Inertia $\in \mathbb{R}^{20}$
TORCS	25.02	22.75	Friction & Bump Size $\in \mathbb{R}^{10}$

6 CONCLUSION

We are the first to propose a systematic view of adversarial attack against neural network policies. Dynamic decision-making system trained using RL algorithms can be highly vulnerable to adversarial attacks. Our work shows that when the neural network being trained on is available, it is able to completely pit the original agent. More importantly, even if the agent's network structure or the model's parameters are not available at attack time, we are still able to achieve near white-box level attack effect by querying the original black-box policy network or by training a new policy to imitate the original policy. Traditional attack was only applied on state observations while our work for the first time shows that when the dynamics of the environment is perturbed, the original trained model would almost certainly fail and it is able to change the dynamics to a certain conformation such that the original agent will fail in a specific way. The safety of real world RL system is currently a big concern in deploying these models. Our proposed several attack methods can be useful reference when training robust reinforcement learning systems.

REFERENCES

Bau, D.; Zhou, B.; Khosla, A.; Oliva, A.; and Torralba, A. 2017. Network dissection: Quantifying interpretability of deep visual representations. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 3319–3327. IEEE.

Behzadan, V., and Munir, A. 2017. Vulnerability of deep reinforcement learning to policy induction attacks. *arXiv preprint arXiv:1701.04143*.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47:253–279.

Bhagoji, A. N.; He, W.; Li, B.; and Song, D. 2017. Exploring the space of black-box attacks on deep neural networks. *arXiv preprint arXiv:1712.09491*.

Carlini, N., and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy, 2017*.

Chen, P.-Y.; Zhang, H.; Sharma, Y.; Yi, J.; and Hsieh, C.-J. 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.

Dai, X.; Li, C.-K.; and Rad, A. B. 2005. An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control. *IEEE Transactions on Intelligent Transportation Systems* 6(3):285–293.

Dhariwal, P.; Hesse, C.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; and Wu, Y. 2017. Openai baselines.

Evtimov, I.; Eykholt, K.; Fernandes, E.; Kohno, T.; Li, B.; Prakash, A.; Rahmati, A.; and Song, D. 2018. Robust physical-world attacks on machine learning models. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE.

Ghory, I. 2004. Reinforcement learning in board games. *Department of Computer Science, University of Bristol, Tech. Rep*.

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

Hannun, A.; Case, C.; Casper, J.; Catanzaro, B.; Diamos, G.; Elsen, E.; Prenger, R.; Satheesh, S.; Sengupta, S.; Coates, A.; et al. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

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

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Laskov, P., and Lippmann, R. 2010. Machine learning in adversarial environments. *Machine learning* 81(2):115–119.

Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17(1):1334–1373.

Li, B., and Vorobeychik, Y. 2014. Feature cross-substitution in adversarial classification. In *Advances in Neural Information Processing Systems*, 2087–2095.

Li, B., and Vorobeychik, Y. 2015. Scalable optimization of randomized operational decisions in adversarial classification settings. In *AISTATS*.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2016. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*.

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 *26th International Joint Conference on Artificial Intelligence*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

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.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 1928–1937.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; Fawzi, O.; and Frossard, P. 2017. Universal adversarial perturbations. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 1765–1773. IEEE.

Pan, X.; You, Y.; Wang, Z.; and Lu, C. 2017. Virtual to real reinforcement learning for autonomous driving. In *British Machine Vision Conference (BMVC)*.

Papernot, N.; McDaniel, P.; Goodfellow, I.; Jha, S.; Celik, Z. B.; and Swami, A. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 506–519. ACM.

Pinto, L.; Davidson, J.; Sukthankar, R.; and Gupta, A. 2017. Robust adversarial reinforcement learning. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, 2817–2826. PMLR.

Rusu, A. A.; Colmenarejo, S. G.; Gulcehre, C.; Desjardins, G.; Kirkpatrick, J.; Pascanu, R.; Mnih, V.; Kavukcuoglu, K.; and Hadsell, R. 2015. Policy distillation. *arXiv preprint arXiv:1511.06295*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 5026–5033. IEEE.

A 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.

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 3.

Table 3: Model performance among different environments

	Torcs	Enduro	Pong	HalfCheetah	Hopper
Episodic reward	1720.8	1308	21	8257	3061
Episodic length	1351	16634	1654	1000	1000

B FINITE DIFFERENCE GRADIENT ESTIMATION USING SFD

Algorithm 1 shows the detailed steps in our obs-fdsamp-bb attack.

