

Zero-One Attack: Degrading Closed-Loop Neural Network Control Systems using State-Time Perturbations

Stanley Bak
Stony Brook University
Stony Brook, NY, USA
stanley.bak@stonybrook.edu

Sergiy Bogomolov
Newcastle University
Newcastle upon Tyne, England
bogom.s@gmail.com

Abdelrahman Hekal
Newcastle University
Newcastle upon Tyne, England
b6062805@ncl.ac.uk

Veena Krish
Stony Brook University
Stony Brook, NY, USA
kveena@cs.stonybrook.edu

Andrew Mata
Stony Brook University
Stony Brook, NY, USA
andrew.mata@stonybrook.edu

Amir Rahmati
Stony Brook University
Stony Brook, NY, USA
amir@cs.stonybrook.edu

Abstract—Autonomous cyber-physical systems with deep-learning components have shown great promise but have so far enjoyed limited adoption. Part of the problem is that, beyond average-case analysis, guaranteeing robustness and reasoning about worst-case behaviors in these systems is difficult. Previous research has developed attacks that can degrade a system’s performance using small perturbations on observed states, as well as ways to retrain the networks that appear to make them robust to such attacks. In this work, we advance the state of the art by developing a new method called the Zero-One Attack, which is able to bypass the current strongest defense.

The Zero-One Attack minimizes reward by combining an outer loop zeroth-order gradient-free optimization with an inner loop, first-order gradient-based method. This setup both reduces the dimensionality of the zeroth-order optimization problem and leverages efficient gradient-based search methods for neural networks, such as projected gradient descent. In addition to state observation noise, we consider a new attack model with bounded perturbations to the execution time instant of the control policy, as real-time schedulers usually guarantee execution once per period, which may not be strictly periodic. On the Mujoco Half Cheetah system with the best current defense, the Zero-One Attack degrades the performance 195% beyond the state-of-the-art, which increases to 522% more degradation when also attacking timing jitter.

Index Terms—CPS, deep-learning, sensor noise, optimization

I. INTRODUCTION

Adding deep-learning components to cyber-physical systems (CPS) has been attracting attention throughout many industry sectors because of the ability of such systems to effectively model and optimize agent-environment interactions [1]. Machine Learning (ML) techniques such as Reinforcement Learning (RL) [2] have proven to be powerful tools for constructing effective neural network controllers in complex environments. However, such techniques are data-driven, and real-world CPS are often safety-critical.

Neural networks for visual classification systems have been shown to be vulnerable to human-eye-imperceptible adversar-

ial perturbations [3], [4]. This has raised questions about the potential for similar vulnerabilities in control systems governed by neural network policies. Recent work has demonstrated such vulnerabilities, extending adversarial attacks to deep-learning policies [5]–[7] where bounded observation noise can significantly degrade performance.

In this work, we propose a new attack that we call the *Zero-One Attack*, which casts performance degradation as an optimization problem. Our attack combines an outer loop zeroth-order gradient-free optimization with an inner loop, first-order gradient-based method. The outer loop optimizes over the system’s actions, which effectively reduces the dimensionality of the zeroth-order optimization problem as the action space is usually lower dimension than the state space. The inner loop then uses efficient first-order optimization methods for neural networks, such as projected gradient descent, in order to find state perturbations that achieve the desired actions.

Time plays a critical role in cyber-physical systems. In classical control, the choice of a discrete controller’s frequency can be theoretically justified based on the properties of the system’s transfer function and the Nyquist sampling theorem [8]. However, reinforcement learning is based on Markov decision processes where time is abstracted away during training, implicitly assuming a constant period between controller execution times. Running the system using a real-time scheduler can introduce timing jitter, and it is unclear how much this can affect an RL controller’s performance. While training with stochastic delays generally improves real-world performance [9], [10], this problem has not yet been studied from a worst-case robustness perspective. We propose a new threat model that considers allowing modifications to the controller’s execution time instant within each control period, while still subject to the periodic deadline guarantees under the standard Liu and Layland scheduling [11]. We adapt the Zero-One Attack to this threat model and generate effective timing perturbation at-

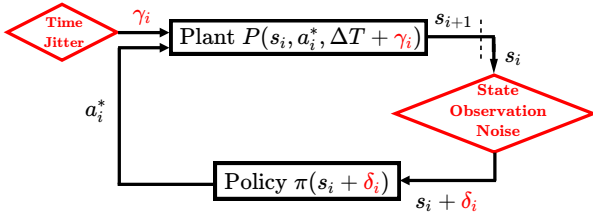


Fig. 1: The state-time threat model in this work considers bounded observation noise δ_i and constrained time jitter γ_i , at each control cycle i .

tacks. Using a combined threat model where both observation noise and time jitter are considered, results in further degraded performance. Figure 1 outlines the combined threat model.

We compare our attack with the state-of-the-art Zhang attack on nominal and adversarially-trained (defended) networks [12]. We evaluate Zero-One Attack using the same systems and networks from the prior work, a set of continuous-control RL benchmarks from the OpenAI Gym [13], as well as the SafeRL Dubins aircraft rejoin task [14] released by the Air Force Research Lab (AFRL).

The main contributions of this paper are as follows:

- We present the Zero-One Attack for attacking neural network control systems using observation perturbations. Our method is stronger than current state-of-the-art attacks even on hardened networks where no previous attacks caused significant degradation.
- We propose a new time jitter threat model. We adapt our method to generate attacks using time and/or state-time perturbations.
- We successfully generate attacks on five complex RL systems, with up to 27 state variables and 8 input actions. Our attack reduces the system’s reward several times more than the state-of-the-art, under identical perturbation bounds.

II. BACKGROUND

This work analyzes closed-loop control systems that use neural network controllers that can be learned via reinforcement learning. We provide some context about these concepts before describing our methodology. We additionally provide some context about observation perturbation techniques that typically fall within the field of Adversarial Machine Learning.

A. Neural Network Control System (NNCS)

Our work concerns closed-loop control systems, typically modeled as a Markov Decision Process (MDP) and governed by a *controller* acting in relation to a *plant*. At each time step i of the MDP, the controller determines and executes an action based on the state, and the dynamics of the environment determine the received reward and next state. We consider control networks with real-valued, multidimensional input (state) and output (control) spaces. In this work, we consider system *trajectories* that result from multiple executions of the NNCS and resulting plant dynamics within the environment. Within this context, we refer to just the sequence of states as the trajectory. Formal terminology is detailed in section III-2.

B. Reinforcement Learning

Complex NNCS policies are often learned through reinforcement learning (RL) frameworks. Control theory and RL share common concepts of a system state, an agent operating in an environment, and a policy that determines the actions of the agent. RL approaches learning optimal actions leverage the notion of a Reward R , a judgment of how well the agent has performed in the environment. Figure 1 (excluding the red diamonds) provides an illustration of the control policy and plant response in a closed-loop manner. RL algorithms seek to determine a policy that maximizes this reward. We analyze the robustness of trained policies; the development of more robust RL training algorithms is out of scope for this work. While not a requirement of our approach, we mainly consider policies trained via Proximal Policy Optimization, PPO, [15] which is considered among the state-of-the-art and has shown success across a variety of RL problems.

