Poisoning Offline Reinforcement Learning to Promote Distributional Shift in Online Finetuning

Zishun Yu[1] **Siteng Kang**[1] **Xinhua Zhang**[1]

[1]Department of Computer Science, University of Illinois Chicago, Chicago, IL, USA

Abstract

Offline-to-online reinforcement learning has recently been shown effective in reducing the online sample complexity by first training from offline collected data. However, this additional data source may also invite new poisoning attacks that target offline training. In this work, we reveal such vulnerabilities by proposing a novel data poisoning attack method, which is stealthy in the sense that the performance during the offline training remains intact, but the online fine-tuning stage will suffer a significant performance drop. Our method leverages the techniques from bi-level optimization to promote the distribution shift under offline-to-online reinforcement learning. Experiments on four environments confirm the satisfaction of the new stealthiness requirement, and can be effective in attacking with only a small budget and without having white-box access to the victim model.

1 INTRODUCTION

Offline reinforcement learning (RL) has recently opened up new opportunities of leveraging offline batch data to improve the RL algorithms, significantly reducing the online sample complexity of interacting with the environment (Levine et al., 2020). It is particularly valuable for many applications where directly applying an automated policy can be dangerous, expensive, or unethical. For example, educational assistants, autonomous driving, and healthcare.

However, due to the limited coverage of offline data or the suboptimality of the demonstrator (Fu et al., 2020), a purely offline trained model is generally not effective when deployed online, and a common wisdom is to fine-tune it via additional online interactions, whose sample complexity is expected to be saved thanks to the initialization from offline training.

Interestingly, such a direct offline-to-online transfer (O2O) is often plagued with catastrophic performance drop at online transfer, which poses safety challenges for the real system such as driving and therapy. This is primarily due to the distributional shift of the state (Fujimoto et al., 2019; Kumar et al., 2019; Fu et al., 2019; Kumar et al., 2020a), and the Q-value has not been well estimated for the state-actions lying outside the offline distribution (Farahmand et al., 2010; Munos, 2005).

Fortunately, previous research (Kumar et al., 2020b; Kostrikov et al., 2022; Lee et al., 2022; Nakamoto et al., 2023) shows that improved O2O RL methods can significantly control the distribution shift while retaining the online sample efficiency. Typical solutions include endowing conservatism on offline Q-function approximation (Kumar et al., 2020b; Nakamoto et al., 2023), or regularizing the divergence between the learned policy and the behavior policy (Nair et al., 2020). Lee et al. (2022) directly manipulates the replay buffer to ensure the data sampled from the buffer are likely on-policy, thereby avoiding the inaccurate approximation for OOD data. Instead of directly transferring the offline critic to online, Yu & Zhang (2023) introduces an addition step which aligns the critic with the actor upon the end of offline training, ensuring that the critic would not differ much from the actor, hence avoiding severe online policy change caused by poor value estimation. Zhang et al. (2022) and Wang et al. (2023) both consider training a set of policies rather than a single one, as such redundancy could avoid the risk of unrecoverable performance drop of a single policy, facilitated by their choice of policy selector.

There is still a long list O2O methods that emerged recently (Wagenmaker & Pacchiano, 2023; Chen & Wen, 2023; Mark et al., 2023; Lei et al., 2023, etc.). Among the aforementioned works, surgery on the Q-function is one of the most prevalent principles to address O2O. As O2O heavily depends on a “well-behaved” Q-function, it also creates vulnerability in such scenarios, as one may manipulate Q-functions in a malicious way.

The key question we investigate in this paper is

> Are the O2O algorithms robust to reward poisoning on the offline batch data?

Since offline data often comes from crowd-sourcing or other third parties, it may carry malicious poisons that catastrophically damage the online fine-tuning while remaining stealthy by keeping the offline performance competitive.

In general, poison attack is performed on the training data, such that the models trained with it will perform poorly on the test scenarios. In O2O RL, the attacker may alter the state, action, or reward of the offline data. In this paper, we focus on poisoning of rewards, and aim to achieve two objectives:

- **Effectiveness**: after offline training on the poisoned data, the agent will suffer a catastrophic performance drop at the beginning of the online fine-tuning, compared with its (online) performance at the end of the offline training.
- **Stealthiness**: during the offline training, the performance as measured by interacting with the environment (but not using it to update the model) should be similar to that achieved by a clean trained agent. This is in addition to the standard ℓ_p norm constraints on the magnitude of reward modification.