B.1 ERROR BOUND ANALYSIS

Here we provide an analysis of our SFD sampling algorithm and estimate the amount of non-trivial gradients that can be estimated using our method. We also give an upper bound of the truncation error of gradient estimation using finite difference method for high dimensional functions such as neural networks. The setting is that given a function f with parameter w: $f(w) : X \to Y$, where $X \in \mathbb{R}^n$ and $Y \in \mathbb{R}^1$, our goal is to estimate the gradient $\nabla_X f \in \mathbb{R}^n$. In order to simplify analysis, we use a sampling algorithm which is an analogy of the algorithm used in the paper with one-dimensional input: At the start, we uniformly sample k points. Then in each iteration, for each of the points in the previous iteration whose gradient magnitudes are no less than a threshold value θ, we sample one point in the neighborhood of this point in a certain direction. The proof of the corollary is in the appendix.

Corollary 2. *We focus on estimating gradients of dimensions where the gradient magnitude is no less than a threshold $\theta > 0$, and assume that the gradients at other dimensions can be ignored. Denote e_i as the i-th dimension variable of X, we make the following assumptions on f:*

- *A1: The gradient of the function f satisfies β-smoothness: $|\nabla_{e_i} f - \nabla_{e_{i+1}} f| \leq \beta$.*
- *A2: Assumptions on the absolute value of gradient at neighboring dimensions. Suppose the current sampled point is e_i, and the next sampled point is e_j, where $j = i+1$ or $j = i-1$. If*

Algorithm 1: Adaptive sampling based finite difference (ASFD) algorithm

input: $s \in \mathbb{R}^d$: State vector;
Q: Q value network;
k: # of item to estimate gradient at each step ;
n: # of iteration;
δ: finite difference perturbation value ;
output: gradient $\nabla_s Q(s)$;

$Initialization$:$\nabla_s Q(s) \leftarrow \mathbf{0}$;
Randomly select k indexes from $\{1, 2, \cdots, d\}$ to form an index set P ;
for $t \leftarrow 0$ **to** n **do**
  for $j \in P$ **do**
    /* Estimate gradient for position j */;
    Get $v \in \mathbb{R}^d$ such that $v_j = 1$ and $v_i = 0, \forall i \neq j$;
    Set $\nabla_s Q(s)_j = \frac{Q(s+\delta v) - Q(s-\delta v)}{2\delta}$;
  end
  P' = Find top $k/2$ largest gradient value's positions from P;
  P = indexes of neighbors of P';
end
Return: $\nabla_s Q(s)$

$\beta + \theta \leq |\nabla_{e_i} f|$, then with probability $0 < q < 1$, $\theta \leq |\nabla_{e_j} f| \leq \beta + \theta$ and with probability $1 - q$, $|\nabla_{e_j} f| \geq \beta + \theta$. If $\theta \leq |\nabla_{e_i} f| \leq \beta + \theta$, then with probability q, $|\nabla_{e_j} f| \leq \theta$ and with probability $1 - q$, $\theta \leq |\nabla_{e_j} f| \leq \beta + \theta$.

We show that in each round of sampling, the probability of sampling dimensions with gradient magnitude no less than θ is larger than $1 - q$. Further, denote $P(\theta)$ as the probability of sampling dimensions with gradient magnitude no less than θ using random sampling, we also have that as long as $1 - q > P(\theta)$, then in each round of sampling, our algorithm has higher probability of sampling dimensions with nontrivial gradients (magnitude no less than θ) than random sampling method, where the probability of sampling nontrivial gradients is $P(\theta)$.

Furthermore, suppose there are S_θ dimensions in X where the magnitude of gradients are no less than θ, and assume we estimated in total S_θ dimensions' effective gradients using a perturbation strength of h. Then the truncation error of gradients estimation is upper bounded by the following inequality.

$$\|\nabla_X \hat{f} - \nabla_X f\|_1 \leq S_\theta C h^2 + (n - S_\theta)\theta, \tag{2}$$

where C is some constant, and $\nabla_X \hat{f}$ is the estimated gradient of f with respect to X.

