Truly Black-box Attack on Reinforcement Learning via Environment Poisoning

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

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

ABSTRACT

Transferable environment-poisoning attack (TEPA) has been proposed recently to force a Reinforcement Learning (RL) agent to learn a target policy designed by an attacker. Comparing with the majority of training-time attack approaches, TEPA is more realistic because of its effectiveness for black-box RL agents. However, the knowledge of the victim's environment dynamics model is at the heart of TEPA's foundation, which constraints its scalability to sophisticated real-world scenarios. In this paper, we study the training-time attack on RL in truly black-box settings where both RL agent and environment dynamics are unknown. We leverage latent representation of the RL agent's stochastic process (i.e., the combination of policy and dynamics) to train an adaptive attack strategy. In the preliminary experiment, we show that our attack strategy performs successfully in an unknown environment, and is generally effective for different black-box RL agents.

KEYWORDS

Reinforcement Learning; Environment Poisoning; Security

1 INTRODUCTION

Adversarial attacks on Reinforcement Learning (RL) have been widely studied to understand its vulnerability, which promotes robust RL algorithm design. Test-time attacks [1–3, 5, 11] and training-time attacks [6, 9, 12, 13] are two main categories in this research area. Unlike test-time attacks which degrade the performance of a deployed RL policy, training-time attacks interfere with the agent's learning process via reward poisoning [6, 13] or environment-dynamics poisoning [9, 12], and thereby enforce a target policy designed by the attacker.

Most training-time attack approaches are developed in white-box settings where all the information about the RL agent is known by the attacker, such as the agent's learning algorithm and policy model. Recently, Xu et al. [12] propose a transferable environment-poisoning attack (TEPA) which is effective for black-box RL agents (i.e., learning algorithm and policy model are unknown). In TEPA, an attack strategy is trained based on a white-box proxy agent, and then transferred to attack a black-box victim agent. Such effectiveness on black-box victims makes the training-time attack more realistic in real-world scenarios. However, the knowledge of the environment dynamics model lies at the heart of TEPA's foundation. In other words, the victim's environment is assumed to be white-box for the attacker. This assumption limits the scalability and applicability of TEPA because it is difficult for the attacker to know environment dynamics models in sophisticated simulations or real-world applications. For example, the attacker can manipulate the road friction to affect the self-driving policy. However, it is challenged for the attacker to exactly know how the road condition responds to brakes (i.e., environment dynamics model). Therefore, it is necessary to develop an environment-poisoning attack method in *double* black-box settings, where both the victim's learning algorithm and the environment dynamics model are unknown to the attacker. We term such an attack approach as Truly Black-box Attack.

In this work, we study the environment-poisoning attack problem in the double black-box setting. We inherit the attack framework from TEPA, and pursue the same attack objective as TEPA. Specifically, the attack problem is formulated as bi-level Markov Decision Process (MDP) architecture. The attack objective is to stealthily force an unknown RL agent to learn a target behaviour by tweaking the unknown environment. To handle the double black-box setting, we leverage an Encoder Dual-Decoder network to learn the latent representation of the victim's stochastic process from its experienced trajectories. Here, the victim's stochastic process is determined by its policy and environment dynamics. The latent representation is set as the attacker's input, which enables attack actions to respond to the victim's behaviour features and the environment dynamics features. Additionally, the attacker's optimization objectives should be built as the combined deviation of *a)* the victim's actual policy from the target one; *b)* the tweaked environment dynamics from their natural form. Since both victim's policy and dynamics are unknown, we approximate the combined deviation in the latent space.

Overall, the contributions of this paper can be summarized as follows:

- We study the environment-poisoning attacks on training-time RL in *double* black-box settings, where victim's policy and environment dynamics are unknown.
- We propose to learn the latent representation of the victim's stochastic process from its trajectories, and thereby train an adaptive attack strategy directly on a black-box victim.
- We present preliminary experimental results to show that the attack strategy is effective for different black-box victims in one unknown environment.

2 RELATED WORK

In this section, we review existing works about adversarial attacks against RL which are classified as two branches: test-time attacks and training-time attacks.