These definitions of stealthiness and effectiveness are particular realistic. As O2O RL is a two-phase learning scheme, attacks that aim to undermine the offline performance may be of less risk to the system because the victim can detect the low performance of the offline model. However, an attacker that is stealthy offline but effective online could be more surprising and harmful. Therefore, understanding such vulnerability of O2O RL is essential towards robust O2O transfer.

Our contribution is to achieve these goals, revealing the vulnerability of O2O RL to data poisoning attack. Our innovations can be summarized as follows:

- We propose the first poisoning attack on O2O RL that promotes the distributional shift and Q-function overestimation.
- We achieve the poisoning through an efficient bi-level optimization technique.
- Our approach requires no access to the victim agent or the online environment.

We applied our poisoner to Frozen Lake and three locomotion environments from D4RL (Fu et al., 2020). The stealthiness is clearly verified, and it is shown more effective in compromising online fine-tuning performance than other baselines.

2 RELATED WORK

The vulnerability to various types of attacks has been well studied in supervised learning field. Evasion attack (Goodfellow et al., 2015) assumes the attacker can manipulate testing inputs after the victim model is trained. Data poisoning attack, on the other hand, is performed on the training inputs. The attacker may insert (Chen et al., 2017) or modify the training inputs (Biggio et al., 2012; Shafahi et al., 2018) to undermine the performance of the trained victim model.

Attacks in Online RL Huang et al. (2017) and Sun et al. (2020) studied evasion attacks in online RL that apply modifications on the "critical point" of the input state to fool a pre-trained online RL agent. Xu et al. (2021) proposed a data poisoning attack in online RL by using a malicious RL agent as an attacker to find the minimal change on the environment dynamic in order for the victim agent to choose the target action on a target state. Zhang et al. (2020) achieved a similar goal by altering only the training reward. They performed the attack by constructing a target Q table such that at the target state, the Q value is maximized at the target action chosen by the attacker. The attacker then modifies the training reward so the victim agent will learn such a Q table. Rakhsha et al. (2021) further extended the attack to a black-box setting by additionally explore the environment uniformly. They then used the explored data to simulate the victim agent and perform the poisoning attack.

Attacks in Offline RL Ma et al. (2019) is one of the first works that try to perform targeted data poisoning attack in offline RL. They model the attack as a convex optimization problem on the training reward and forced the victim to learn a targeted policy. Gong et al. (2022) proposed the first backdoor attack in offline RL by altering the training observations. After training on the poisoned data, the victim model will perform normally on clean testing data, while choosing the low-reward actions on observations with predesigned triggers. More recently, Wu et al. (2023) designed a data poising attack specifically on multi-agent RL so that the victim agents will learn a policy targeted by the attacker.

However, existing attacks in online RL require access to online environment and are therefore infeasible in many practical scenarios. On the other hand, offline RL attacks leads to poor performance during the validation and can be detected before online fine-tuning. None of them can be directly applied to O2O RL settings to achieve our objectives.

3 PROBLEM SETUP

In this section, we set up the three participants in the O2O poisoning problem: the environment, the victim agent, and the attacker.

3.1 PRELIMINARY

We formulate the **RL process** via the standard Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mu_0)$. Here $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in [0, 1)$ is the discount factor, and $\mu_0 : \mathcal{S} \to \mathbb{R}$ is the initial state distribution.

For the **victim agent**, we define its policy $\pi(a|s)$ as a probability distribution of taking action a at state s. The agent's goal is to find the optimal policy that maximizes the expected return $\pi^* = \arg\max_\pi \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$.

In offline RL, there is a batch of transitions $\mathcal{D} = \{(s, a, r, s')\}$ that are obtained by applying an unknown behavior policy on the environment. In the O2O literature, it has been shown that offline conservative Q-learning (CQL, Kumar et al., 2020b) followed by an off-policy algorithm—often soft actor critic (SAC, Haarnoja et al., 2018)—for online fine-tuning is a strong yet simple baseline (Lee et al., 2022; Yu & Zhang, 2023). Intuitively, it is effective because CQL provides a good Q-function initialization that suppresses Q-values for out-of-distribution (OOD) actions, avoiding poor online exploration led by false over-estimation. And using off-policy algorithm online allows faster learning as the Q-function is now freed from conservative constraints/penalties.