B.2 Proof of Corollary 2

Proof. We use $P(x)$ to denote the probability that the gradient magnitude of a dimension is no less than x. Initially, we sample k points uniformly. We have that out of k sampling points, there are $P(\beta + \theta) \cdot k$ points whose gradient magnitudes are no less than $\beta + \theta$. Similarly, we have $P(\theta) \cdot k$ points where the gradient magnitudes are no less than θ. At the second iteration, we start from $P(\theta) \cdot k$ points. According to the transition dynamics, we have $(1 - q)P(\beta + \theta) \cdot k$ points whose gradient magnitudes are no less than $\beta + \theta$. Similarly, we have $q \cdot P(\beta + \theta) \cdot k + (1 - q)[P(\theta) - P(\beta + \theta)]k$ points with gradient magnitudes in the range $[\theta, \beta + \theta]$. Now in t^{th} iteration, we denote a_t as the number of points in this iteration with gradient magnitudes no less than $\beta + \theta$ and b_t as the number of points in this iteration with gradient magnitudes in range $[\theta, \beta + \theta]$. The recurrence relationship is as follows:

$$a_t = (1 - q) \cdot a_{t-1}, \quad b_t = q \cdot a_{t-1} + (1 - q) \cdot b_{t-1}, \tag{3}$$

where $a_1 = (1-q)k \cdot P(\beta+\theta)$ and $b_1 = (1-q)k \cdot P(\theta) + (2q-1)k \cdot P(\beta+\theta)$. We have:

$$\begin{aligned} \frac{b_t}{(1-q)^t} &= \frac{q}{(1-q)^2} a_1 + \frac{b_{t-1}}{(1-q)^{t-1}} \\ \frac{b_t}{(1-q)^t} &= \frac{q}{(1-q)^2} a_1 (t-1) + \frac{b_1}{(1-q)}. \end{aligned} \tag{4}$$

Then we have:

$$a_t = (1-q)^{t-1} a_1, \quad b_t = (1-q)^{t-1} \cdot [(t-1)\frac{q}{1-q} \cdot a_1 + b_1] \tag{5}$$

Now we show that in each iteration, the probability that our algorithm samples points with gradient magnitudes no less than θ is larger than that of random sampling ($P(\theta)$).

$$\frac{a_t + b_t}{a_{t-1} + b_{t-1}} = (1-q) \cdot \frac{a_1 + t \cdot \frac{q}{1-q} \cdot a_1 + b_1}{a_1 + (t-1)\frac{q}{1-q} \cdot a_1 + b_1} > 1 - q \tag{6}$$

From above, we can see that in each iteration, we have at least probability $1-q$ to sample the points with gradient magnitudes no less than θ. Now since q measures the degree of steepness between two points (typically around $1/2$) and when the gradient distribution is sparse, $P(\theta)$ is small if θ is relatively large. Therefore as long as $1 - q > P(\theta)$, then our sampling algorithm outperforms the random sample counterpart.

Now we prove the second part of our corollary: When $x \in \mathbb{R}^1$, assume function f is C^∞, by Taylor's series we have

$$\begin{aligned} f(x+h) &= f(x) + f'(x)h + \frac{h^2}{2} f''(x) + \frac{h^3}{3!} f^{(3)}(x) + \cdots \\ f(x-h) &= f(x) - f'(x)h + \frac{h^2}{2} f''(x) - \frac{h^3}{3!} f^{(3)}(x) + \cdots . \end{aligned} \tag{7}$$

Combine the two equations we get

$$\frac{f(x+h) - f(x-h)}{2h} - f'(x) = \sum_{i=1}^{\infty} \frac{h^{2i}}{(2i+1)!} f^{(2i+1)}(x), \tag{8}$$

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

$$\left| \frac{f(x+h) - f(x-h)}{2h} - f'(x) \right| \leq Ch^2, \tag{9}$$

where $C = \sup_{t \in [x-h_0, x+h_0]} \frac{f^{(3)}(t)}{6}$, and $0 < h \leq h_0$.

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

$$\left| \frac{f(x_i+h) - f(x_i-h)}{2h} - f'(x_i) \right| \leq Ch^2, \tag{10}$$

Then for those dimensions where the gradient magnitudes are no less than θ, then from the above estimation, the total truncation error caused by these dimensions are no greater than $S_\theta C h^2$. For those dimensions where the gradient magnitudes are less than θ, then we can see that the error caused by not considering these dimensions are no greater than $(n - S_\theta)\theta$. All in all, the truncation error of gradients estimation is upper bounded by the following inequality.

$$\|\nabla_X \hat{f} - \nabla_X f\|_1 \leq S_\theta C h^2 + (n - S_\theta)\theta, \tag{11}$$

where C is some constant, and $\nabla_X \hat{f}$ is the estimated gradient of f with respect to X.