For test-time attacks, the attacker's objective is to degrade the deployment performance of the agent's learned and fixed policy. Here, the policy itself is not subject to manipulation. Test-time

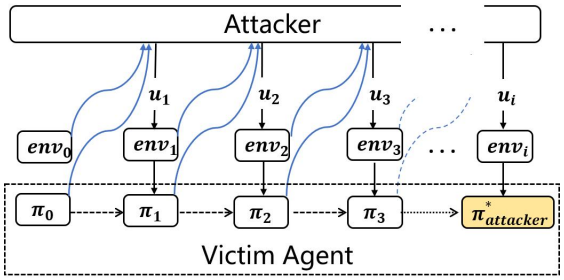

Figure 1: Attack Deployment [12]

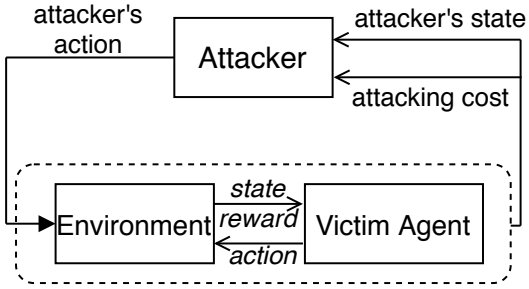

Figure 2: Architecture illustration [12]

attacks have been developed in both white-box settings [3, 5, 11] and black-box settings [1, 2]. In white-box settings, the victim's learning algorithm and policy model are known by the attacker. When deploying the policy, the victim's perceived states are perturbed, and thereby its decision on selecting actions is misled. In black-box settings, attack approaches [1, 2] utilize transferability of adversarial examples (i.e., crafted malicious inputs [7]) to attack different RL agents which perform the same task.

Different from test-time attacks, training-time attacks propel the RL agent towards a target policy designed and chosen by the attacker. To manipulate the agent's policy itself, the attacker is required to interfere with the agent's learning process from the perspectives of reward signals or environment dynamics. First, some approaches [6, 13] poison the agent's policy by manipulating rewards at training time. For example, [6] poisons rewards in the training data set for batch RL, and [13] studies adaptive reward-poisoning attacks on a Q-learning RL agent. Second, some approaches [9, 12] investigate environment-dynamics poisoning attacks. For instance, [9] designs environment transition probabilities for cyclic RL tasks while [12] changes environment hyper-parameters in episodic MDP.

For training-time attacks on RL, the majority of existing approaches [6, 9, 13] is infeasible for black-box RL agents, except that [12] leverages the transferability of the poisoned training environment to attack black-box RL agents. However, [12] designs its attack strategy based on the white-box environment-dynamics model. In other words, the attacker is assumed to know how the environment responds to the agent's action. This assumption is difficult to be satisfied in real-world applications or sophisticated simulations. To further improve the applicability of training-time attacks, we study training-time attacks on RL in *double* black-box settings. Specifically, our proposed method enables the attacker to learn an adaptive attack strategy on a black-box victim in an unknown environment.

3 PRELIMINARY

In TEPA [12], the attacker enforces a target policy on the victim by manipulating how the environment responds to the victim's action. As Figure 1, responding to the victim's momentary policy, the attacker tweaks the victim's training environment at regular intervals of the victim's learning process, until the victim acquires a target policy designed by the attacker. TEPA formulates the environment-poisoning attack problem as a bi-level Markov Decision Process (MDP) architecture which consists of a victim-level MDP and an attack-level MDP, shown as Figure 2.

Victim-level MDP. The victim is an RL agent of which Markovian environment is represented by the tuple $< S, A, T, d_0, \hat{\gamma}, r >$. Here, S is the set of environment states and $a \in A$ is the agent's action. $T(s'|s, a, u_{1:i})$ represents the victim's environment dynamics function which is affected by the aggregate changes of hyper-parameters $u_{1:i}$ (referring to attack-level MDP). It describes how the victim's state s changes given that its environment was changed by $u_{1:i}$ and its action was a. $d_0(s)$ is a distribution over the agent's initial states and $\hat{\gamma}$ is the discount factor. $r : S \times A \times S \to \mathbb{R}$ represents the victim's reward function. The victim aims to learn an optimal policy $\pi(a|s)$ for maximized cumulative discounted rewards.

