Should Importance of an Attack's Future be Determined by its Current Success?

Ridhima Bector
Nanyang Technological University
Singapore, 639798
ridhima001@e.ntu.edu.sg

Hang Xu
Nanyang Technological University
Singapore, 639798
hang017@e.ntu.edu.sg

Abhay Aradhya
Nanyang Technological University
Singapore, 639798
abhayaradhya@ntu.edu.sg

Chai Quek
Nanyang Technological University
Singapore, 639798
ashcquek@ntu.edu.sg

Zinovi Rabinovich
Nanyang Technological University
Singapore, 639798
zinovi@ntu.edu.sg

ABSTRACT

Safety and robustness of Reinforcement learning (RL) solutions have become increasingly crucial as RL is being deployed in safety-critical industries such as construction, aviation, autonomous driving, etc. Successful RL deployment is endangered by an array of attacks, of which the most insidious are training-time attacks (TTAs), due to their capability of inculcating behavioral loopholes into an RL strategy. Several works develop and study *constructive* TTAs, where the attacker forces a specific, target behavior upon a training RL agent (victim). Herein the target behavior is un-adoptable under the original dynamics of the environment and hence the attacker learns a strategy to appropriately alter the dynamics of the victim's environment. In contrast to previous works, we study target behaviors that are un-adoptable in the default environment due to *both* environment dynamics *as well as* sub-optimality with respect to the victim's objective(s). To find efficient attacks in this context, we develop a novel reinforcement-learning algorithm, γDDPG, that dynamically alters the attack policy planning horizon based on the victim's current behavior. This improves effort distribution across the attack timeline and reduces the effect of uncertainty in the blackbox setting. To demonstrate the features of our method and better relate the results to prior research, we borrow a 3D Grid World domain from the latter for our experiments.

KEYWORDS

Training-Time Attack, Dynamic Discount, Deep Reinforcement Learning

1 INTRODUCTION

Over the last decade, Reinforcement Learning (RL) has dramatically altered the decision-making research landscape and produced several AI breakthroughs [3, 26, 29]. In turn, deployment of RL solutions in safety-critical domains necessitates research on their safety and robustness. Success of RL stands threatened by a wide variety of attacks, most insidious of which are training-time attacks that "pre-program" back-doors and behavioral triggers into an RL strategy. In training-time attacks, the attacker learns to optimally modify/poison a victim RL agent's sensor(s), processor(s), memory, and/or environment while the victim agent trains to learn its task. This work aims to develop and study a blackbox training-time environment-poisoning attack that modifies/poisons the dynamics of the victim agent's environment without accessing any internal mechanism of the victim. In particular, we seek a constructive attack, i.e., the objective is to push the victim to acquire an attacker-desired target-behavior. A target-behavior can be any behavior that the victim agent will not learn by itself in the original environment. The un-adoptability of this target behavior can be due to environment dynamics, sub-optimality with respect to victim's objectives, or both. Prior works focus on feasibility and hence experimented with target behaviors that are optimal (discrete environments) or nearly optimal (continuous environments) with respect to the victim's objective, but un-adoptable due to environment dynamics. This work develops and studies attacks wherein the target behavior is unadoptable in the default environment due to both, environment dynamics as well as sub-optimality with respect to victim's objectives. In addition to pushing the victim agent towards this strictly sub-optimal target behavior, the attack must also preserve the environment as much as possible or, equivalently, reduce the effort expended to modify it. Attack actions are thus constrained by the magnitude of change a single attack action is permitted to make, as well as by treating environment modification effort as a second objective in the attacker's optimization problem. The attacker, therefore, faces a multi-objective problem of finding an attack strategy that: a) generates the target behavior in the victim with high accuracy, and b) has low-effort environment modifications.

Now, commonly, an RL agent's objectives are represented by a reward signal and the agent strives to find a behavior/policy which, when executed in the given environment, maximizes the produced cumulative reward. Likewise, in the attack domain, the attacker's reward typically inculcates both attack objectives: the accuracy with which the victim adopts the target behavior, and the effort applied by the attacker, in terms of environment modifications, to achieve this accuracy. This can be done either by having several reward terms, allowing for prioritization of attacker's objectives. Or, as is done in [33, 34], by measuring the discrepancy between combined behavior-environment pairs. More specifically, Kullback Leibler Divergence Rate (KLR) can provide a unified estimate of effort and effectiveness of an attack by measuring the discrepancy between the combination of the victim's actual behavior with the poisoned environment *and* the combination of the target behavior with the default environment. Both approaches have their shortcomings. Due to high

Proc. of the Adaptive and Learning Agents Workshop (ALA 2023), Cruz, Hayes, Wang, Yates (eds.), May 29-30, 2023, London, UK, https://alaworkshop2023.github.io/. 2023.

symmetry, the KLR-based approach cannot properly distinguish between a high-accuracy, medium-effort behavior-environment pair and a medium-accuracy, low-effort pair. At the same time, weighted multiple terms of reward cannot address the fact that some behavior/environment discrepancies cancel each other and, are thus, irrelevant.

In this paper, we propose an alternative route. We avoid packing both attack effort and effectiveness into a single element of the attacker's problem. Rather, we use both the reward and the reward discounting factor to encode *and* prioritize these objectives. We propose a novel reinforcement-learning algorithm, γDDPG that supports such a dual-priority dual-objective optimization. Herein, the discount function, γ adapts in response to the current level of effort exerted by the attacker to create a bounded search space that on one hand reduces the effect of uncertainty in the given partially observable environment (blackbox setting), and on the other hand bounds the lower priority objective (attacker effort), enabling the attacker to efficiently push a victim to adopt a strictly sub-optimal target behavior in the blackbox setting.

2 RELATED WORK

Non-Constant Discounts: In this work, we seek to exploit flexible discount treatment of future rewards to capture the constraints on the attack effort. Though in a different context, several other works adapt the discount during training as well [5, 6, 8, 12, 25, 30, 31, 35, 36]. In particular, in Multi-Objective MDPs (MO-MDPs), Gunarathna et al. 2022 use state-dependent discounts to induce different time-scales for the optimization of different objectives. Their work requires specification of several hyperparameters in terms of weights as well as discounts of the different objectives and works with a fixed set of discount constants. In contrast, we introduce a novel dual-priority dual-objective MDP framework that neither needs weights and discounts to be known in advance nor requires separate optimization of each objective. In our framework, the higher priority (primary) objective is taken as the RL agent's reward, while the lower priority (secondary) objective is used to condition the discount function. The discount adapts to the current state and modifies the algorithm's search space so as to optimize the primary objective while keeping the secondary objective bounded.