□

C PROOF OF COROLLARY 1

Proof. Recall the definition of Q value is

$$Q(s_\tau, a_\tau) = \mathbb{E}[\sum_{t=\tau}^{T} \gamma^{t-\tau} r_t | s_\tau, a_\tau]. \tag{12}$$

Assume a fixed horizon of length H. Suppose we are to attack state s_m and state s_n where the Q value variance of this two states are $Var(Q(s_m))$ and $Var(Q(s_n))$, and assume $m < n$. Denote the state action pair Q value after attack is $Q'(s_m, a_m)$ and $Q''(s_n, a_n)$. Then the total discounted expected reward for the entire episode can be expressed as

$$\begin{aligned} Q' &= Q(s_0, a_0) - \gamma^m Q(s_m, a_m) + \gamma^m Q'(s_m, a_m) \\ Q'' &= Q(s_0, a_0) - \gamma^n Q(s_n, a_n) + \gamma^n Q''(s_n, a_n). \end{aligned} \tag{13}$$

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

$$\begin{aligned} Q'' =& Q(s_0, a_0) - \gamma^m Q(s_m, a_m) + \gamma^m Q(s_m, a_m) \\ & - \gamma^n Q(s_n, a_n) + \gamma^n Q''(s_n, a_n). \end{aligned} \tag{14}$$

Subtract Q'' by Q' we get

$$\begin{aligned} Q' - Q'' =& \gamma^m (Q'(s_m, a_m) - Q(s_m, a_m)) + \gamma^n Q(s_n, a_n) \\ & - \gamma^n Q''(s_n, a_n) \\ =& - \gamma^m [Q(s_m, a_m) - Q'(s_m, a_m) \\ & - \gamma^{n-m} (Q(s_n, a_n) - Q''(s_n, a_n))]. \end{aligned} \tag{15}$$

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

$$\begin{aligned} Q(s_m, a_m) - Q'(s_m, a_m) &> Q(s_n, a_n) - Q''(s_n, a_n) \\ &> \gamma^{n-m} [Q(s_n, a_n) - Q''(s_n, a_n)]. \end{aligned} \tag{16}$$

Therefore $Q' - Q'' < 0$ which means attack state m the agent will get less reward 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

$$Q(s_m, a_m) - Q'(s_m, a_m) < Q(s_n, a_n) - Q''(s_n, a_n). \tag{17}$$

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

Table 4: number of query for SFD on each image among different settings. (The query time for FD is 14112)

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

D RESULTS FOR DYNAMICS ATTACK

We include here the environment rollout sequence for dynamics attack experiment in figure 7, figure 8 and figure 9. 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, and the last image in the attacked dynamics using RL sequence denotes the attacked results using our method. It can be seen from these figures that our method is very effective at achieving targeted attack while using random search, it is very hard to achieve this.

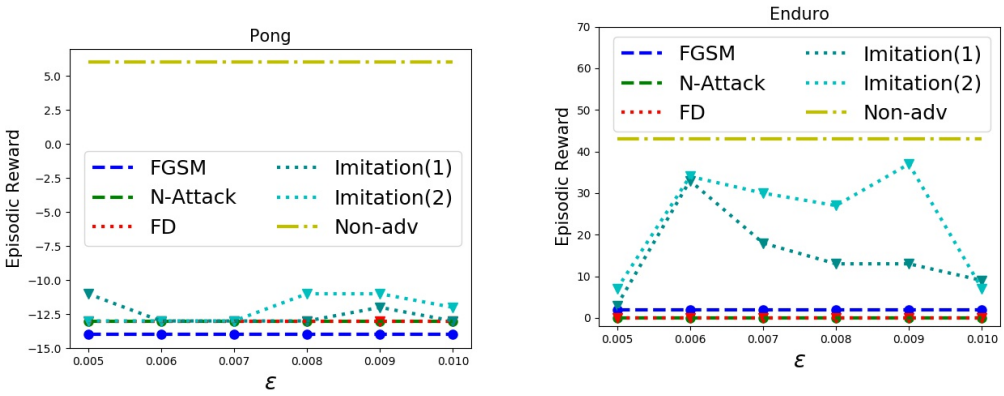

Figure 8: Episodic rewards among different attack methods on Atari games. Dotted lines are blackbox attack while dash lines are whitebox attack.