Attack-level MDP. The attacker is regarded as an RL agent which regards the victim as a dynamic system. The attacker's environment is formulated as a Markovian process, which is described as the tuple $< \Theta, U, F, \theta^*, \gamma, c >$. Here, Θ is the attacker's state space. $\theta \in \Theta$ represents the victim's policy and θ^* represents the target policy designed by the attacker. $u \in U$ is the attack action which represents changes to environment hyper-parameters. Here, the changes of hyper-parameters affect how the environment responds to the victim's actions. In other words, the attacker's aggregate actions $\{u_1, \ldots, u_i\}$ manipulate the victim's transition probabilities $T(s'|s, a, u_{1:i})$ (refer to victim-level MDP). u_0 means that the environment is not changed by the attacker. $F : \Theta \times U \times \Theta \to [0, 1]$ is the probabilistic state transition function of the attacker. $F(\theta'|\theta, u)$ represents how the victim's policy π_θ will update in the training environment poisoned by u. γ is the discount factor. $c : \Theta \times U \to \mathbb{R}$ is the attacker's cost caused by the victim's actual policies and manipulated environment dynamics.

The attack objective is to force the victim to learn the target policy with minimized changes to the environment dynamics. Specifically, the attacker aims to minimize the deviation between the victim's policy θ with the target one θ^*, as well as minimizing the deviation between the tweaked environment $T_{u_{1:i}}$ with the default one T_{u_0}. Thus, the attack cost c_i at attack epoch i is designed as:

$$
\begin{aligned}
c_i(\theta, u) &= D_{KLR}(P_i||P^*) \\
&s.t. \\
D_{KLR}(P_i||P^*) &= \sum_{s,a} q_i(s)\pi_{\theta_i}(a|s) D_i^{KL}(s,a) \\
D_i^{KL}(s,a) &= \sum_{s',a'} P_i(s',a'|s,a) \log \frac{P_i(s',a'|s,a)}{P^*(s',a'|s,a)} \\
P_i(s',a'|s,a) &= T_{u_{1:i}}(s'|s,a)\pi_{\theta_i}(a'|s') \\
P^*(s',a'|s,a) &= T_{u_0}(s'|s,a)\pi_{\theta^*}(a'|s') \\
q_i(s') &= \sum_{s,a} q_i(s)\pi_{\theta_i}(a|s) T_{u_{1:i}}(s'|s,a)
\end{aligned}
\tag{1}
$$

Here, $q_i(s)$ is the stationary state distribution of the victim's environment MDP. $P_i(s', a'|s, a)$ is a stochastic process under victim's policy $\pi_{\theta_i}(a|s)$ in environment $T_{u_{1:i}}(s'|s, a)$. $P^*(s', a'|s, a)$ is the ideal stochastic process desired by the attacker, which is determined by the target policy $\pi_{\theta^*}(a'|s')$ and the default environment dynamics $T_{u_0}(s'|s, a)$.

The attacker's optimization problem is to minimize the cumulative discounted cost $C = \sum_{i=1}^{\infty} \gamma_a^i c_i$, where i is the attack epoch. Based on the definition of attack cost as Equation 2, the knowledge of stationary state distribution $q_i(s)$ and transition probabilities $T(s'|s, a)$ are necessary to calculate the attack cost c_i. Therefore, the environment dynamics model should be white-box to the attacker.

4 PROBLEM STATEMENT

We study environment-poisoning attack in *double* black-box settings: 1) no prior knowledge on the dynamics model of the victim's training environment; 2) no information about the victim's learning algorithm and policy model. We assume the attacker can tweak the hyper-parameters of the training environments, such as the road friction in self-driving simulations. Additionally, we assume the attacker can fully observe the victim's behaviour (i.e., state and action trajectories).

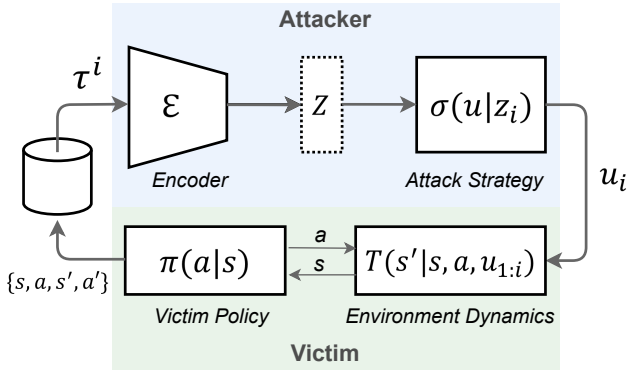

Figure 3: Attack illustration in double-blackbox settings

As Figure 3, the attack procedure consists of two stages: first to represent the victim's stochastic process based on its trajectories; and then to decide attack actions responding to the representation information.