C. Gradient-based Adversarial Machine Learning Attacks

We leverage techniques from a vast field of work that focuses on adversarial example generation for deep neural networks. These attacks generate a slightly-altered version of a given input that would cause a target network to return an unexpected output. Adversarial evasion attacks were first shown on deep networks for image classification [3], [16]. Early techniques typically took advantage of the gradient of the network, which is used to indicate how slight changes of an input example could cause large changes in the expected output. Since the introduction of these techniques, further work has shown that adversarial examples can transfer across networks (alleviating the need to access the target network’s weights) [17], be crafted in real time [18], and can pose a threat to systems operating in the physical world [19], [20]. For our purposes, we seek to leverage a powerful attack assumes access to network gradients, which we refer to as a function GB , as we are interested in evaluating network robustness.

III. PRELIMINARIES

1) Threat Model

We investigate the performance of an RL-learned agent under the influence of an adversary that can perturb the observed state or the actuation time of the system. We look for small deviations to the observed state or actuation time that can result in significant performance decay. In Figure 1, normal closed-loop behavior involves alternating execution between a Plant that advances the state of the system and a trained Policy that determines the agent’s actions within the environment. We represent the adversary’s influence within the State Observation Noise and Time Jitter scopes in red diamonds.

The State Observation Noise scope represents cases where the state observations s_i are altered before being processed by the policy. Rather than operating on the true state of the environment, the policy returns actions based of an adversarial state, s_i^* . In this case, the adversary does not have direct control over the environment; rather, control over the observation of the environment perceived by the agent. For example, a

compromised physical sensor would relay incorrect information about the true environment to the agent, without requiring an alteration of the physical environment.

The Time Jitter scope represents cases where the adversary influences the actuation time of the system’s controls. We assume that the the learned policy executes near-instantaneously, compared with the actuation period of the system. Rather than waiting to apply the action at the next prescribed periodic time, the system might execute the control behavior at some deviation in time. Various real-world factors can affect execution time, such as computational resources (denial-of-service attacks), power management, and system load. However, we keep the constraint that exactly one action must have been actuated within each actuation time period.

For both spheres of influence, we assume that the adversary has white-box access to the weights of the trained neural network controller, which represents the strongest threat model in this context.

2) Notation

This work analyses the behavior of control systems over a total simulation time T . We refer to the state of the agent in the environment at a given time step $i \in [1, 2, ..T]$ as $s_i \in S$ where $S \subseteq \mathbb{R}^n$. The control behavior of the agent is represented by actions $a_i \in A$ where $A \subseteq \mathbb{R}^m$. The agent’s policy $\pi : \mathbb{R}^n \rightarrow \mathbb{R}^m$ executes a neural network to determine a control action given a perceived state: $a_i = \pi(s_i)$. While it is sometimes the convention to differentiate between the full state of the environment and the observed subset of that state that is fed to the policy (the "observation"), for simplicity of notation, we refer to the observed network input also as the "state". We only investigate perturbations to the portion of the state received by the agent’s policy; other state variables critical to steps of the environment are unchanged.

Given state and action pair (s_i, a_i) and actuation time period ΔT , the next state of the environment is defined using the plant dynamics P by the following: $s_{i+1} = P(s_i, a_i, \Delta T)$. This closed-loop behavior is shown in black in Figure 1 (ignoring the injected perturbations in red).

We additionally introduce the notions of adversarial perturbations and jitter that influence states of the system over time. At a given time step i , a current state s_i , and a noise bound ϵ , an adversarial observation perturbation vector δ_i is defined as $\delta_i \leq \epsilon$ itemwise. The next state of the system is determined by the action decided by the policy π on the perturbed observations, along with the plant dynamics over a given actuation period. The controller is presented with $s_i + \delta_i$, but the ground-truth environment state is unaffected. The next state is calculated by: $s_{i+1} = P(s_i, \pi(s_i + \delta_i), \Delta T)$.

Adversarial jitter γ_i is a perturbation to ΔT at time step i , within half the nominal sampling period: $\gamma_i \in [-\frac{\Delta T}{2}, \frac{\Delta T}{2}]$. The combined threat of both adversarial states and time-based jitter is represented as: $s_{i+1} = P(s_i, \pi(s_i + \delta_i), \Delta T + \gamma_i)$.

In the following sections, we denote any altered parameter with a star: *. In particular, we refer to the adversarial state

$s_i^* = s_i + \delta_i$, an adversarial action $a_i^* = \pi(s_i^*)$ and adversarial jitter $\Delta T_i^* = \Delta T + \gamma_i$.

IV. METHODOLOGY

We formulate the problem of finding the minimum-reward trajectory with maximum noise perturbation ϵ as a constrained optimization on the state variables. Using gradient methods, the search space can be transformed from the state onto the action space, which generally has a smaller dimensionality. We describe these two optimization steps of our overall approach in subsection IV-A. In the following subsection, we add the adversarial jitter as another optimization parameter to create a general adversarial attack method.

A. Zero-One Attack Approach

1) State Optimization

The task of finding noise perturbations to maximally degrade the performance of an agent over a full trace can be expressed as the following constrained optimization problem:

$$\begin{aligned} \underset{\delta_1 \dots \delta_N}{\operatorname{argmin}} \quad & \sum_{i=0}^N R(s_i, a_i) \\ \text{subject to} \quad & s_{i+1} = P(s_i, a_i, \Delta T) \\ & a_i = \pi(s_i + \delta_i) \\ & \delta_i < \epsilon \end{aligned} \tag{1}$$

where over a trajectory defined by state values over time steps $i: s_{i=1} \dots s_{i=T}$, P represents the state transitions governed by environment dynamics, π the agent’s policy, R the reward function, ϵ the attack budget, δ_i the adversarial state perturbation at time step i and s_0 the initial state. We seek to find a series of state perturbations that minimize the reward achieved by the agent over the simulation length.

While the objective function can be expressed using the environment and network controller, it is highly complex. The environment can be considered a black-box function with highly non-linear dynamics, while the neural network controller contains non-linear operations from the activation layers. This makes set-based analysis infeasible for higher-dimension environments and large networks.

An initial naive approach is to optimize the state parameters on the constrained state space around the observation. However, this becomes an infeasible problem when considering dimensionality. For example, the HalfCheetah benchmark has a $n = 17$ dimension state space and $N = 1000$ step time horizon, which yields an overall 17,000-dimensional optimization problem that is unrealistically large to effectively search over with current tools.

Dimensionality can be further reduced by splitting the total time horizon into smaller pieces to analyze separately. While this approach loses the possibility of finding the optimal solution and adds a partition hyper-parameter τ , we reduce the original optimization into several smaller optimization problems, which become more feasible to analyze. We then solve the optimization problem 2 for each partition sequentially. The

choice of τ is important as the trade-off can have immediate and long-term reduced performance depending on the size of the parameter.

Continuing our dimensionality example, the HalfCheetah environment has a $|A| = 6$ dimension action space, and a cursory hyper-parameter search reveals $\tau = 20$ as a suitable partition parameter. The optimization problem is now partitioned into 50 smaller optimization problems of dimensionality $|A| * \tau = 120$.