Adaptive MDP: Predictive models, when used to influence the very system they model, end up introducing themselves as a variable into the system. Such predictions are termed *performative* as they can potentially modify the target distribution that they aim to predict. Predictive models that do not take performativity into account experience a *concept shift* – a change in the underlying data distribution over time – and deal with it by periodically retraining the model using new data. In fact, performative stability [2, 16, 19, 21] and model optimality [17] are a serious concern. Mandal et al. 2022 and Bell et al. 2021 study performative prediction in sequential decision-making, where the policy of a reinforcement-learning agent influences the underlying reward and transition dynamics of its environment. The agent's environment, modeled as an MDP, therefore adapts in response to the agent's behavior. They explore deterministic and non-deterministic MDP adaptation respectively and show that convergence to the optimal policy cannot be guaranteed under the latter setting. This seemingly echoes our dual-MDP architecture and bodes ill. However, if we consider the attacker to be the "performative" agent – its MDP does not actually adapt, since the victim is not aware of the attack. The victim's MDP does adapt, but the attack is explicitly calculated to *stably* influence the victim, rather than just challenging it.

Unsupervised Environment Design (UED): Design of RL environments takes a lot of time and effort, is error-prone, and is infused with designer bias. UED is a recent paradigm that aims to not only automate this step but also to generate environment distributions that are conducive to emergent complexity, robustness, and efficient transfer learning in RL agents. Adversary and evolution-based UEDs create challenging environments by aiming to minimize the learning agent's rewards [4, 18, 20], while sample-based UEDs sample environments with high learning potential [9, 10, 27]. Destructive training-time environment-poisoning attacks can be compared to a hypothetical negative-UED, whose aim is to automate design of environment distributions that result in minimization of learning agent's (victim's) performance. However, we focus on a *constructive* attack, seeking to instill a target behavior, not just destroy the victim's performance. As the target behavior is un-adoptable wrt the victim's objective, positive UEDs would not be suitable for our purpose either.

3 METHODOLOGY

This section first presents the overall interaction structure between the attacker and the victim and then describes the proposed adaptive-discounting reinforcement learning algorithm titled γDDPG, used by the attacker.

3.1 System Architecture

We follow a bi-level hierarchical framework (Figure 1), wherein the attacker as well as the victim is an independent reinforcement-learning agent with its individual learning algorithm, memory, and policy. In order to learn a given task, the victim trains to maximize its cumulative discounted rewards by interacting with the victim environment, unaware of the attacker. The attacker, on the other hand, observes these interactions of the victim with its environment, processes these observations into an approximation of the victim's behavior, and takes an action to modify the victim environment. The goal of the attacker is to sequentially and minimally modify the victim environment dynamics to drive the victim to adopt the attacker-desired target behavior. Therefore, the overall system is formed by two nested closed-loop processes, wherein the attacker, as well as the victim, is modeled as a Markov Decision Process (MDP).

Victim MDP: The victim's MDP can be denoted by the tuple $< S, A, T_{u_i}, R_v, q_0, \gamma_v >$ where $S = s_1, s_2, ...$, and $A = a_1, a_2, ...$ are the victim's states and actions respectively; $R_v : S \times A \times S \rightarrow \mathbb{R}$ is the reward function which encodes the victim's task; $\gamma_v \in (0, 1)$ is the discount factor, $q_0(S)$ is the distribution over initial states; and, $T_{u_i} : S \times A \times S \rightarrow [0, 1]$ is the probabilistic transition function, where u_i denotes the environment parameterization that has resulted from the first i interventions on the environment, by the attacker. In particular, T_{u_0} refers to the original, unaltered dynamics of the victim environment. The objective of the victim is to find an optimal policy within the experienced environment.

Attacker MDP: In our blackbox setting, the attacker's Markov process is partially observable in nature, as the attacker does not have access to the victim policy directly. However, explicitly solving such a POMDP is taxing. Instead, we take a page from the Belief MDPs [11]; our attacker views the behavior approximated from observing the victim's actions as the attacker's state, and any implied discrepancy is absorbed by the stochastic transition function. The attacker's Markov process can be represented by the tuple $< X, U, F, R, \tau^*, \gamma >$, where: $X = [T_{u_{i-1}}, \phi_{u_{i-1}}]$ is the attacker's state space comprising the victim environment dynamics, $T_{u_{i-1}}$ and the victim's behavior, $\phi_{u_{i-1}}$ that emerged in response to those dynamics; U is the attacker's action space, i.e., the set of all permissible changes that can be applied to the victim environment dynamics, such that action u_i when applied on the environment with dynamics $T_{u_{i-1}}$ results in an environment with dynamics T_{u_i}. Please note the aggregate nature of the notation: environment changes by attack actions $u_0, u_1, ..., u_i$ *accumulate* and create T_{u_i}. $F : X \times U \times X \rightarrow [0, 1]$ is the stochastic transition function that describes the response of the victim to environmental experiences, i.e., how the victim's behavior changes in response to changes in the environment dynamics; $R : X \times U \times X \rightarrow \mathbb{R}$ is the attacker reward function that describes attack effectiveness, i.e., how close the victim's behavior is to the target (attacker-desired ideal) behavior τ^*. $\gamma : X \rightarrow [\gamma_{min}, \gamma_{max}]$ is the adaptive discount function that tunes the importance of long-term rewards based on the accumulated environment modifications (attacker effort) carried out until state x_{u_i}. The attacker seeks to optimize its expected total discounted reward, wherein the combination of R and γ simulate dual optimization of (maximum) attack effectiveness with (minimum) effort, by a policy of the form $\sigma : X \rightarrow U$, $\sigma(u_i|x_{i-1})$. In other words, the attacker seeks the most *efficient* way to push the victim to converge to the target policy τ^*.