Accordingly, the training of attack strategy includes two steps: 1) latent representation learning and 2) attack strategy learning. First, learning latent representation is to infer the environment dynamics information $T_{u_{1:i}}(s'|s, a)$ from transitions samples $\{s, a, s'\}$, as well as the victim's policy feature $\pi_\theta(a'|s')$ from state-action pairs $\{s', a'\}$. The learned representation embedding z is the input to the attack strategy. Second, for attack strategy learning, the objective is to minimize the cumulative discounted attack costs $C = \sum_{i=1}^{\infty} \gamma_a^i c_i$. Here, the attack cost c_i at each attack epoch is defined as:

$$
\begin{aligned}
c_i(\theta, u) &= \Delta(P_i||P^*) \\
&s.t. \\
P_i(s',a'|s,a) &= T_{u_{1:i}}(s'|s,a)\pi_{\theta_i}(a'|s') \\
P^*(s',a'|s,a) &= T_{u_0}(s'|s,a)\pi_{\theta^*}(a'|s')
\end{aligned}
\tag{2}
$$

In double black-box settings, both $T_{u_{1:i}}(s'|s, a)$ and $\pi_{\theta_i}(a'|s')$ are unknown to the attacker. As a result, the attack cost c_i can not be computed directly. To solve this problem, we utilize the latent representation z of the victim's stochastic process $P(s', a'|s, a)$ and approximate the attack cost in the latent space.

5 METHODOLOGY

In this section, we present the proposed approach to learn the attack strategy in *double* black-box settings. As described in Section 4, the training of the attack strategy consists of two parts: latent representation learning and attack strategy learning. Specifically, the attacker first learns a latent representation that captures the combined information about the victim's environment dynamics and policy from its trajectory samples. Then, responding to the representation information, the attacker learns an attack strategy that minimizes the cumulative attack cost measured in the latent space.

5.1 Latent Representation Learning

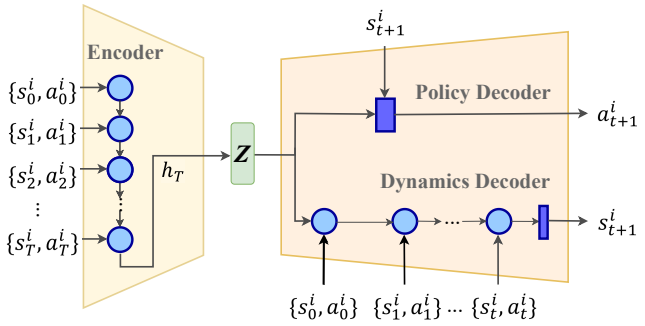

Figure 4: Design of autoEncoder: Encoder and Dual Decoder neural network

In double black-box settings, the attacker has no knowledge on the victim's environment-dynamics model and policy model, instead, the attacker can fully observe the victim's trajectory τ. The trajectory collected in attack epoch i is represented as $\tau^i = \{s_0^i, a_0^i, s_1^i, a_1^i, ..., s_T^i, a_T^i\}$. The attacker leverages the trajectory to learn z^i which is the latent representation of stochastic process $P_i(s', a'|s, a) = T_{u_{1:i}}(s'|s, a) \times \pi_{\theta_i}(a'|s, a)$.

As Figure 4, we introduce an autoencoder network consisting of one encoder network $\mathcal{E}$ and dual decoder networks $\{\mathcal{D}^{\pi}, \mathcal{D}^{T}\}$. More specifically, the encoder $\mathcal{E}$ models $f_{\phi}(z^i|\tau^i)$, which embeds the trajectory τ^i and generates the latent embedding z^i. The encoder $\mathcal{E}$ is designed as a Long Short Term Memory (LSTM) network to learn long-term dependencies in the trajectory. Then, to learn the encoder $\mathcal{E}$, we interpret the embedding z^i using two decoders: a policy decoder $\mathcal{D}^{\pi}$ and a dynamics decoder $\mathcal{D}^{T}$. Here, the policy decoder $\mathcal{D}^{\pi}$ is a multilayer perceptron (MLP) network. It models $f_{\omega}(a^i|s^i, z^i)$, which maps the victim's state s and the embedding z^i to the distribution over the victim's actions. The dynamics decoder $\mathcal{D}^{T}$ is a LSTM network which models $f_{\eta}(s^i_{t+1}|s^i_t, a^i_t, z^i)$. It conditions on the embedding z^i and the state-action pair $\{s^i_t, a^i_t\}$ to predict the next state s^i_{t+1}. To train the autoencoder network, we maximize the following negative cross-entropy objective:

$$\mathbb{E}_{\hat{\tau},\tau}\left[\sum_{\langle s,a,s',a'\rangle\sim\tau} \log f_{\omega}(a'|s', f_{\phi}(\hat{\tau})) + \log f_{\eta}(s'|s, a, f_{\phi}(\hat{\tau}))\right]$$