The choice of optimizer is important: the cost function is expensive because network gradient attacks are generated for each state in the environment loop. As a result, limiting the number of samples is advantageous. Additionally, derivatives can be ill-defined at certain samples for the cost function because of possible discontinuities caused by the non-determinism of the gradient attack as well as the constraints on the magnitude of the attack.

We choose Zeroth order optimization as it is well-suited to our problem: the complexity of our objective in Equation 2 makes it difficult to estimate gradients or devise a surrogate objective.

2) Action Optimization

Systems with large state dimensionality and searches over large time partitions τ still pose a challenge for the above optimization. This challenge is familiar to global optimization solvers, which typically require fine-tuned exploration and exploitation parameters. Figure 6 illustrates the limitations of optimization over the state space, performing worse than the state-of-art attack.

We circumvent this problem with a technique to transform the parameters of the optimization problem from the state to the action variables. The action spaces of RL environments typically have smaller dimensionality than the state space. This is a fair assumption for real-world cyber-physical systems, as the sensors in an autonomous system will likely significantly outnumber the actuators.

We leverage existing methods for computing adversarial examples on neural networks to perform the space-action transformation. These attack methods estimate a small perturbation applied to a given input that would maximize a given adversarial objective. These attacks also apply to neural network-based policies: given a policy network π , input state s_i and adversarial action a_i^* , an evasion attack will find an adversarial state s_i^* in a region close to s_i that satisfies $a_i^* = \pi(s_i^*)$.

In our application, s_i^* must be within the noise threshold ϵ defined for the environment. We define this neighborhood region $\mathbb{B}(s_i, \epsilon) := \{ |s_i - s_i^*|_\infty \leq \epsilon, \forall s_i^* \in \mathbb{B} \}$. We note that a_i^* might not be reachable for any $s_i^* \in \mathbb{B}(s_i, \epsilon)$. Thus, we make a relaxation that the gradient-based attack attempts to find the adversarial action \tilde{a}_i^* where $|a_i^* - \tilde{a}_i^*| \leq |a_i^* - a_i'|$ for any reachable action a_i' obtained by the policy from samples within $\mathbb{B}(s_i, \epsilon)$. Intuitively, we are searching for a perturbation of the given state that would yield an action closest to the target adversarial action.

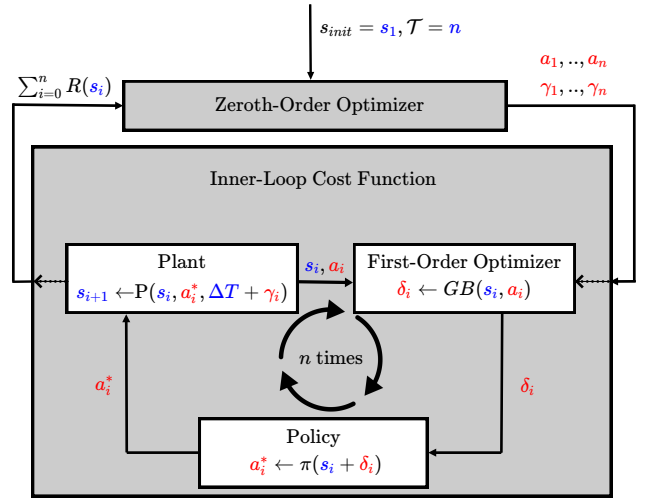


Fig. 2: Zero-One Attack in the combined threat model.

Constrained gradient attack methods have been an active research area in the field of adversarial machine learning, and several techniques exist. Our chosen implementation is detailed in Appendix B. Finding adversarial perturbations require solving a complex optimization problem where current techniques aren't guaranteed to find the optimal solution. However, they can extract gradient information so they effectively search over the state space within local neighborhoods.

The optimization problem with a new search space is framed below where τ represents the time partition parameter (number of timesteps to consider within a window), and GB is a Gradient-Based attack technique for finding adversarial states. We look for an adversarial state that minimizes the sum of rewards over all potential adversarial actions. This optimization is performed iteratively over every consecutive time horizon of length τ from $n = 0, 1, \dots, \frac{N}{\tau}$.

$$\begin{aligned} & \underset{a_1^* \dots a_\tau^*}{\operatorname{argmin}} \sum_{i=n*\tau}^{(n+1)*\tau} R(s_i, a_i^*) \\ & \text{subject to } s_{i+1} = P(s_i, a_i^*, \Delta T) \\ & a_i^* = \pi(s_i + \delta_i) \\ & \delta_i = GB(s_i, a_i) \\ & a_i \in A \end{aligned} \quad (2)$$

Algorithm 1 outlines the inner-loop cost function for a trajectory and outer-loop optimization procedure.

B. Zero-One Attack Extension to Actuation Time

The way we framed the optimization problem for observation attacks also allows us to identify other adversarial parameters that would influence the agent's trajectory. We focus on control actuation time, which has been explored with stochastic strategies but not from an explicitly adversarial perspective.

We similarly look to find a sequence of actuation periods that minimizes the total rewards achieved by the agent. The outer optimization can be expressed by the following:

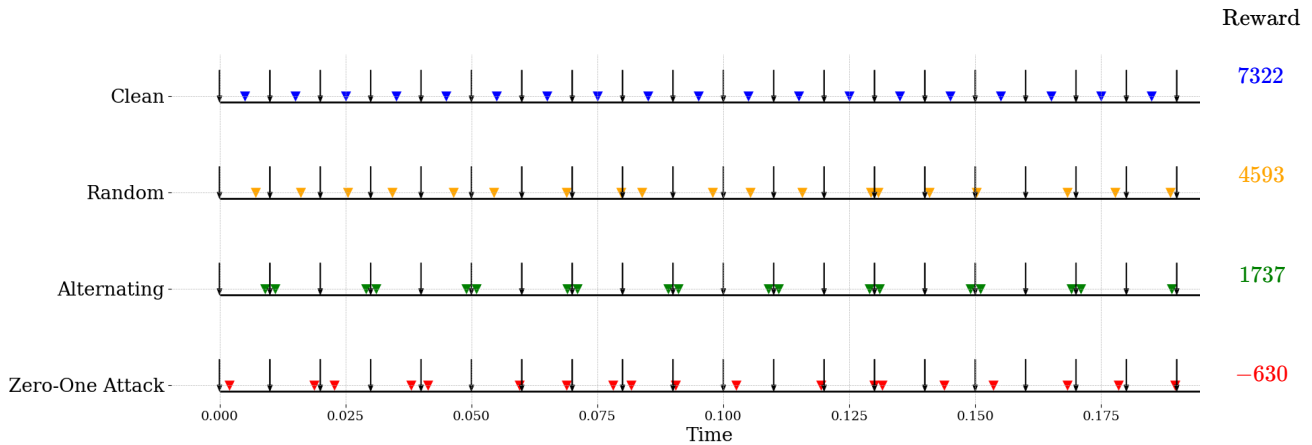


Fig. 3: Average reward achieved over simulations for one evaluated environment, Mujoco HalfCheetah, resulting from manipulations to actuation time. Each horizontal axis represents time. We show the starts/ends of each potential actuation period (black arrows) and the actual actuation times resulting from an 4 attack methods. For Clean Simulations, the true actuation time is constant periodic: the control is executed at the middle of each actuation period. For a Random attack, the control behavior is executed at a random position within each period. The Alternating Actuation attack represents an intuitive worst-case: the control behavior executes at the earliest or last available time in consecutive periods, leading to the largest gaps. Lastly, our attack, Zero-One Attack strategically identifies the worst-case actuation time per cycle to specifically degrade performance. The final average reward achieved for the Half Cheetah simulations are displayed to the right. Our Attack causes significantly more degradation than the intuitive Alternating worst-case, which is stronger than Random attacks. Full evaluations are discussed in Section V-D and Table II.

$$\begin{aligned}
 & \underset{\gamma_1^* \dots \gamma_T^*}{\operatorname{argmin}} \sum_{i=n*\tau}^{(n+1)*\tau} R(s_i, a_i) \\
 & \text{subject to } a_{i+1} = \pi(s_{i+1}) \\
 & \quad s_{i+1} = P(s_i, a_i, \Delta T + \gamma_i^*) \\
 & \quad -\frac{\Delta T}{2} \leq \gamma_i^* \leq \frac{\Delta T}{2}
 \end{aligned} \tag{3}$$

where ΔT is the actuation time period and γ_i^* is the adversarial jitter.