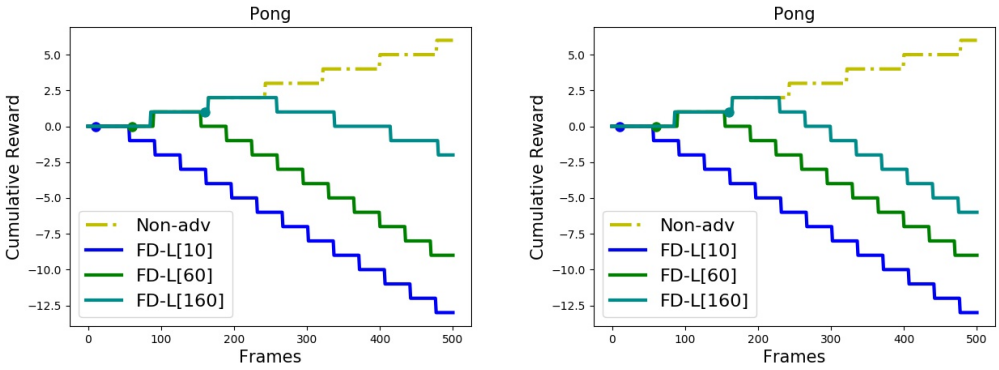

Figure 9: ϵ= 0.005

Figure 10: ϵ= 0.01

Figure 11: Cumulative reward after adding optimal state based universal perturbation on Pong game

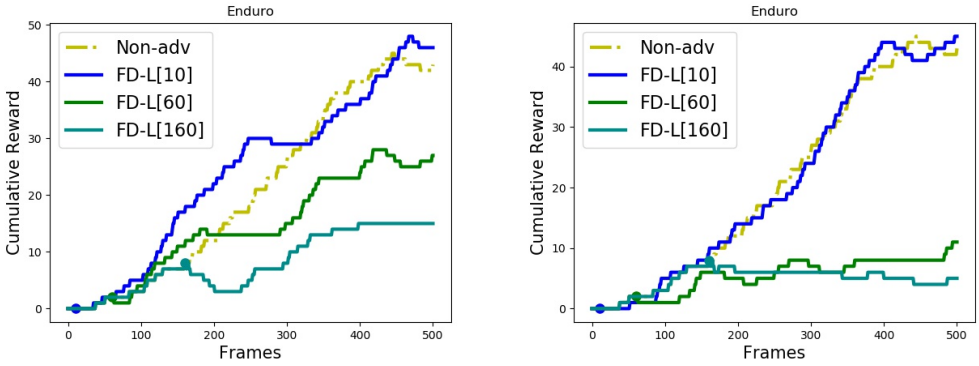

Figure 12: ϵ=0.005

Figure 13: ϵ=0.01

Figure 14: Cumulative reward after adding optimal state based universal perturbation on Enduro game

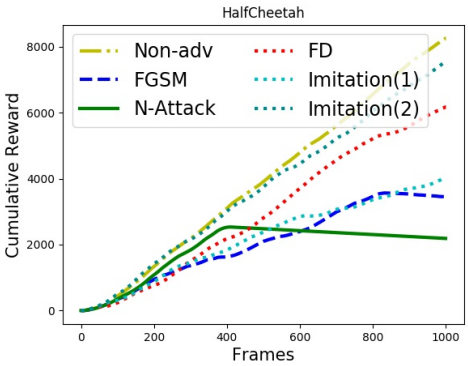

Figure 15: ϵ=0.005 — HalfCheetah

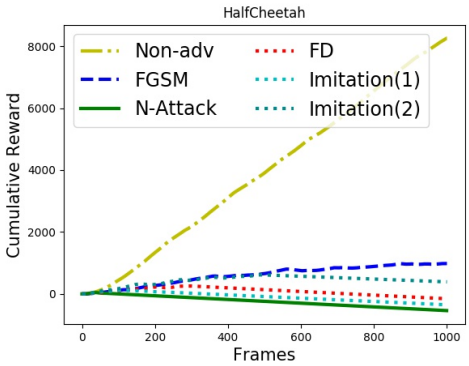

Figure 16: ϵ=0.01 — HalfCheetah

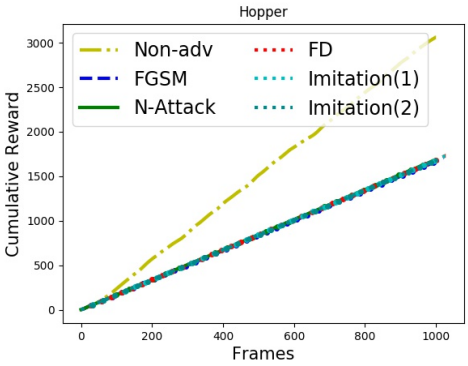

Figure 17: ϵ=0.005 — Hopper

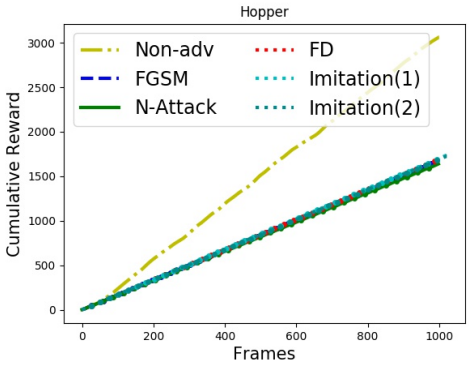

Figure 18: ϵ=0.01 — Hopper

Figure 19: Performance among different attack methods on Mujoco. We use the format "L_∞ bound—Environment" to label the settings of each image.