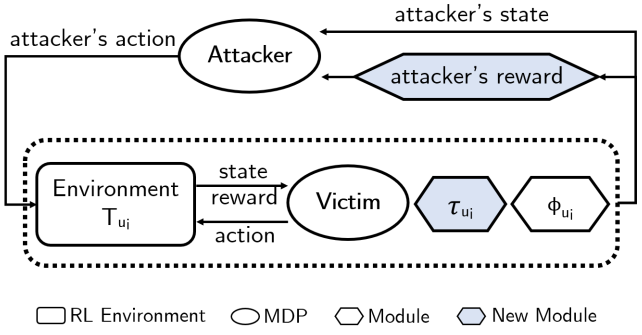


Figure 1: Bi-Level Attack Framework

3.2 γDDPG Algorithm

We train the attacker to efficiently and constructively influence the victim to adopt the attacker-desired target policy τ^*. Every action u_i of the attack is conditioned on the current victim behavior $\phi_{u_{i-1}}$ and the current victim environment dynamics $T_{u_{i-1}}$. We assume that this attack conditioning occurs in a blackbox setting, i.e., without any access to the victim's inner mechanisms or representations, during both, the attacker's training *and* testing. Thus, the victim's behavior can only be approximated through across-policy behavior traces that the attacker observes while the victim trains. In general, the victim would update its policy with a non-trivial frequency. This frequency can be so high that each state-action pair of its behavior trace would originate from a slightly different policy. In

Algorithm 1 τ Computation Algorithm

Receive victim environment with dynamics T_{u_i}
Initialize victim's Q table
Initialize τ_{u_i} with the no-action symbol for each state
Initialize state
for episode = 1, M_v **do**
  while done != True **do**
    action = Softmax_Action(Q)
    next_state, reward, done = Env-T_{u_i}(action)
    τ_{u_i}(state) $\leftarrow$ action
    Q $\leftarrow$ TD_Update(Q)
    state $\leftarrow$ next_state
  end while
end for

Algorithm 2 γDDPG Algorithm

Randomly initialize critic $Q(x, u|\theta^Q)$ and actor $\sigma(x|\theta^\sigma)$ networks with weights θ^Q and θ^σ
Initialize target networks Q' and σ' with weights $\theta^{Q'} \leftarrow \theta^Q, \theta^{\sigma'} \leftarrow \theta^\sigma$
Initialize replay buffer R
for episode = 1, M_a **do**
  Initialize a random process χ for action exploration
  Receive initial observation state x_0
  for t = 1, T_a **do**
    Select action $u_t = \sigma(x_{t-1}|\theta^\sigma) + \chi_t$ according to the current policy and exploration noise
    Execute action u_t to poison environment $T_{u_{t-1}}$
    Observe reward r_t
    $x_t \leftarrow [T_{u_t}$, Auto_Encoder(Algorithm 1 (T_{u_t}))]
    Compute γ_t using Equation 5
    Store transition $(x_{t-1}, u_t, r_t, x_t, \gamma_t)$ in R
    Sample a random minibatch of N transitions $(x_{i-1}, u_i, r_i, x_i, \gamma_i)$ from R
    Set $y_i = r_i + \gamma_i Q'(x_i, \sigma'(x_i|\theta^{\sigma'})|\theta^{Q'})$
    Update critic σ by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(x_{i-1}, u_i|\theta^Q))^2$
    Update actor policy Q using sampled policy gradient:

$$\nabla_{\theta^\sigma} J \approx \frac{1}{N} \sum_i \nabla_u Q(x, u|\theta^Q) \; |_{x=x_{i-1}, u=\sigma(x_{i-1})} \; \nabla_{\theta^\sigma} \sigma(x|\theta^\sigma) \; |_{x_{i-1}}$$

    Update the target networks Q' and σ':

$$\theta^{Q'} \leftarrow \rho\theta^Q + (1-\rho)\theta^{Q'}$$
$$\theta^{\sigma'} \leftarrow \rho\theta^\sigma + (1-\rho)\theta^{\sigma'}$$

  end for
end for

our experiments, we assume as much. Now, to approximate the victim's policy we would need traces of multiple epochs of the victim's training process. But conditioning an attack on such a large volume of data is impractical. Instead, we preprocess these traces by storing the last observed victim action corresponding to each observed victim state and assign a "no-action" symbol to unvisited states. This behavior information will hereafter be denoted as $\tau_{u_{i-1}} = \{s_1, a_1; s_2, a_2; ...; s_N, a_N\} \forall s_n \in S$, where a_n is the latest action taken by the victim in state n or the no-action symbol in case state s_n was never visited by the victim, and N is the total number of states in the victim environment. As $\tau_{u_{i-1}}$ contains the latest action / no-action symbol corresponding to all states, $\tau_{u_{i-1}}$'s size can still explode in high-dimensional environments. To combat this issue, the current paper learns a low-dimensional latent space, Φ of victim behaviors using an auto-encoder model. The model consists of an encoder q_e that takes the victim's $\tau_{u_{i-1}}$ as input and outputs the corresponding latent behavior $\phi_{u_{i-1}}$; and a decoder q_d that takes two inputs, the latent behavior $\phi_{u_{i-1}}$ and a victim environment state s_n, and outputs the probability with which the victim will take each available action in the given state s_n.

Now, as our attacker-MDP suggests, we seek to balance attack effectiveness and effort. However, unlike prior works, where the balance was achieved through *reward* elements' merge, we distribute the responsibility between distinct MDP components. Namely, while attack effectiveness (accuracy) remains with the reward function, the effort is controlled by a dynamic reward discounting. We implement this architecture in our γDDPG algorithm (Figure 2), appropriately subverting the original DDPG [14]. γDDPG is made capable of prioritizing attack accuracy over attacker effort by maximizing attack accuracy within an effort-bounded search space. This search space is created at every attack step by adapting the attacker MDP's discount factor (γ) conditioned on the current attacker effort. The adapting discount factor modifies γDDPG's Bellman update to alter the level of importance that the algorithm accords to long-term rewards.In further detail, when the attacker begins to train using γDDPG and takes the first action to poison the victim's environment dynamics, its next-state is a low attacker-effort state due to existence of magnitude constraints on attack actions. Corresponding to this low effort, the discount function (γ) takes a small value. A small discount biases the attacker's precedence to short-term rewards, pushing γDDPG to search for a high-accuracy state near the current state. As attack actions are constrained in nature, this high-accuracy state will be reachable using a bounded level of effort i.e. it will require the attacker to make only a few modifications to the victim's original environment. The constraints, therefore, ensure that the attacker effort and thereby the discount factor increase gradually during the search process. A greater discount factor increases the radius of search for γDDPG, as high rewards in further states begin getting noticed. γDDPG therefore gradually moves to further-away states (higher-effort states) when it is unable to find a higher-accuracy state nearby (low-effort). This increase in radius also increases the probability of γDDPG finding a way back to (now far-away) low-effort states, if they can provide the attacker with higher accuracy than seen so far. A jump to a low-effort state will again reduce the discount factor, enabling γDDPG to search for better accuracy states near this low-effort state. Therefore, at each timestep, the adaptive discount factor creates a bounded-effort search space similar to the trust regions created by adaptive step sizes [24]; and γDDPG looks for the highest accuracy state within this space.