Figure 3 illustrates how perturbed actuation times can significantly impact the performance of an RL system. The arrows indicate the beginning and end of each actuation time window ΔT , while the marker represents the actual time an action is performed. We assume that the policy executes near-instantaneously, and that the control behavior can be executed at any point within a ΔT time frame. We impose a requirement that exactly one marker must exist in each time window. Four different actuation time patterns are illustrated: clean, random, alternating, and our method, Zero-One Attack. The clean time pattern executes at a constant time period and represents the expected behavior of the system. The Random pattern illustrates performance of the system in response to random fluctuations in actuation time, which has received most attention in related literature. The alternating pattern was chosen as an intuitive worst-case attack strategy that would yield the largest degradation, as it results in the largest possible gaps between consecutive actuations. The last row illustrates our adversarial attack: the precise actuation times are strategically computed for each period to degrade performance. We display total reward achieved by one of our evaluations for these time

patterns to the right to indicate how we measure the success of the attack.

V. EVALUATION

The current state-of-the-art sensor noise attack generation is by Zhang *et al.* [12], which we denote as Zhang’s attack. This attack trains a neural network model that outputs the adversarial noise to add at each trajectory state, significantly outperforming all other attack generations before it. In our evaluation, we compare Zero-One Attack to theirs under the same conditions. We encourage readers to refer to their work for a comprehensive comparison between Zhang’s attack and attack methods before it.

Additionally, we compare Zero-One Attack against their nominal PPO network controllers as well as adversarially-trained (ATLA) controllers on which no current adversarial attack generator tool has been successful in significantly degrading performance. In particular, we demonstrate our attack performance on the following policies for each environment: (1) Nominal PPO, (2) Defended PPO using ATLA adversarial training, and (3) Defended PPO using an ATLA-retrained network with an LSTM (Long Short-Term Memory) component and State-Adversarial regularization from their earlier work [21]. The ATLA-LSTM + SA Regularization is their most robust model.

A. Environments

We evaluate our approach on an array of continuous-control tasks. We selected four standard Mujoco environments from the Open AI Gym [13] to directly compare our method with Zhang *et al.*. We additionally include an example of a continuous-control aerospace task from the SafeRL suite of benchmarks to illustrate our method’s performance within a safety-critical typical control system setting. We use the 2D

Algorithm 1 Zero-One Attack

```
1: Initialize:  $s_{init} \leftarrow s_{start}$ 
2: for each time partition window do
3:    $\triangleright$  Solve for each time window sequentially using 0th
   order optimization
4:    $\{a_1^*, \dots, a_{\mathcal{T}}^*\} \leftarrow \operatorname{argmin}_A \text{ZERO-ORDERCOST}(s_{init})$ 
5:    $s_{end} \leftarrow \text{SIM-ENV}(s_{init}, a_1^*, \dots, a_{\mathcal{T}}^*)$ 
6:    $s_{init} \leftarrow s_{end}$ 
7: end for
8:
9: function SIM-ENV( $s_1, a_1, \dots, a_n, P, \pi$ )
10:  for 1..n do
11:     $a_i \leftarrow \pi(s_i)$ 
12:     $s_{i+1}, R_{i+1} \leftarrow P(s_i, a_i, \Delta T)$ 
13:  end for
14:  return  $s_n$ 
15: end function
16:
17: function ZERO-ORDERCOST( $a_1, \dots, a_{\mathcal{T}} \mid s_{i=1}, P, \pi, \epsilon$ )
18:   $\triangleright$  Inner optimization on selected actions  $a_i$ , initial state
  in time partition,  $s_{i=1}$ , state dynamics  $P$ , policy  $\pi$ , and
  noise bound  $\epsilon$  using 1st order technique
19:  Initialize: GB attack on  $\pi$  with constraint  $\epsilon$ 
20:  for  $t = 1 \dots \tau$  do
21:     $\delta_i \leftarrow \text{GB}(s_i, a_i)$ 
22:     $a_i^* \leftarrow \pi(s_i + \delta_i)$   $\triangleright$  Adversarial action
23:     $s_{i+1}, R_{i+1} \leftarrow P(s_i, a_i^*, \Delta T)$ 
24:  end for
25:  return  $\sum_{i=1}^{\mathcal{T}} R_i$   $\triangleright$  Total reward for attacked trajectory
26: end function
```

version of the Dubins Rejoin task from the SafeRL suite, representing two aircrafts in a coplanar flight. The wingman aircraft is considered the agent, and the trainable policy should guide the wingman into formation flight around the lead during the simulation without colliding. Illustrations of these five environments are shown in Table IV in Appendix A, along with their descriptions.

B. Optimization Tools

We used the ZOOpt library for the outer optimization and Projected Gradient Descent for the inner optimization for all evaluations. Details of these techniques are provided in Appendix B.

C. Observation Noise Evaluation

We evaluate our attack method on four continuous RL control tasks from the OpenAI Gym [13] that use the Mujoco engine [22] for physical dynamics.

These results are presented in Table I. Additionally, we report the clean reward for each network as a reference. There is an unusual decrease in performance between the Nominal and Defended (ATLA) model performance under Zhang’s attack, which is unintuitive. We believe that this is a side effect of the inconsistent nature of the attack due to the use of function



Fig. 4: State of HalfCheetah agent in clean and attacked simulations. We observed that the clean policy leads the agent to run steadily forwards (right), past the end of the stage. The policy attacked using Zhang’s attack causes the cheetah to slowly inch backwards. In contrast, the policy attacked using Zero-One Attack leads the cheetah to run backwards, beyond the start of the stage.

approximators.

Our method outperforms Zhang’s attack in every benchmark and network except for the PPO model for the Hopper environment, which it minimally underperforms. Interestingly, the ATLA retrained networks, while resistant to Zhang’s attack, were less robust than the PPO model to our attack method. This observation is especially evident in the HalfCheetah and Walker2D environments. While the LSTM models were still more robust across the board, our method was shown to degrade performance by an additional 54% to 78% compared to the state-of-the-art. Hopper and Walker2D both have safety conditions that cause the simulation to end early when violated, which we achieve for the LSTM models that Zhang’s attack cannot find. While Half Cheetah and Ant do not have safety conditions, we find rewards that are, on average, about 2400 less.