Here, $\hat{\tau}$ and τ are two different trajectories in the same attack epoch to perform cross training.

5.2 Attack Strategy Learning

With the latent representation z of the victim's stochastic process $P(s', a'|s, a)$, the attacker obtains the combined feature about the victim's policy $\pi(a'|s')$ and its environment dynamics $T(s'|s, a)$. Similarly, the attacker can also acquire the latent representation z^* of the ideal stochastic process $P^*(s', a'|s, a)$. To measure the attack cost which is denoted as the difference between $P(s', a'|s, a)$ and $P^*(s', a'|s, a)$, we approximate the cost using the embedding z and z^* in the latent space. We use Cosine Similarity to measure distance between z and z^*, and denote the attack cost c_i as:

$$c_i(z^i, z^*) = 1 - \frac{z^i \cdot z^*}{\|z^i\| \, \|z^*\|} \tag{3}$$

The training of the attack strategy is shown in Algorithm 1. Here, Line 7 $\sim$ 8 describes that the attacker poisons the environment dynamics responding to the latent representation. Line 10 $\sim$ 11 means that the attacker trains its Encoder-Decoder Network using the victim's latest trajectories. Line 12 shows that the attacker acquires the latent representation of the victim's stochastic process and Line 13 shows the attack cost is computed in the latent space. Then, the attacker's experience transitions $\{z, u, z', c\}$ are saved in the replay buffer $\mathcal{B}$ for the attack strategy updating, as Line 14 $\sim$ 15. When the victim's trajectory matches the target trajectory, the attack is successful.

Algorithm 1 Training of Attack Strategy

1: **Input**:
   τ^*: target trajectory designed by attacker
   τ^0: initial trajectory of victim
2: **Output**:
   σ: attack strategy network
3: **for all** *episode_num* **do**
4: reset training environment and revert victim
5: $z^0 \leftarrow \mathcal{E}(\tau^0)$: latent representation of τ^0
6: **for** i=1 to I_{max} **do**
7: $u_i \leftarrow \sigma(z_{i-1})$: attacker choose attack action
8: $T_i \leftarrow T_{u_{1:i}}$: attacker poison environment dynamics
9: victim learns policy in environment T_i
10: $\tau^i \rightarrow \mathcal{M}$: attacker collects victim's trajectory
11: encoder $\mathcal{E}$ and decoder $\mathcal{D}^{\pi,T}$ are trained using τ^i, τ^*
12: $z^i \leftarrow \mathcal{E}(\tau^i)$: attacker obtains latent representation of τ^i
13: $c_i(z^i, z^*)$: attacker computes attack cost
14: $\mathcal{B} \leftarrow \{z_{i-1}, u_i, z_i, c_i\}$: transitions is saved to replay buffer
15: attack strategy network σ is updated
16: **if** $\tau^i == \tau^*$ **then**
17: $attackDone$ = 1: go to the next episode
18: **end if**
19: **end for**
20: **end for**

6 EXPERIMENT

In this section, we show preliminary experimental results to show the proposed approach is effective to manipulate a black-box RL agent's policy in a black-box training environment.

6.1 Experiment Settings

6.1.1 Experiment Environment. We adopt the same experiment environment, 3D Grid World, as TEPA [12]. The 3D grid world is proposed in [8] to simulate a mountain or rugged terrain. As shown in Figure 5, there are associated elevations among cells in the 3D grid world. The success of moving from one cell to the neighboring one is proportional to the relative elevation between these two cells. Thus, changing elevation can affect how the environment responds to the agent's actions, which is a mechanism to modify the environment dynamics. The 3D grid world is the victim agent's environment. The victim aims to find an optimal path from the start cell to the destination. At each time step, the agent can move in one of the four cardinal directions with the reward −1.