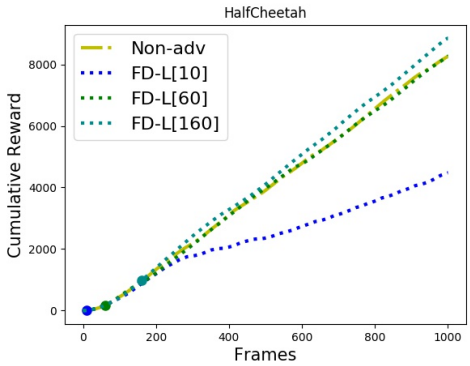

Figure 20: ϵ=0.005 — HalfCheetah

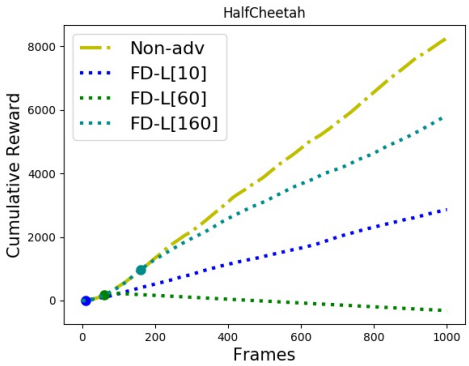

Figure 21: ϵ=0.01 — HalfCheetah

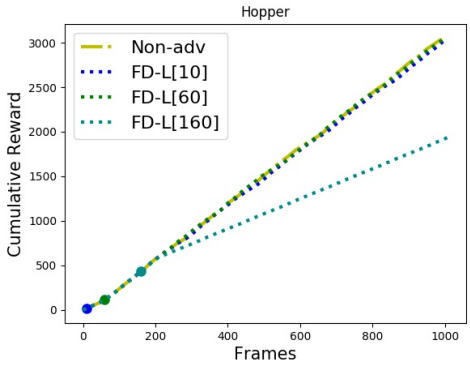

Figure 22: ϵ=0.005 — Hopper

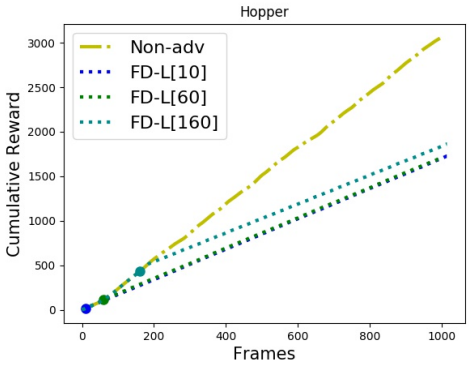

Figure 23: ϵ=0.01 — Hopper

Figure 24: Cumulative reward after adding optimal state based universal perturbation on Mujoco. We use the format "L_∞ bound— Environment" to label the settings of each image.

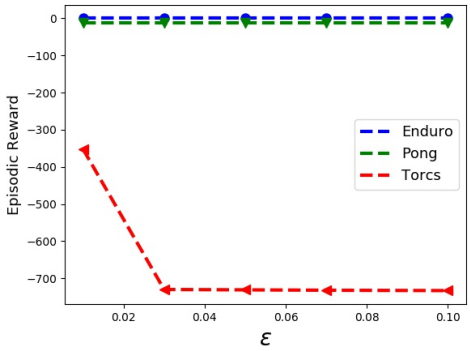

Figure 25: Atari and Torcs

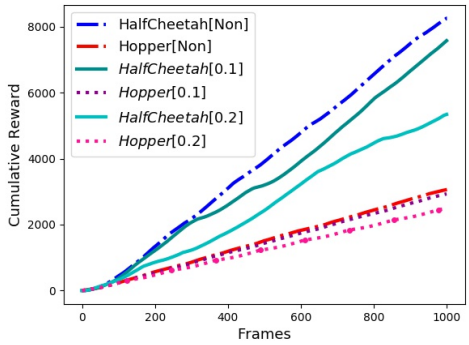

Figure 26: Mujoco

Figure 27: Attack Action

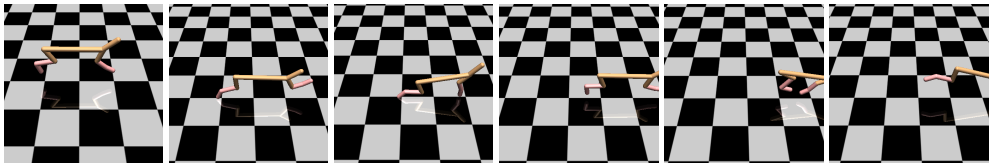

Figure 28: Behavior of HalfCheetah Following Learned Policy under Normal Dynamics