Interpretations of performance degradation. Our method is guided by total rewards, which is a heuristic not only for how well the agent achieves a goal but also for the overall safety and stability of the system. However, the reward function can be replaced by any metric, including STL robustness metric that represent the safety of the system. We explore this possibility further in the Dubins Rejoin case study in section V-F. While we do not embed explicit safety criteria within our adversarial objectives, we notice that the attacked trajectories of the Mujoco robotic trials behave in an intuitively counterproductive and unsafe manner. Figure 4 shows the final state of a HalfCheetah simulation for clean and attacked (Zhang’s and Zero-One Attack) policies. The cheetah runs to the right of the stage (*i.e.*, forward) when the policy is not adversarially attacked. Zhang’s attack leads it to collapse and inch backward slowly, while our attack leads the cheetah to run backward.

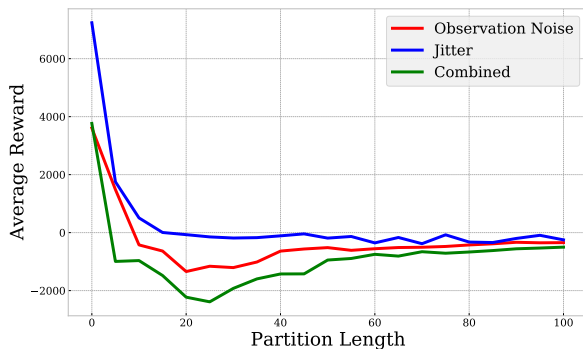
D. Time Jitter Evaluation

We evaluate our jitter optimization method on the same Mujoco environments and report the results in Table II. For performance comparison, we evaluate two other jitter patterns, random and alternating. The results show that certain environments are more vulnerable to jitter than others. For example, a random attack on Hopper and Walker degrades performance (as a percentage of total reward) significantly more than in Half Cheetah and Ant.

In one attack instance, the Ant PPO model, our optimized

TABLE I: Average Reward \pm Standard Deviation over 10 different starting states under sensor attacks.

Environment	Attack Strength δ	Network	Clean Reward	Zhang’s Attack [12]	Zero-One Attack
Half Cheetah	0.15	Nominal	7234 \pm 112	-530 \pm 220	-1337 \pm 284
		Defended (ATLA)	5644 \pm 42	3129 \pm 79	-1707 \pm 247
		Defended (ATLA:LSTM + SA reg)	6675 \pm 167	5466 \pm 194	3098 \pm 377
Hopper	0.075	Nominal	3389 \pm 6	638 \pm 3	763 \pm 303
		Defended (ATLA)	2493 \pm 1085	1001 \pm 30	684 \pm 96
		Defended (ATLA:LSTM + SA reg)	3374 \pm 533	2077 \pm 865	1212 \pm 236
Walker2D	0.05	Nominal	4282 \pm 1054	1079 \pm 112	743 \pm 428
		Defended (ATLA)	3555 \pm 965	2206 \pm 815	293 \pm 118
		Defended (ATLA:LSTM + SA reg)	4022 \pm 44	3940 \pm 74	1246 \pm 110
Ant	0.15	Nominal	5877 \pm 152	161 \pm 34	-407 \pm 185
		Defended (ATLA)	4640 \pm 47	-195 \pm 606	-406 \pm 217
		Defended (ATLA:LSTM + SA reg)	5306 \pm 168	3560 \pm 93	1177 \pm 198

**Fig. 5:** HalfCheetah Average Reward under Zero-One Attack, on 10 different starting states, over varying time partitions τ . A clear performance valley is present for $\tau = 20 - 25$, which is the sweet spot between exploitation and exploration for our method in this environment.

approach performs worse, but similar, to the alternating attack pattern. This is an example where our approach doesn’t necessarily find the global optima, in part due to the highly non-linear nature of the search problem. There also appears to be no correlation between improved performance to jitter attacks and any network defense, which is intuitive since the defense is designed specifically against observation attacks.

Overall, our optimization approach outperforms the baseline Random and Alternating methods, showcasing that jitter added in particular ways can have a substantial influence on the agent’s performance. This is best shown in Hopper and Walker, where optimizing jitter results in almost immediate failure, causing the reward to be close to or less than 0. Notably, these results challenge the intuitive assumption that an Alternating pattern results in worst-case behavior. Our attack illustrates that there exist vulnerabilities in policies trained under MDP assumptions – slight deviations to the actuation time can degrade the agent’s performance in extreme ways.

E. State-Time Noise Evaluation

Additionally, we show that combining observation and time attacks can further degrade model performance. At each time-step, we identify adversarial perturbations of both δ and γ .

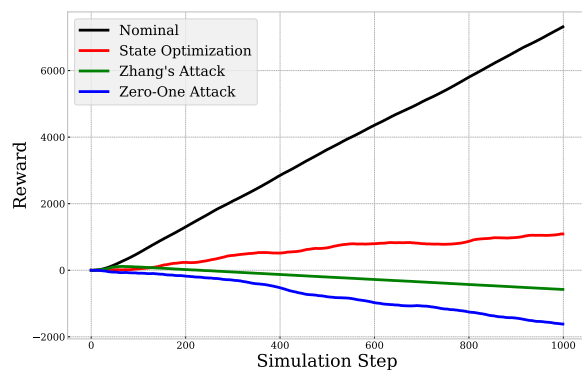
**Fig. 6:** HalfCheetah Adversarial Traces that show the reward achieved by various agents over the total simulation time. The Nominal agent is not attacked, Zhang’s Attack agent is attacked using the Zhang *et al.* method, and the remaining two agents are attacked using Zero-One Attack. State Optimization uses an attack that only uses the outer optimization detailed in Section IV-A1. The full Zero-One Attack attack that leverages both optimization techniques. In both methods, we empirically search for and set the best partition parameter τ .

Table III displays reward achieved by variants of the Zero-One Attack: for state observation attacks only (column 4), for actuation jitter only (column 5), and both combined (column 6). The combined approach is generally extremely powerful: compared with the clean reward, we observe degradation over 100%. In a few cases, the combined approach is only comparable with the Observation-only or Jitter-only variants, which we attribute to randomness over optimization parameters and the challenge of the increased search dimension within the combined optimization problem.

F. Safe-RL Case Study

We implement the Zero-One Attack on the DubinsRejoin flight formation task from the SafeRL benchmark suite developed by the Air Force Research Lab (AFRL) [14]. This demonstrates the generalizability of our attack beyond traditional RL benchmarks; moreover, we illustrate how the attack can be used to falsify a specification that is different than the reward used to train the system. The networks were trained using a four-part reward function that aims to navigate the fol-

TABLE II: Actuation time jitter attack results. Shown are the average ending reward \pm standard deviation of simulations for each environment over 10 different starting states given various actuation time schedules. The Clean Reward column displays the reward for controllers actuating on the default periodic schedule.