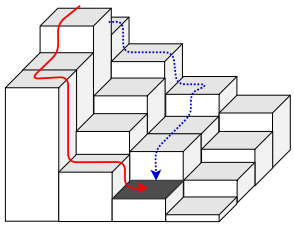

Figure 5: 3D Grid World: the shallow cell is the goal state. The blue line is the victim's optimal path while the red line is the target path designed by the attacker

6.1.2 Implementation. The attack strategy is represented by a multi-layer neural network as INPUT(10)-FC(400)-ReLU-FC(300)-ReLU-FC(16). The attack strategy is trained using deep RL learning algorithms, Deep Deterministic Policy Gradient (DDPG) [4]. Additionally, the victim agent uses Q-learning as its algorithm. The parameter configurations of Q-learning algorithm is: $\epsilon = 0.1, \gamma = 1.0, \alpha = 0.1$.

6.1.3 Measurement. The attack is regarded as successful if the victim agent chooses desired actions a^* in target states s^*. *Attack Success Rate* is denoted as the percentage of the target state set that has been attacked successfully. In this experiment, we measure the attack performance using the *attack success rate* during the victim agent's learning process.

6.1.4 Baseline. We choose TEPA [12] as our baseline. As described in Section 3, TEPA is effective to attack black-box victims due to the *transferability* of one poisoned training environment. However, TEPA is only effective in the white-box environment because its attack cost computations require state transition probabilities and stationary state distributions as Equation 2. In this experiment, we evaluate the attack performance in *double* black-box settings with the comparison of TEPA.

6.2 Experiment Results and Discussion

Attack Performance Evaluation. To evaluate the proposed attack approach in the double black-box setting, we compare the attack performance with TEPA [12]. Under double black-box settings, the attacker knows nothing about the victim's environment-dynamics model and policy model. The attack strategy is thereby trained based on the victim's experienced trajectories. We evaluate our attack strategy and TEPA strategy on 10 victim agents, respectively. As shown in Figure 6, the x-axis represents timesteps of victim agent's learning process, and the y-axis describes the evaluation performance measured by attack success rate. Figure 6 shows that the attack success rate of the proposed approach is quite similar to the baseline TEPA. This result clearly shows that the proposed attack approach successfully forces the black-box victim to learn the target behaviour in a black-box environment. Furthermore, this also indicates our proposed solution which approximates attack costs, i.e., computing the deviation in latent space as described in Equation 3, is effective.

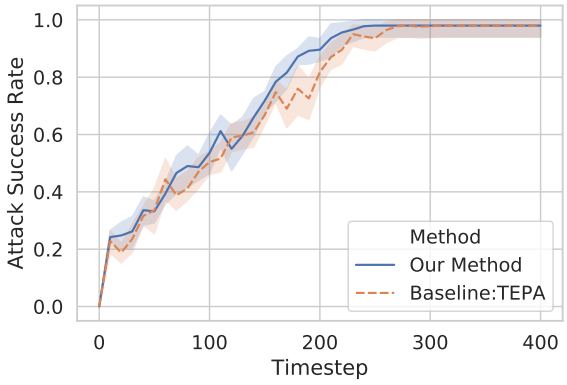

Figure 6: Attack performance with the comparison of TEPA

Generalization Evaluation. We evaluate the generalization of the attack strategy on victims adopting different learning algorithms. In this experiment, the attack strategy is trained on the Q-learning victims and evaluated on three kinds of victims adopting learning algorithms SARSA, Monto Carlo and Q-learning [10]. Figure 7 shows that the strategy successfully attacks these three types of victims. It indicates that the attack strategy, which is learned on one black-box victim in the unknown training environment, is also effective for different black-box victims in the same training environment. Therefore, our proposed attack strategy is insensitive to victims' learning algorithms and thereby is effective for different algorithm-based victims.

The baseline TEPA learns an attack strategy on a white-box proxy agent and then transfers the strategy to a black-box victim. Our work removes the constraints of using a transfer-learning capable white-box agent, which trains the attack strategy directly on the black-box victim. This attack strategy is generally effective for various black-box RL agents, as shown in Figure 7.