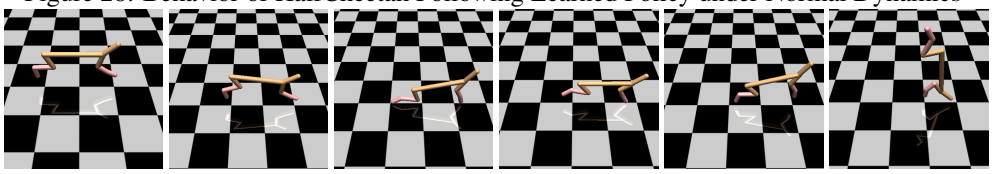

Figure 29: Behavior of HalfCheetah Following Learned Policy under Abnormal Dynamics

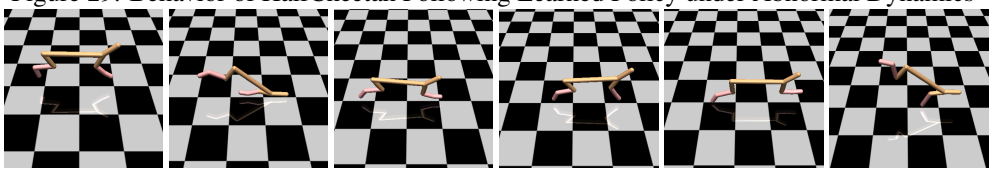

Figure 30: Behavior of HalfCheetah Following Learned Policy under Attacked Dynamics using RL

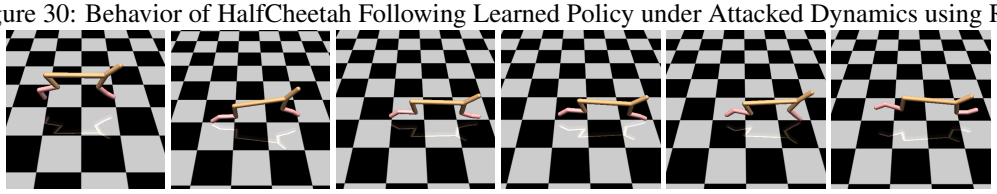

Figure 31: Behavior of HalfCheetah Following Learned Policy under Attacked Dynamics using Random Search

Figure 32: Results for Dynamics Attack on HalfCheetah

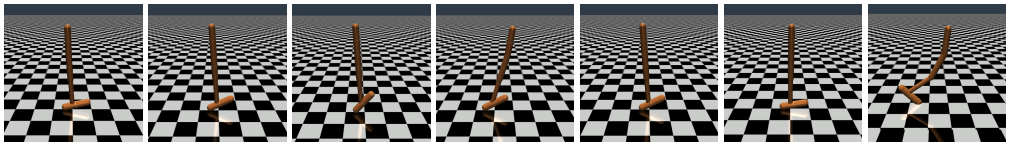

Figure 33: Behavior of Hopper Following Learned Policy under Normal Dynamics

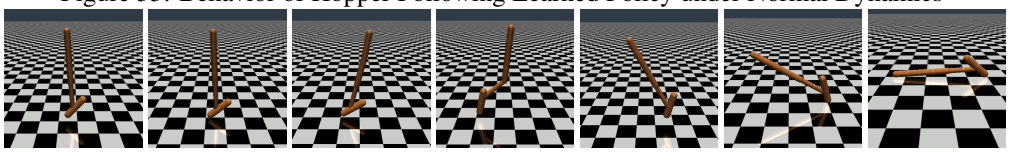

Figure 34: Behavior of Hopper Following Learned Policy under Abnormal Dynamics

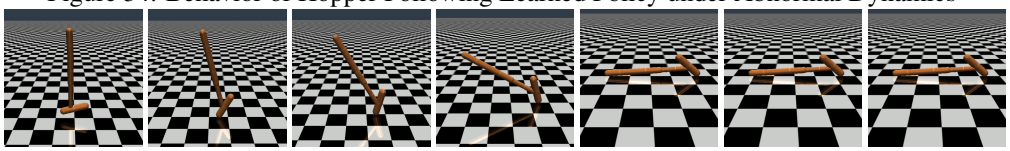

Figure 35: Behavior of Hopper Following Learned Policy under Attacked Dynamics using RL