The aforementioned adaptive discount which is a function of the effort executed by the attacker on the victim environment can be modeled in different ways. The attacker executes effort on the victim environment to poison/modify the environment's transition dynamics. This effort can therefore be computed in terms of the distance between the original and the current environment dynamics. However different environment dynamics at the same distance to the original dynamics can result in different accuracy of target behavior adoption. Therefore, we adopt the aforementioned MDP-based formulation but modify it to compute distance between target-current (current environment * target behavior) and perfect (original environment * target behavior) MDPs instead of vanilla-current (current environment * current behavior) and perfect MDPs. The target-current MDP models the current environment dynamics coupled with the victim's target behavior as a stochastic Markov process and therefore computes target-behavior conditioned effort associated with the current state. The target-current MDP over state-action pairs is here onwards denoted as $P_{u_i}(s_{j+1}, a_{j+1}|s_j, a_j)$. Here, j denotes victim-level time step, just as i has been used to denote attacker-level time step. This target-current process is described in Equation 3. The effort is then computed as the Wasserstein distance between the k^{th} step distributions corresponding to the target-current and perfect processes respectively. This work adopts the partial target behavior design of Xu et al. [34]:

$$\tau^*_{u_i}(s) = \begin{cases} a^*_n & s_n \in S^* \\ \tau_{u_i}(s_n) & s_n \notin S^* \end{cases} \tag{1}$$

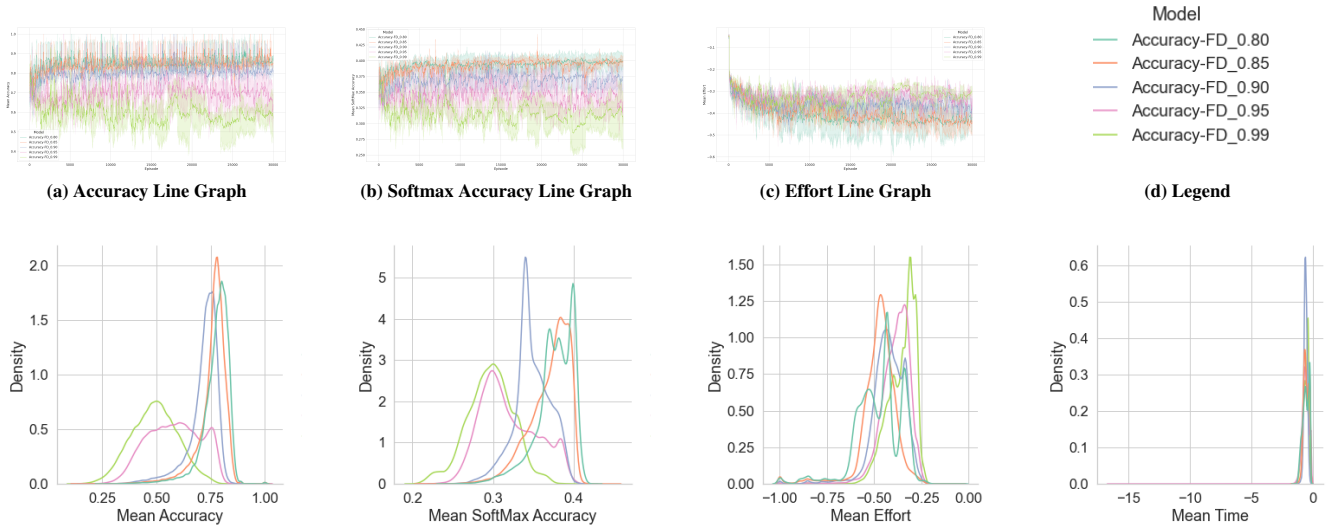
Here S^* is the target state set, a^*_n is the target action for target state s_n, and $\tau_{u_i}(s)$ is the latest behavior of the victim observed in the environment with transition dynamics T_{u_i}. As the attacker cannot access the victims' policy in the given blackbox setting, it approximates the victim policy π_{u_i}, using the last h actions taken by the victim in each state. The target policy distribution $\pi^*_{u_i}$ is then constructed by taking a copy of π_{u_i} and modifying it by assigning probability 1.0 to all target actions (and 0.0 to non-target actions) w.r.t. each target state. The vanilla-current $P^v_{u_i}$, target-current P_{u_i} and perfect $P^*_{u_0}$ processes are defined as:

$$\begin{aligned} \text{Vanilla-Current MDP} &= P^v_{u_i}(s_{j+1}, a_{j+1}|s_j, a_j) \\ &= T_{u_i}(s_{j+1}|s_j, a_j)\ \pi_{u_i}(a_{j+1}|s_{j+1}) \end{aligned} \tag{2}$$

$$\begin{aligned} \text{Target-Current MDP} &= P_{u_i}(s_{j+1}, a_{j+1}|s_j, a_j) \\ &= T_{u_i}(s_{j+1}|s_j, a_j)\ \pi^*_{u_i}(a_{j+1}|s_{j+1}) \end{aligned} \tag{3}$$

$$\begin{aligned} \text{Perfect MDP} &= P^*_{u_0}(s_{j+1}, a_{j+1}|s_j, a_j) \\ &= T_{u_0}(s_{j+1}|s_j, a_j)\ \pi^*_{u_i}(a_{j+1}|s_{j+1}) \end{aligned} \tag{4}$$

As noted in Section 1, prior works utilize KLR [23]) to compute the divergence between Markov processes [33, 34] and utilize negative of this divergence as the attacker reward. KLR computes divergence between probability distributions of different trajectories in the two given Markov processes but does not take the underlying metric space into account. This work hypothesizes that a measure that computes the distance between the k^{th} step probability distributions of two Markov processes while respecting the underlying



(a) Accuracy Line Graph (b) Softmax Accuracy Line Graph (c) Effort Line Graph (d) Legend

(e) Accuracy Histogram (f) SoftMax Accuracy Histogram (g) Effort Histogram (h) Time Histogram

Figure 2: Accuracy, Effort, and Time of γDDPG with fixed Bellman discounts 0.80, 0.85, 0.90, 0.95, and 0.99