Environment	Network	Clean Reward	Random	Alternating	Zero-One Attack
Half Cheetah	Nominal	7234 \pm 112	4014 \pm 627	1340 \pm 632	-264 \pm 229
	Defended (ATLA)	5644 \pm 42	4305 \pm 1259	1613 \pm 1149	-166 \pm 509
	Defended (ATLA:LSTM + SA reg)	6675 \pm 167	2957 \pm 1519	1678 \pm 953	-69 \pm 168
Hopper	Nominal	3389 \pm 6	1061 \pm 11	1024 \pm 25	8 \pm 1
	Defended (ATLA)	2493 \pm 1085	1062 \pm 27	1119 \pm 45	5 \pm 1
	Defended (ATLA:LSTM + SA reg)	3374 \pm 533	1064 \pm 89	1113 \pm 59	4 \pm 0
Walker2D	Nominal	4282 \pm 1054	1165 \pm 195	953 \pm 42	-30 \pm 14
	Defended (ATLA)	3555 \pm 965	956 \pm 68	525 \pm 426	-46 \pm 36
	Defended (ATLA:LSTM + SA reg)	4022 \pm 44	1025 \pm 46	1009 \pm 45	1 \pm 3
Ant	Nominal	5877 \pm 152	2088 \pm 1250	1391 \pm 498	1494 \pm 436
	Defended (ATLA)	4640 \pm 47	4340 \pm 396	2902 \pm 1088	2454 \pm 515
	Defended (ATLA:LSTM + SA reg)	5306 \pm 168	4016 \pm 1035	2546 \pm 1118	1416 \pm 206

TABLE III: Zero-One Attack under all three threat models, observation noise, jitter, and both. Shown are the average rewards \pm standard deviation over 10 different starting states. In parenthesis, we include the percentage degradation achieved with respect to the average clean reward.

Environment	Network	Clean Reward	Zero-One Attack:		
			Observation Perturbations Only	Zero-One Attack: Time Jitter Only	Zero-One Attack: Combined
Half Cheetah	Nominal	7234 \pm 112	-1337 \pm 284 (118%)	-264 \pm 229 (104%)	-2412 \pm 300 (133%)
	Defended (ATLA)	5644 \pm 42	-1707 \pm 247 (130%)	-166 \pm 509 (103%)	-3737 \pm 312 (166%)
	Defended (ATLA:LSTM + SA reg)	6675 \pm 167	3098 \pm 377 (54%)	-69 \pm 168 (101%)	-840 \pm 312 (113%)
Hopper	Nominal	3389 \pm 6	763 \pm 303 (77%)	8 \pm 1 (100%)	7 \pm 1 (100%)
	Defended (ATLA)	2493 \pm 1085	684 \pm 96 (73%)	5 \pm 1 (100%)	4 \pm 0 (100%)
	Defended (ATLA:LSTM + SA reg)	3374 \pm 533	1212 \pm 236 (64%)	4 \pm 0 (100%)	4 \pm 0 (100%)
Walker2D	Nominal	4282 \pm 1054	743 \pm 428 (83%)	-30 \pm 14 (101%)	-8 \pm 6 (100%)
	Defended (ATLA)	3555 \pm 965	293 \pm 118 (92%)	-46 \pm 36 (101%)	-32 \pm 12 (101%)
	Defended (ATLA:LSTM + SA reg)	4022 \pm 44	1246 \pm 110 (69%)	1 \pm 3 (100%)	3 \pm 3 (100%)
Ant	Nominal	5877 \pm 152	-407 \pm 185 (107%)	1494 \pm 436 (75%)	-417 \pm 91 (107%)
	Defended (ATLA)	4640 \pm 47	-406 \pm 217 (109%)	2454 \pm 515 (47%)	-392 \pm 58 (108%)
	Defended (ATLA:LSTM + SA reg)	5306 \pm 168	1177 \pm 198 (78%)	1416 \pm 206 (73%)	695 \pm 179 (87%)

lower aircraft into formation at a specified distance behind the lead without colliding. For our case study, we use the minimize the distance between the aircraft in lieu of total reward, to represent an adversarial objective intended to cause a collision.

Figure 7 shows an example of an adversarial trajectory found by our approach using state perturbations with attack strength $\epsilon = 0.3$. Within the 2D state space, the lead and wingman aircrafts are flying independently. The wingman’s controller should guide it into formation flight along the route of the lead, without getting too close and colliding. For the nominal controller, the wingman (blue) closely follows the lead (black) at a safe distance. The Nominal controller is well-trained: a collision was extremely rare. The optimized adversarial observation perturbations from Zero-One Attack can be seen to guide the wingman into a collision state (red). We were able to realize collisions using our attack approach for all attempted trials, indicating that the approach effectively searches the observation state space.

Actuation-time perturbations alone were unable to cause collisions, revealing that the plant and controller were robust to adversarial jitter. However, we found that in the combined threat model, Zero-One Attack finds unsafe trajectories with attack strength $\epsilon = 0.15$, half the attack strength than needed in the state observation noise threat model alone.

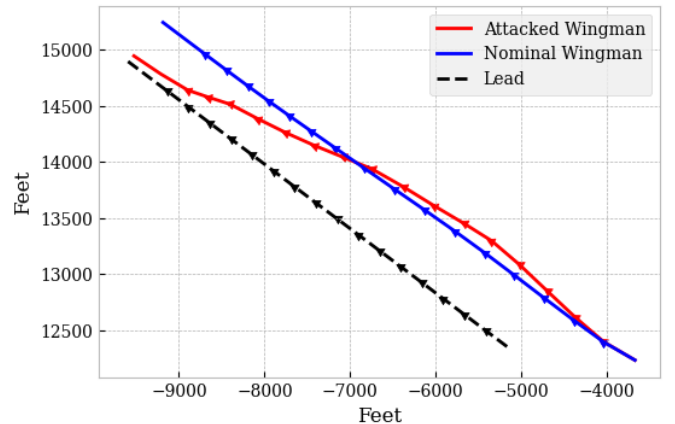


Fig. 7: Example trajectories of lead and wingman aircrafts ahead of a collision caused by observation perturbations within the 2D state space. All trajectories shown represent aircraft positions over time, starting at the bottom right of the figure and concluding at the top left (direction of arrows). Without the manipulated observations, the wingman (blue) is controlled by the neural network policy to fly steady in formation at a specified radius behind the lead (black). Zero-One Attack finds perturbations at each timestep that would instead lead the wingman to collide with the lead (red).

VI. RELATED WORK

Several works have approached the problem of robust reinforcement learning systems from various perspectives. One line of work looks at training agents that can recover well from a changing environment. Robust Markov decision processes (RMDP) [23] facilitate RL modeling in the presence of stochastic physical environment variables (*i.e.*, varying gravity, mass, friction, etc..). Other frameworks address a similar problem by training an agent within two-player, minimax optimization [24]–[26]. These works seek to answer a challenging but different problem from ours: we investigate robustness to misread state observations (through adversarial attacks or faulty sensors) and do not assume an adversary can arbitrarily manipulate the underlying true environment.

A more relevant line of work has focused on developing observational attacks on RL agents, wherein the underlying true environment during the attack is unchanged. Behzadan and Munir [7], Huang et al [5], and Kos and Song [6] all showed definitively in 2017 that modern deep RL policies for Atari games are vulnerable to standard gradient-based adversarial machine learning techniques. Improvements on this technique sought to incorporate some form of forecasting, to reduce the number of attacked time steps and illustrate agents may be vulnerable to single points of failure [27], [28].