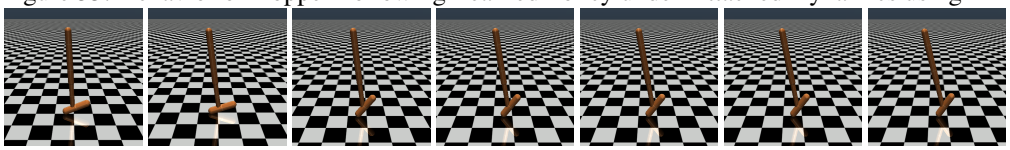

Figure 36: Behavior of Hopper Following Learned Policy under Attacked Dynamics using Random Search

Figure 37: Results for Dynamics Attack on Hopper

Figure 38: Behavior of TORCS Following Learned Policy under Normal Dynamics

Figure 39: Behavior of TORCS Following Learned Policy under Abnormal Dynamics

Figure 40: Behavior of TORCS Following Learned Policy under Attacked Dynamics using RL

Figure 41: Behavior of TORCS Following Learned Policy under Attacked Dynamics using Random Search

Figure 42: Results for Dynamics Attack on TORCS

REFERENCES

Bau, D.; Zhou, B.; Khosla, A.; Oliva, A.; and Torralba, A. 2017. Network dissection: Quantifying interpretability of deep visual representations. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 3319–3327. IEEE.

Behzadan, V., and Munir, A. 2017. Vulnerability of deep reinforcement learning to policy induction attacks. *arXiv preprint arXiv:1701.04143*.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47:253–279.

Bhagoji, A. N.; He, W.; Li, B.; and Song, D. 2017. Exploring the space of black-box attacks on deep neural networks. *arXiv preprint arXiv:1712.09491*.

Carlini, N., and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy, 2017*.

Chen, P.-Y.; Zhang, H.; Sharma, Y.; Yi, J.; and Hsieh, C.-J. 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.

Dai, X.; Li, C.-K.; and Rad, A. B. 2005. An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control. *IEEE Transactions on Intelligent Transportation Systems* 6(3):285–293.

Dhariwal, P.; Hesse, C.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; and Wu, Y. 2017. Openai baselines.

Evtimov, I.; Eykholt, K.; Fernandes, E.; Kohno, T.; Li, B.; Prakash, A.; Rahmati, A.; and Song, D. 2018. Robust physical-world attacks on machine learning models. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE.

Ghory, I. 2004. Reinforcement learning in board games. *Department of Computer Science, University of Bristol, Tech. Rep*.

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

Hannun, A.; Case, C.; Casper, J.; Catanzaro, B.; Diamos, G.; Elsen, E.; Prenger, R.; Satheesh, S.; Sengupta, S.; Coates, A.; et al. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

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

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Laskov, P., and Lippmann, R. 2010. Machine learning in adversarial environments. *Machine learning* 81(2):115–119.

Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17(1):1334–1373.

Li, B., and Vorobeychik, Y. 2014. Feature cross-substitution in adversarial classification. In *Advances in Neural Information Processing Systems*, 2087–2095.

Li, B., and Vorobeychik, Y. 2015. Scalable optimization of randomized operational decisions in adversarial classification settings. In *AISTATS*.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2016. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*.

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 *26th International Joint Conference on Artificial Intelligence*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

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.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 1928–1937.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; Fawzi, O.; and Frossard, P. 2017. Universal adversarial perturbations. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 1765–1773. IEEE.

Pan, X.; You, Y.; Wang, Z.; and Lu, C. 2017. Virtual to real reinforcement learning for autonomous driving. In *British Machine Vision Conference (BMVC)*.

Papernot, N.; McDaniel, P.; Goodfellow, I.; Jha, S.; Celik, Z. B.; and Swami, A. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 506–519. ACM.

Pinto, L.; Davidson, J.; Sukthankar, R.; and Gupta, A. 2017. Robust adversarial reinforcement learning. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, 2817–2826. PMLR.

Rusu, A. A.; Colmenarejo, S. G.; Gulcehre, C.; Desjardins, G.; Kirkpatrick, J.; Pascanu, R.; Mnih, V.; Kavukcuoglu, K.; and Hadsell, R. 2015. Policy distillation. *arXiv preprint arXiv:1511.06295*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 5026–5033. IEEE.