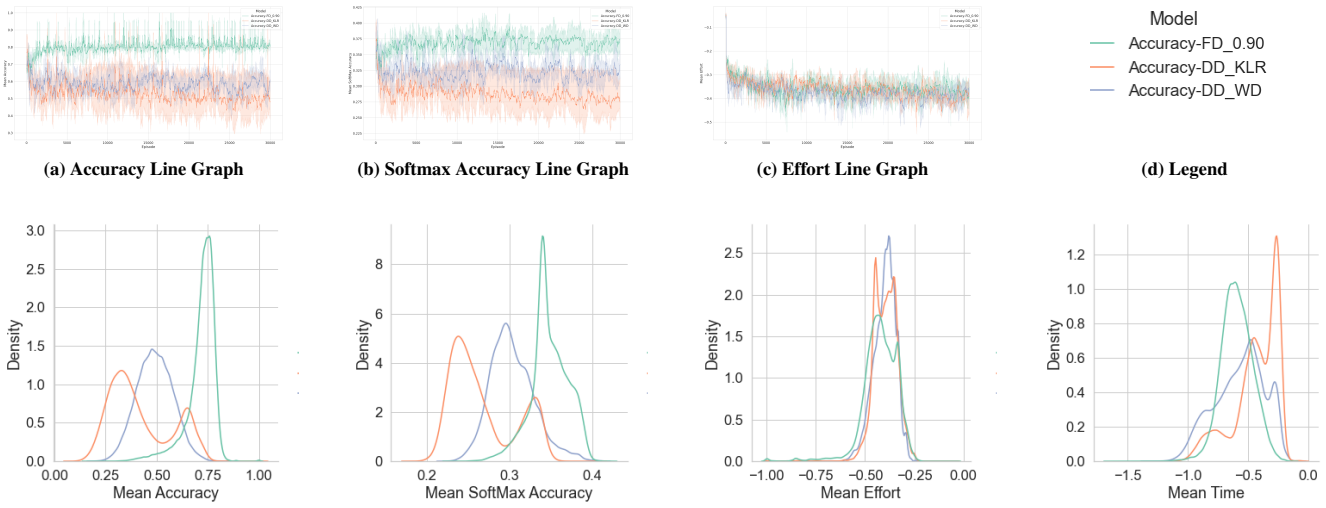
geometry of the metric space provides a better estimate of the difference between the two given Markov processes when compared to KLR. To test this hypothesis we use Wasserstein distance to compute the distance between the target-current and perfect Markov processes. Wasserstein distance possesses an additional property of being insensitive to small changes in the probability distributions. This property is advantageous in the current uncertain blackbox setting where τ_{u_i} being an approximate representation of the actual policy of the victim can be noisy in nature. Let $p^k_{u_i}$ and $p^{k*}_{u_0}$ be the k^{th} step probability distributions of the target-current and perfect processes respectively. The Wasserstein 1-distance between these distributions termed TargetWD is defined below. Here, β is a transport plan, $d(x, y)$ is the distance between x and y, and $p^k_{u_i}$ and $p^{k*}_{u_0}$ are written as p^k and p^{k*} respectively.

$$\gamma \propto \text{TargetWD}(p^k, p^{k*}) := \left(\inf_{\beta \in \text{B}(p^k, p^{k*})} \mathbb{E}_{(x,y)\sim\beta} d(x, y)\, \text{d}\beta(x, y) \right) \tag{5}$$

4 EXPERIMENTS

Several works in the training-time attack domain develop, test, and study attacks on a navigational agent (victim) whose objective is to find the shortest path to the goal state [22, 32–34]. Herein the victim's navigation environment can be discrete or continuous. In discrete environments, the target behavior selected in these works was an alternate path to the goal state that is un-adoptable due to environment dynamics but equal in length to the optimal path (i.e. optimal with respect to the victim's objectives). On the other hand, in continuous environments, a longer trajectory along the optimal path was selected as the target behavior (i.e. nearly-optimal with respect to the victim's objectives). The current research aims to build an attacker that learns a high-accuracy, low-effort strategy to modify the stochastic dynamics of a discrete environment in order to push a training victim agent to learn a strictly sub-optimal target behavior. This target behavior must be un-adoptable in the default environment due to both, environment dynamics as well as sub-optimality with respect to victim's objectives. In the following experiments, a path that is three times the length of the optimal path (shortest high-probability path under the original dynamics) is chosen as the strictly sub-optimal target behavior.

In order to better align our contribution with prior works, we utilize the 3D Grid World [22] to test and establish the quality of the proposed methodology. This environment simulates an uneven terrain on a $2D$ grid of cells. The unevenness corresponds to the 3^{rd} dimension of the grid and is due to the elevation/altitude associated with each grid cell. The relative elevation between two cells affects the transition probability between them. A change in this relative elevation thus changes how the environment responds to a navigating agent's actions. The navigating agent's task is to find the optimal path from the start cell to the goal cell and its state is its position inside the grid world. At each time step, the agent observes its state and takes one step in any of the four cardinal directions (N,S,E,W). The agent receives a reward of -1 for every action it takes until it reaches the goal state. A given agent training episode terminates once the agent reaches the goal state or maximum time has elapsed. This pushes the agent to find a shortest path to the goal cell. This environment also allows the presence of an additional agent, the elevation expert who can view the altitude of each grid cell and take a constrained action to modify it. The elevation expert's state space comprises of the grid cells' altitudes along with the navigational



(a) Accuracy Line Graph **(b) Softmax Accuracy Line Graph** **(c) Effort Line Graph** **(d) Legend**

(e) Accuracy Histogram **(f) SoftMax Accuracy Histogram** **(g) Effort Histogram** **(h) Time Histogram**

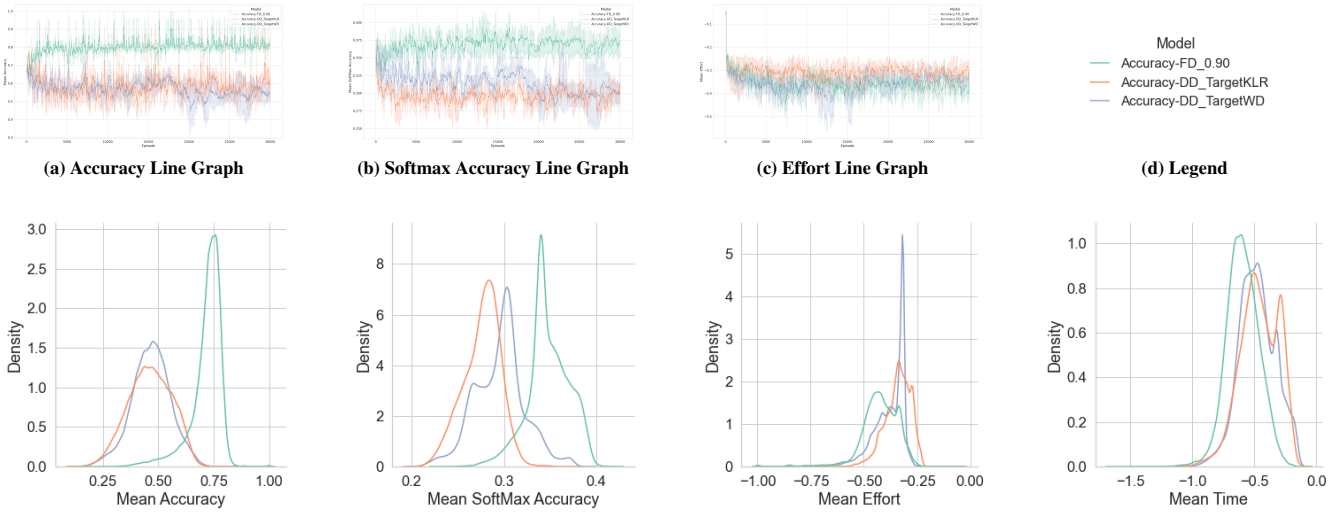
Figure 3: Accuracy, Effort, and Time of γDDPG with adaptive Bellman discounts KLR and WD