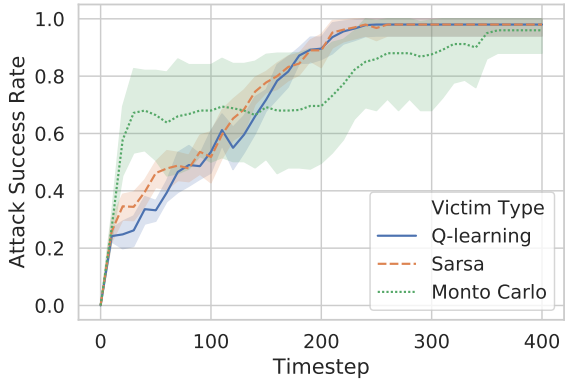

Figure 7: Attack performance on different types of victims

7 CONCLUSION AND FUTURE WORK

In this work, we study the training-time attack on Reinforcement Learning in *double* black-box setting, where both victim's environment dynamics and learning algorithm are unknown. The attack objective is to stealthily force an unknown agent to learn a target policy via poisoning the unknown environment dynamics. In double black-box settings, the attacker learns the latent representation of the victim's policy features and environment-dynamics features from the victim's trajectory samples. Based on the latent representation, the attacker learns an adaptive strategy by approximating attack costs in the latent space. In the preliminary experiment, we evaluate the attack performance of the proposed approach with the comparison of the baseline TEPA. Results show that the attack strategy performs well and is general for different black-box agents.

This is one work in process. The proposed approach is only evaluated in one simplistic experiment domain. In the ongoing work, we are implementing the attack approach in more sophisticated simulations, to demonstrate the scalability of our proposed approach.

8 ACKNOWLEDGEMENTS

This work was supported by SCSE NTU SUG "Choice Manipulation and Security Games". The computational work for this paper was fully performed on resources of the National Supercomputing Centre (NSCC), Singapore (https://www.nscc.sg).

REFERENCES

[1] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. 2017. Adversarial Attacks on Neural Network Policies. In *Proceedings of the 5th International Conference on Learning Representations (Workshop)*. ICLR, Vancouver, BC, Canada.

[2] Matthew Inkawhich, Yiran Chen, and Hai Li. 2019. Snooping Attacks on Deep Reinforcement Learning. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*. IFAAMS, Auckland, New Zealand, 557–565.

[3] Jernej Kos and Dawn Song. 2017. Delving into Adversarial Attacks on Deep Policies. In *Proceedings of the 5th International Conference on Learning Representations (Workshop)*. ICLR, Vancouver, BC, Canada.

[4] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. In *Proceedings of the 4th International Conference on Learning Representations*.

[5] Yenchen Lin, Zhangwei Hong, Yuan-Hong Liao, Mengli Shih, Mingyu Liu, and Min Sun. 2017. Tactics of Adversarial Attack on Deep Reinforcement Learning Agents. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. IJCAI, Melbourne, Australia, 3756–3762.

[6] Yuzhe Ma, Xuezhou Zhang, Wen Sun, and Jerry Zhu. 2019. Policy Poisoning in Batch Reinforcement Learning and Control. In *Proceedings of the 33th Conference on Neural Information Processing Systems*. ACM, Vancouver, Canada, 14570–14580.

[7] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in Machine Learning: from Phenomena to Black-box Attacks using Adversarial Samples.

[8] Zinovi Rabinovich, Lachlan Dufton, Kate Larson, and Nick Jennings. 2010. Cultivating desired behaviour: Policy teaching via environment-dynamics tweaks. In *Proceedings of the 10th International Conference on Autonomous Agents and MultiAgent Systems*. IFAAMS, Toronto, Canada, 1097–1104.

[9] Amin Rakhsha, Goran Radanovic, Rati Devidze, Xiaojin Zhu, and Adish Singla. 2020. Policy Teaching via Environment Poisoning: Training-time Adversarial Attacks against Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, Vienna, Austria, 7974–7984.

[10] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction*. MIT press, Cambridge, Massachusetts, USA.

[11] Edgar Tretschk, Seong Joon Oh, and Mario Fritz. 2018. Sequential Attacks on Agents for Long-Term Adversarial Goals. In *Proceedings of the ACM Computer Science in Cars Symposium*. ACM, Munich, Germany.

[12] Hang Xu, Rundong Wang, Lev Raizman, and Zinovi 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.

[13] Xuezhou Zhang, Yuzhe Ma, Adish Singla, and Xiaojin Zhu. 2020. Adaptive Reward-Poisoning Attacks against Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, Vienna, Austria, 11225–11234.