The strongest attack to-date by Zhang [12] introduces the state-adversarial Markov decision process (SA-MDP) [21] to facilitate the framework of an “optimal” adversary for perturbing observations within a defined bounded set to diminish rewards, akin to the notion of minimal adversarial perturbations when minimizing the bounded set. They propose a method for training an approximation of this optimal adversary, and they show that robust agent policies can be obtained by using this adversary during retraining. Their retrained, *ATLA*, PPO models perform well in response to their trained adversary. Our work adopts and builds upon this threat model for observation noise perturbations, using their proposed RL benchmarks and networks, to advance the state-of-art.

In parallel, the formal methods community has taken multiple approaches to verification of these systems [29], [30]. Typical solutions aim to find tight bounds on the uncertainty around given examples that would alter a network’s output [31]–[33], but solutions to the problem have been limited to low-dimensional systems with relatively few time-steps. Ongoing challenges stem from non-linear dynamics and exponential blow-up from set splitting on non-linear activation functions. In contrast, RL benchmarks generally contain significantly more difficult control problems than current tools can verify [34]. Recent work has attempted to perform closed-loop verification on systems with sensor noise using these tools [35], [36], but use is limited.

Considering actuation time, a line of work has focused on improving the robustness of control systems to delayed environments. Most of this work has focused on domain adaptation: *i.e.*, ensuring that a model trained in simulation can perform well in the physical world. Previous works have used do-

main randomization techniques to address transferring policies across domains that may have different state transition delays. [9], [37] The closest works to ours have specifically sought to estimate expected behavior of RL systems in response to action delays. Sim2Real incorporates sampled execution delays into the training process to develop RL policies robust to delays within the entire closed-loop process. [10] Secondly, Chen *et al.* reformulated a standard markov decision process to incorporate multi-step time delays. To our knowledge, none have addressed actuation-time jitter from an adversarial perspective, rather than random jitter.

VII. CONCLUSION

In this work, we present an adversarial attack generation method called the Zero-One Attack. Our attack targets neural network control systems and degrades performance by adding bounded observation and/or timing noise. Physical world deployments of NNCS will be subject to sensor noise and timing jitter arising from the real-time scheduler, and control theoretic means to evaluate robustness are not applicable to this class of systems. Thus, until formal verification methods for NNCS can scale to handle complex networks and systems as well as provide global robustness guarantees, the most practical approach is to run strong attacks such as ours in order to evaluate the possible performance degradation from such noise.

We evaluated the Zero-One Attack on adversarially-defended networks and achieved significant performance degradation beyond the state-of-the-art. Further, we introduced the bounded time perturbation threat model and adapted the Zero-One Attack to find adversarial traces using only time or both state-time perturbations. With both state and time perturbations, we showed the attack strength gets amplified—noise within smaller bounds can degrade performance significantly.

Future work is needed to adapt the method for defense—to train more robust models. Particularly, the current runtime of the approach is large making it impractical for use within an adversarial retraining scheme, where often thousands of attacked traces are needed.

VIII. ACKNOWLEDGEMENTS

This material is based upon work supported by the Air Force Office of Scientific Research and the Office of Naval Research under award numbers FA9550-19-1-0288, FA9550-21-1-0121, FA9550-23-1-0066 and N00014-22-1-2156, and the National Science Foundation under Award No. 2237229. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force or the United States Navy.

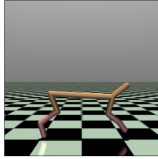
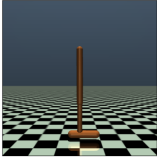
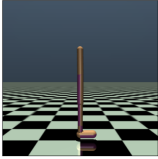
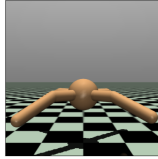

REFERENCES

- [1] R. Baheti and H. Gill, “Cyber-physical systems,” *The impact of control technology*, vol. 12, no. 1, pp. 161–166, 2011.
- [2] Y. Li, “Deep reinforcement learning: An overview,” 2018.
- [3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” in *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