agent's behavior, while its action space is a vector of real numbers $[x^1, x^2, x^3, ..., x^M], x \in [-1.0, 1.0]$ where M is the total number of cells in the grid. In this work, the victim is a navigating agent while the attacker is the elevation expert.

The performance of the attacker is measured in terms of the accuracy (Attack Accuracy) and strength (Attack SoftMax Accuracy) with which the victim (unknowingly) adopts the target behavior; the cumulative changes brought about in the victim environment by the attacker (Attacker Effort); and time taken to carry out the attack (Attack Time). **Attack Accuracy (**abbreviated as **@Acc)** computes the level of adoption of the target behavior by the victim. A target state (state included in the target path) is assigned an adoption accuracy of 1.0 if the victim assigns highest probability to the target action (attacker-desired victim-action) in that state. The final accuracy is the sum of the adoption accuracies of all target states, divided by the number of target states. The convergence rate of this measure reflects how quickly the attack enabled target behavior adoption in the victim. **Attack SoftMax Accuracy (@SoftAcc)** computes the probability assigned to the target path by the victim and is computed as the sum of the target actions' probabilities in the target states, divided by the number of target states. This measure reflects the strength with which the victim adopts the target behavior. **Attacker Effort (@Effort)** computes the degree to which the attacker modifies the environment and is computed as the mean of the absolute difference between the previous and current attack time step's grid cells' altitudes. **Attack Time (@Time)** is the computation time taken by the attacker to carry out an attack action. @Time is inclusive of the observation-time wherein the attacker observes the victim while it trains in the attacked/poisoned/modified environment.

The core contribution of this work is a novel RL algorithm, γDDPG that is capable of carrying out dual-priority dual-objective optimization. We investigate several variations of this algorithm and compare the best-performing variations to a state-of-the-art baseline, TEPA. The performance of the various models is demonstrated via histograms that capture overall statistics as well as line graphs that depict performance across training time. In line graphs, the mean of each metric is computed for every attacker training episode and its maximum value across a sliding-window of length 75 is presented such that the x-axis represents the attacker training episodes and the y-axis represents the maximum mean value of the metric within the window. The attacker training episode is a 15-step sequential attack on a freshly initialized victim wherein attack step 0 corresponds to the original environment with default dynamics, and the episode ends when the victim has adopted the target behavior with 1.0 @Acc or max attack steps (15) have elapsed. Each histogram and line graph represents four training runs with different seeds, except for the ones corresponding to Experiment 4 wherein a single training run is presented because the baseline model requires an exorbitant amount of time to carry out the attacker's training. The effort and time plots represent negative of mean @Effort and mean @Time respectively in order to standardize that values on the right in histograms and graphs on top in line graphs are better, across all metrics.

This work takes state-of-the-art training-time environment-poisoning attack generation methodology, TEPA [34] as its baseline. TEPA is an auto-encoder-based model that is shown capable of pushing a victim agent in whitebox and proxy-blackbox adversarial settings to adopt a target behavior that is optimal with respect to the victim's objectives, but un-adoptable due to environment dynamics. TEPA uses state, action trajectories to represent the victim's behavior and utilizes the negative KLR between the vanilla-current and perfect processes as the attacker's reward.



(a) Accuracy Line Graph **(b) Softmax Accuracy Line Graph** **(c) Effort Line Graph** **(d) Legend**

(e) Accuracy Histogram **(f) SoftMax Accuracy Histogram** **(g) Effort Histogram** **(h) Time Histogram**

Figure 4: Accuracy, Effort, and Time of γDDPG with adaptive Bellman discounts TargetKLR and TargetWD

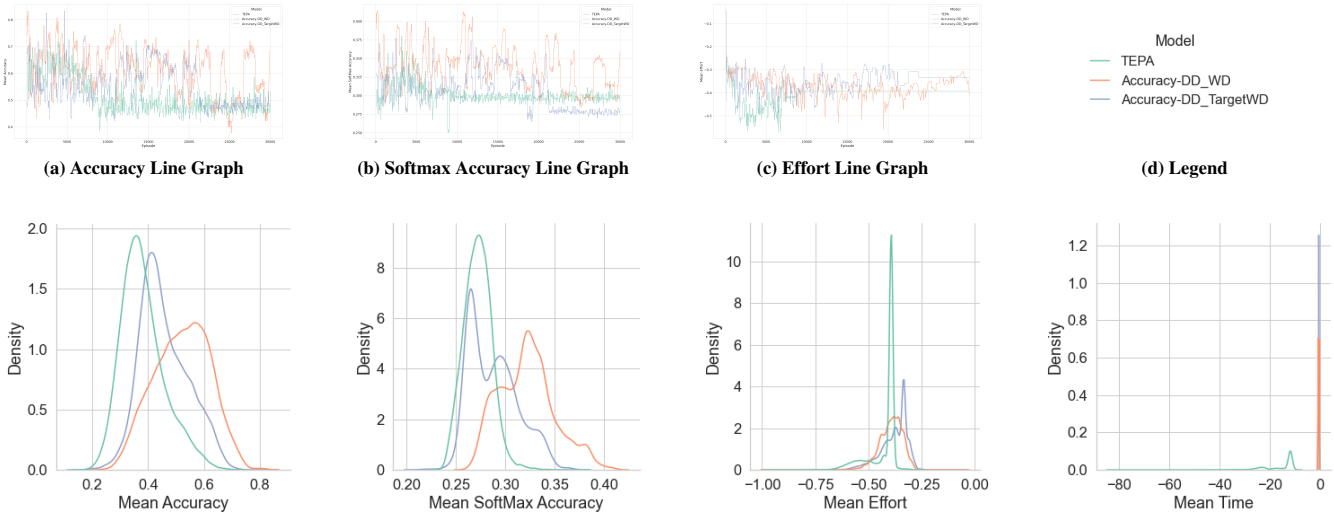
Experiment 1 is designed to demonstrate the capability of the discount factor to function as a means of bounding the lower-priority objective (minimize @Effort) while reducing the effect of uncertainty so as to aid in the optimization of the primary objective (maximize @Acc) in a high-dimensional space. Figure 2 shows that strategies found by lower discount factors exert a slightly higher @Effort to achieve high @Acc and @SoftAcc. This implies that reducing the search space around the current state, reduces the effect of uncertainty in the high-dimensional blackbox (partially-observable) setting; enabling γDDPG to find strategies that achieve high @Acc while exerting a bounded @Effort. Additionally, the variance of @Acc and @SoftAcc across training increases with increasing γ. This reflects the difficulty faced by RL algorithms while optimizing in high-dimensional non-convex spaces and illustrates the potential of the discount factor in facilitating dual-priority dual-objective optimization. In this work, 0.90 is chosen as the best fixed discount as it offers the best balance between @Acc and @Effort. However, as this methodology requires a grid-search to find the optimal fixed discount factor (γ), it cannot be used in settings where the victim task or victim environment changes with time, in a manner that the optimal discount factor of the attacker also undergoes modification. This problem associated with fixed discounts is solved in the current work with the aid of **TargetWD** based adaptive/dynamic discount introduced in Section 3.

Experiments 2 and 3 compare different adaptive discounts with the best fixed discount (0.90) found via grid search. Prior works [22, 34] utilize negative KLR between the vanilla-current (Eq. 2) and perfect MDPs (Eq. 4) as the attacker reward because reducing this KLR pushes the current MDP towards the perfect MDP (high @Acc and low @Effort). The adaptive discount function proposed in this paper (TargetWD) on the other hand computes Wasserstein Distance (WD) between target-current (Eq. 3) and perfect MDPs (Eq. 4), which only incorporates @Effort with respect to the target behavior. In order to better understand the individual contributions of the WD metric and target-current MDP based distance in bounding the lower-priority objective (@Effort) and reducing uncertainty, we compare four adaptive discount functions:

- KLR - KLR(Vanilla-Current MDP, Perfect MDP)
- WD - WD(Vanilla-Current MDP, Perfect MDP)
- TargetKLR - KLR(Target-Current MDP, Perfect MDP)
- TargetWD - WD(Target-Current MDP, Perfect MDP)

These four adaptive discount functions must undergo normalization as the range of KL divergence [13] and Wasserstein distance [28] is $[0, \infty]$. A short single-seed experiment of 3k training episodes was conducted to test the effect of different normalization ranges on the performance of KLR and WD adaptive discounts. In this experiment, different normalization ranges with lower bounds greater than 0.5 and higher bounds equal to 0.99 were tested. These values were chosen to support prioritization of maximizing accuracy over minimizing effort. WD achieved higher mean @Acc with high frequency in all ranges. In order to give KLR adaptive discounts a better chance, we optimized both ranges to compare the best performers from each method.

Experiment 2, presented in Figure 3 compares KLR and WD adaptive discounts to 0.90 fixed discount, while Experiment 3, presented in Figure 4 compares TargetKLR and TargetWD adaptive discounts to 0.90 fixed discount. WD and TargetWD adaptive discounts frequently find strategies with higher @Acc and @SoftAcc than KLR and TargetKLR adaptive discounts respectively. Moreover, some of the strategies found by WD and TargetWD have better @SoftAcc than majority of the strategies found by best fixed discount found via grid-search (0.90). Furthermore, all adaptive discounts are able to achieve these accuracies while executing a bounded level of @Effort on the victim environment. These results imply that in the given



(a) Accuracy Line Graph **(b) Softmax Accuracy Line Graph** **(c) Effort Line Graph** **(d) Legend**

(e) Accuracy Histogram **(f) SoftMax Accuracy Histogram** **(g) Effort Histogram** **(h) Time Histogram**

Figure 5: Accuracy, Effort, and Time of baseline TEPA vs γDDPG with adaptive Bellman discounts WD and TargetWD

setting, in order to ensure a high level of adoption (@Acc) as well as adherence (@SoftAcc) to the target behavior, the adaptive discount function must respect the underlying geometry of the metric space; compute the difference between two Markov processes using distances between k^{th} step probability distributions instead of divergence between trajectory distributions; and be insensitive to small differences in the k^{th} step probability distributions corresponding to the Markov processes. Furthermore, the @Time graph in Figures 3 and 4 show that adaptive discounts find strategies that take lesser time to carry out the attack compared to strategies found by the best fixed discount.

Experiment 4, presented in Figure 5 compares the best-performing models in experiments 2 and 3 i.e. WD and TargetWD, with the state-of-the-art baseline TEPA, in order to highlight the overall contribution of this work. In comparison to strategies found by WD and TargetWD adaptive discounts, TEPA strategies achieve much lower @Acc and @SoftAcc in spite of taking 180 times more @Time to execute. Moreover, it is interesting to note that after approximately 11k training episodes, @Acc and @SoftAcc of TEPA begin to fluctuate within very small fixed ranges while @Effort and @Time plots become almost constant. This suggests that TEPA gets stuck in a local optima and is unable to completely exit it even after approximately 20k training episodes. On the other hand, while both WD and TargetWD strategies perform better than the baseline w.r.t. all metrics; WD strategies achieve higher @Acc and @SoftAcc compared to TargetWD whereas TargetWD strategies execute lower @Effort. Given that lower @Effort is preferred in this work, we choose TargetWD as the best adaptive discount.

5 CONCLUSION AND FUTURE WORK

This paper introduces a novel category of training-time environment-poisoning attacks wherein the attacker pushes the victim towards a strictly sub-optimal target behavior. This strictly sub-optimal target behavior is un-adoptable in the original environment due to both, environment dynamics as well as sub-optimality with respect to victim's objectives. In order to make an attacker capable of carrying out such attacks, we introduce a novel reinforcement-learning algorithm titled γDDPG that utilizes an adaptive Bellman discount factor to support dual-priority dual-objective optimization in a partially observable setting. This dynamic discount bounds γDDPG's search space conditioned on the accumulated modifications executed on the victim environment until the current attack step. The bounded search space, on one hand, bounds the lower priority objective (minimize Attacker Effort) and on the other hand, reduces uncertainty associated with the partially-observable environment and thereby aids in optimization of the primary objective (maximize Attack Accuracy). We show that Wasserstein distance based adaptive discounts perform better than Kullback Leibler divergence based adaptive discounts.