- [4] B. Luo, Y. Liu, L. Wei, and Q. Xu, "Towards imperceptible and robust adversarial example attacks against neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.
- [5] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial Attacks on Neural Network Policies," *arXiv:1702.02284 [cs, stat]*, Feb. 2017, arXiv: 1702.02284. [Online]. Available: <http://arxiv.org/abs/1702.02284>
- [6] J. Kos and D. Song, "Delving into adversarial attacks on deep policies," *arXiv:1705.06452 [cs, stat]*, May 2017, arXiv: 1705.06452. [Online]. Available: <http://arxiv.org/abs/1705.06452>
- [7] V. Behzadan and A. Munir, "Vulnerability of deep reinforcement learning to policy induction attacks," in *Machine Learning and Data Mining in Pattern Recognition*, P. Perner, Ed. Cham: Springer International Publishing, 2017, pp. 262–275.
- [8] G. F. Franklin, J. D. Powell, A. Emami-Naeini, and J. D. Powell, *Feedback control of dynamic systems*. Prentice hall Upper Saddle River, 2002, vol. 4.
- [9] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.
- [10] S. S. Sandha, L. Garcia, B. Balaji, F. Anwar, and M. Srivastava, "Sim2real transfer for deep reinforcement learning with stochastic state transition delays," in *Conference on Robot Learning*. PMLR, 2021, pp. 1066–1083.
- [11] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.
- [12] H. Zhang, H. Chen, D. Boning, and C.-J. Hsieh, "Robust reinforcement learning on state observations with learned optimal adversary," 2021.
- [13] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.
- [14] U. J. Ravaioli, J. Cunningham, J. McCarroll, V. Gangal, K. Dunlap, and K. L. Hobbs, "Safe reinforcement learning benchmark environments for aerospace control systems," in *2022 IEEE Aerospace Conference (AERO)*. IEEE, 2022, pp. 1–20.
- [15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
- [16] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *stat*, vol. 1050, p. 20, 2015.
- [17] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *International Conference on Learning Representations*, 2016.
- [18] Y. Gong, B. Li, C. Poellabauer, and Y. Shi, "Real-time adversarial attacks," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 4672–4680.
- [19] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song, "Robust physical-world attacks on machine learning models," *arXiv preprint arXiv:1707.08945*, vol. 2, no. 3, p. 4, 2017.
- [20] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112.
- [21] H. Zhang, H. Chen, C. Xiao, B. Li, M. Liu, D. Boning, and C.-J. Hsieh, "Robust deep reinforcement learning against adversarial perturbations on state observations," 2021.
- [22] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.
- [23] V. Goyal and J. Grand-Clement, "Robust markov decision process: Beyond rectangularity," 2021.
- [24] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," 2017.
- [25] A. Gu, S. Lu, P. Ram, and L. Weng, "Min-max bilevel multi-objective optimization with applications in machine learning," 2023.
- [26] P. Czempin and A. Gleave, "Reducing exploitability with population based training," 2023.
- [27] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, "Tactics of Adversarial Attack on Deep Reinforcement Learning Agents," *arXiv:1703.06748 [cs, stat]*, Nov. 2019, arXiv: 1703.06748. [Online]. Available: <http://arxiv.org/abs/1703.06748>
- [28] J. Sun, T. Zhang, X. Xie, L. Ma, Y. Zheng, K. Chen, and Y. Liu, "Stealthy and Efficient Adversarial Attacks against Deep Reinforcement Learning," *arXiv:2005.07099 [cs]*, May 2020, arXiv: 2005.07099. [Online]. Available: <http://arxiv.org/abs/2005.07099>
- [29] T. T. Johnson, D. Manzananas Lopez, L. Benet, M. Forests, S. Guadalupe, C. Schilling, R. Ivanov, T. J. Carpenter, J. Weimer, and I. Lee, "Arch-comp21 category report: artificial intelligence and neural network control systems (ainncs) for continuous and hybrid systems plants," *EPiC Series in Computing*, vol. 80, 2021.
- [30] C. Brix, M. N. Müller, S. Bak, T. T. Johnson, and C. Liu, "First three years of the international verification of neural networks competition (vnn-comp)," *International Journal on Software Tools for Technology Transfer*, pp. 1–11, 2023.
- [31] H.-D. Tran, X. Yang, D. Manzananas Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, and T. T. Johnson, "Nnv: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems," in *International Conference on Computer Aided Verification*. Springer, 2020, pp. 3–17.
- [32] R. Ivanov, T. Carpenter, J. Weimer, R. Alur, G. Pappas, and I. Lee, "Verisig 2.0: Verification of neural network controllers using taylor model preconditioning," in *International Conference on Computer Aided Verification*. Springer, 2021, pp. 249–262.
- [33] A. Abate, D. Ahmed, A. Edwards, M. Giacobbe, and A. Peruffo, "Fossil: A software tool for the formal synthesis of lyapunov functions and barrier certificates using neural networks," in *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: <https://doi.org/10.1145/3447928.3456646>
- [34] M. Everett, "Neural network verification in control," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 6326–6340.
- [35] V. Krish, A. Mata, S. Bak, and A. Rahmati, "Provable observation noise robustness for neural network control systems," *Research Directions: Cyber-Physical Systems*, vol. 1, p. e2, 2023.
- [36] S. Bak, "nmenu: Verification of relu neural networks with optimized abstraction refinement," in *NASA Formal Methods Symposium*. Springer, 2021, pp. 19–36.
- [37] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.
- [38] Y.-R. Liu, Y.-Q. Hu, H. Qian, C. Qian, and Y. Yu, "ZOOpt: a toolbox for derivative-free optimization," *Science China Information Sciences*, vol. 65, no. 10, Sep 2022. [Online]. Available: <https://doi.org/10.1007%2Fs11432-021-3416-y>
- [39] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.
- [40] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *International conference on machine learning*. PMLR, 2020, pp. 2206–2216.
- [41] —, "Minimally distorted adversarial examples with a fast adaptive boundary attack," in *International Conference on Machine Learning*. PMLR, 2020, pp. 2196–2205.
- [42] G. W. Ding, L. Wang, and X. Jin, "AdverTorch v0.1: An adversarial robustness toolbox based on pytorch," *arXiv preprint arXiv:1902.07623*, 2019.
- [43] Y.-Q. Hu, H. Qian, and Y. Yu, "Sequential classification-based optimization for direct policy search," 02 2017.
- [44] Y.-R. Liu, Y.-Q. Hu, H. Qian, and Y. Yu, "Asynchronous classification-based optimization," in *Proceedings of the First International Conference on Distributed Artificial Intelligence*, ser. DAI '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: <https://doi.org/10.1145/3356464.3357709>

APPENDIX A ENVIRONMENT DETAILS

TABLE IV: Details of each environment with depictions. For all environments, the agents are required to achieve a particular goal immediately (move forward or fly towards a target region) and are further rewarded for continuing to achieve this task over time. For the Mujoco environments, the agents are penalized for moving backward (left), falling down, and making high-magnitude, variable actions. The DubinsRejoin agent is penalized for not reaching the rejoin radius within the simulation length or colliding with the lead aircraft.

	HalfCheetah	Hopper	Walker	Ant	DubinsRejoin
					
Task	Apply a torque on joints to make the cheetah run forward (right)	Apply torque on hinges to make hops forward (right)	Apply torque to feet, legs, and thighs in coordination to walk forward (right)	Apply torque on joints in coordination to crawl forward (right)	Control throttle and rudder actuators to fly wingman aircraft at a specified radius around lead
Reward Function	Distance moved forward, with large actions penalized.	Distance moved forward, with large actions penalized. Episode ends when agent falls over.	Distance moved forward, with large actions penalized. Episode ends when agent falls over.	Distance moved forward, with large actions penalized	Time spent in rejoin region, penalized for nearing too close to lead
State Dimension	17	11	17	27	8
Action Dimension	6	3	6	8	2

APPENDIX B IMPLEMENTATION DETAILS

For our evaluations, we use the ZOOpt [38] library for the outer optimization, Projected Gradient Descent for the inner gradient-based optimization, and chose time partition parameter τ empirically for each environment. While any pair of optimization methods could be used for the zeroth-order outer optimization and the gradient-based inner optimization, we found that ZOOpt and PGD performed well across all environments. For the Mujoco environments, we use the trained policies provided by Zhang *et al.*^a for all three networks (Nominal and two ATLA-defended). For the DubinsRejoin environment, we use the SafeRL codebase^b to train a Nominal PPO with the provided default parameters (*rect* norm).

To avoid variance in training, we use Zhang *et al.*'s saved environments for environment set-up, as well as their network architecture, weights, and filters. For consistent attack strength, we use their maximum state perturbation parameter ϵ for each environment, where $s^* \in \mathbb{B}(s, \epsilon)$ which is the l_∞ norm ball around state s (all states are normalized before network execution). We evaluate both attacks on the three network types over ten different starting states and report the average reward and standard deviation.

A. Projected Gradient Descent

Projected Gradient Descent (PGD) [39] is considered a universal first-order technique for finding adversarial examples (*i.e.*, it represents the strongest attack utilizing local first-order in-

formation about a network). PGD is formulated as an iterative optimization process. Given an initial sample, a target action, and perturbation limit ϵ , the sample is iteratively updated in a direction that maximizes an adversarial loss, which is a function of the network's weights and target action. The gradient of this adversarial loss is used to estimate the direction of the update. Each update is followed by a projection back onto the l_{inf} ball (by clipping the values of the adversarial example to $\pm\epsilon$). We tested a handful of commonly used methods (FGSM [16], AutoAttack [40], FABattack [41]) but empirically found that PGD performed well and fast enough to use as our inner optimization method. We use the AdverTorch implementation of PGD [42], which includes initial randomization.

B. State Optimization

The ZOOpt [38] library is a toolbox for zeroth-order optimization suited for high-dimensional problems. Their main optimization technique is from work from Hu *et al.* [43], which uses a classification-based optimizer and proves effective for problems with many local optima. Additionally, an asynchronous version exists [44], allowing for parallelization, which allows the cost function to scale favorably when multiple gradient attacks can be done simultaneously. Other zero-order optimization techniques could have been used; particularly, we tested Bayesian Optimization but found that ZOOpt performed well and didn't require hyperparameter tuning.

^ahttps://github.com/huanzhang12/ATLA_robust_RL

^b<https://github.com/act3-ace/SafeRL>