The attacker approximates the victim's policy using the last action taken by the victim in each environment state. This mechanism can however only be used for victims training in an environment with discrete state space. Similarly, the current formulation of TargetWD adaptive discount requires the underlying target-current MDP and perfect MDP to be constructed on an environment with discrete state and action spaces. Our next step entails extension of the proposed methodology to continuous environments. Moreover, the proposed algorithm supports only dual-objective optimization with two levels of priority. Future work constitutes expanding the developed methodology to multi-objective optimization with more than 2 objectives and priority levels.

ACKNOWLEDGMENTS

This research was supported in part by the NTU SUG "Choice manipulation and Security Games".

REFERENCES

[1] J. Bell, L. Linsefors, C. Oesterheld, and J. Skalse. 2021. Reinforcement learning in Newcomblike environments. *NeurIPS* 34 (2021), 22146–22157.

[2] G. Brown, S. Hod, and I. Kalemaj. 2022. Performative prediction in a stateful world. In *AIStats*. 6045–6061.

[3] N. Brown and T. Sandholm. 2019. Superhuman AI for multiplayer poker. *Science* 365, 6456 (2019), 885–890.

[4] M. Dennis, N. Jaques, E. Vinitsky, A. Bayen, S. Russell, A. Critch, and S. Levine. 2020. Emergent complexity and zero-shot transfer via unsupervised environment design. In *NeurIPS*. 13049–13061.

[5] V. François-Lavet, R. Fonteneau, and D. Ernst. 2015. How to discount deep reinforcement learning: Towards new dynamic strategies. *arXiv preprint arXiv:1512.02011* (2015).

[6] Y. Gu, Y. Cheng, C. L. P. Chen, and X. Wang. 2021. Proximal Policy Optimization With Policy Feedback. *IEEE Trans. SMC: Systems* (2021).

[7] U. Gunarathna, S. Karunasekara, R. Borovica-Gajic, and E. Tanin. 2022. Intelligent Autonomous Intersection Management. *arXiv preprint arXiv:2202.04224* (2022).

[8] L. Hou, Zhengming Wang, and Han Long. 2021. An Improvement for Value-Based Reinforcement Learning Method Through Increasing Discount Factor Substitution. In *Int. Conf. on CSE*. 94–100.

[9] M. Jiang, M. Dennis, J. Parker-Holder, J. Foerster, E. Grefenstette, and T. Rocktäschel. 2021. Replay-guided adversarial environment design. In *NeurIPS*. 1884–1897.

[10] M. Jiang, E. Grefenstette, and T. Rocktäschel. 2021. Prioritized level replay. In *ICML*. 4940–4950.

[11] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101, 1-2 (1998), 99–134.

[12] M. S. Kim, J.-S. Kim, M.-S. Choi, and J.-H. Park. 2022. Adaptive Discount Factor for Deep Reinforcement Learning in Continuing Tasks with Uncertainty. *Sensors* 22, 19 (2022), 7266.

[13] S Kullback and R Leibler. 1951. On information and sufficiencyannals of mathematical statistics, 22, 79–86. *MathSciNet MATH* (1951).

[14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and . Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[15] D. Mandal, S. Triantafyllou, and G. Radanovic. 2022. Performative Reinforcement Learning. *arXiv preprint arXiv:2207.00046* (2022).

[16] C. Mendler-Dünner, J. Perdomo, T. Zrnic, and M. Hardt. 2020. Stochastic optimization for performative prediction. *NeurIPS* 33 (2020), 4929–4939.

[17] J. P. Miller, J. C. Perdomo, and T. Zrnic. 2021. Outside the echo chamber: Optimizing the performative risk. In *ICML*. 7710–7720.

[18] J. Morimoto and K. Doya. 2005. Robust reinforcement learning. *Neural computation* 17, 2 (2005), 335–359.

[19] A. Narang, E. Faulkner, D. Drusvyatskiy, M. Fazel, and L. J. Ratliff. 2022. Multiplayer performative prediction: Learning in decision-dependent games. *arXiv preprint arXiv:2201.03398* (2022).

[20] J. Parker-Holder, M. Jiang, M. Dennis, M. Samvelyan, J. Foerster, E. Grefenstette, and T. Rocktäschel. 2022. Evolving Curricula with Regret-Based Environment Design. *arXiv preprint arXiv:2203.01302* (2022).

[21] J. Perdomo, T. Zrnic, C. Mendler-Dünner, and M. Hardt. 2020. Performative prediction. In *ICML*. 7599–7609.

[22] Z. Rabinovich, L. Dufton, K. Larson, and N. R. Jennings. 2010. Cultivating Desired Behaviour: Policy Teaching Via Environment-Dynamics Tweaks. In *AAMAS*. 1097–1104.

[23] Z. Rached, F. Alajaji, and L. L. Campbell. 2004. The Kullback-Leibler Divergence Rate between Markov Sources. *IEEE Transactions on Information Theory* 50, 5 (2004), 917–921.

[24] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. 2015. Trust region policy optimization. In *ICML*. 1889–1897.

[25] A. Sharma, R. Gupta, K. Lakshmanan, and A. Gupta. 2021. Transition based discount factor for model free algorithms in reinforcement learning. *Symmetry* 13, 7 (2021), 1197.

[26] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. 2017. Mastering the game of go without human knowledge. *Nature* 550, 7676 (2017), 354–359.

[27] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*. 23–30.

[28] L. N. Vaserstein. 1969. Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredachi Informatsii* 5, 3 (1969), 64–72.

[29] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019), 350–354.

[30] Q. Wei and X. Guo. 2011. Markov decision processes with state-dependent discount factors and unbounded rewards/costs. *Operations Research Letters* 39, 5 (2011), 369–374.

[31] M. White. 2017. Unifying task specification in reinforcement learning. In *ICML*. 3742–3750.

[32] H. Xu, X. Qu, and Z. Rabinovich. 2022. Spiking Pitch Black: Poisoning an Unknown Environment to Attack Unknown Reinforcement Learners. In *AAMAS*. 1409–1417.

[33] H. Xu and Z. Rabinovich. 2021. Truly Black-box Attack on Reinforcement Learning via Environment Poisoning. In *ALA WS at AAMAS*.

[34] H. Xu, R. Wang, L. Raizman, and Z. Rabinovich. 2021. Transferable Environment Poisoning: Training-time Attack on Reinforcement Learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. 1398–1406.

[35] N. Yoshida, E. Uchibe, and K. Doya. 2013. Reinforcement learning with state-dependent discount factor. In *ICDL*. 1–6.

[36] M. Zinzuvadiya and V. Behzadan. 2021. State-Wise Adaptive Discounting from Experience (SADE): A Novel Discounting Scheme for Reinforcement Learning (Student Abstract). In *AAAI*. 15953–15954.