As CQL+SAC has served as a common baseline in O2O literature (Lee et al., 2022; Nakamoto et al., 2023; Yu & Zhang, 2023), we used the same CQL+SAC scheme as our victim O2O agent for most of the experiments except the one on the FrozenLake environment, whose action space is discrete. This is because the discrete variant of CQL does not parameterize its policy but induces it from the Q-function via a soft-max. In contrast, the discrete SAC parameterizes its policy via a network, making it infeasible to perform O2O transfer due to the lack of offline policy network. As a result, we used DoubleDQN (Van Hasselt et al., 2016) as the online algorithm in the FrozenLake environment.

Soft Actor Critic SAC is an actor-critic algorithm based on the maximum entropy framework. Akin to canonical actor-critic, it includes actor update and critic update, as shown in (2) and (1), respectively. In particular, we employed SAC-v2 (Haarnoja et al., 2018), an alternative implementation that automatically adjusts the entropy of the policy, via the Lagrangian dual formulation, where the Lagrangian multiplier is often called the temperature α, and its update rule is given in (3).

$$\mathcal{L}_Q^{\text{SAC}}(\psi, \mathcal{D}) := \mathop{\mathbb{E}}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q_\psi(s, a) - y(r, s')\right)^2\right]$$
$$y(r, s') := r + \gamma \mathop{\mathbb{E}}_{a'\sim\pi_\theta(s')}\left[Q_{\bar{\psi}}(s', a') - \alpha \log \pi_\theta(a'|s')\right] \tag{1}$$

$$\mathcal{L}_\pi^{\text{SAC}}(\theta, \mathcal{D}) := \mathop{\mathbb{E}}_{s\sim\mathcal{D}} \mathop{\mathbb{E}}_{a\sim\pi_\theta(s)}\left[\alpha \log \pi_\theta(a|s) - Q_\psi(s, a)\right] \tag{2}$$

$$\mathcal{L}_{\text{temp}}^{\text{SAC}}(\alpha, \mathcal{D}) := -\alpha \mathop{\mathbb{E}}_{s\sim\mathcal{D}} \mathop{\mathbb{E}}_{a\sim\pi_\theta(s)}\left[\log \pi_\theta(a|s) - \bar{\mathcal{H}}\right]. \tag{3}$$

Here the expectation $\mathbb{E}_{a\sim\pi_\theta(s)}$ could be directly evaluated for discrete action spaces and be stochastically approximated for continuous action spaces.

The actor update (2) aims to maximize the Q-values hence maximizing the cumulative rewards alongside the policy's entropy. The critic update (1) aims to find a better soft Q-function approximation by minimizing the squared temporal difference error, where $\bar{\psi}$ stands for target network, a commonly used trick in RL literature to stabilize RL training. It can be often updated using the Polyak averaging (or exponential moving averaging), which is essentially $\bar{\psi} \leftarrow \tau\psi + (1 - \tau)\bar{\psi}$, where $\tau \in (0, 1)$ is a hyper-parameter that controls how fast the target network $\bar{\psi}$ evolves towards the current Q-network ψ. The temperature update (3) automatically tunes $\alpha > 0$ to ensure that the entropy of the policy is lower bounded by a target entropy $\bar{\mathcal{H}}$.

Distribution shift It has been well known that directly applying off-policy algorithms such as SAC to offline RL often leads to poor performance as off-policy RL methods can have arbitrarily poor value estimations (especially over-estimations) for OOD states/actions (Lee et al., 2022; Zhang et al., 2022; Yu & Zhang, 2023; Nakamoto et al., 2023). We next elaborate on the corresponding *distribution shift* issue, which will serve as the key motivation of our attacking algorithm.

Firstly, at training time, the target value for Bellman backups of critic update in (1) uses actions a' sampled from the learned policy π_θ, while the Q function was trained only on actions produced by the offline data under the behavior policy (the expectation over $\mathcal{D}$ in (1)). As a result, the offline learned Q function typically over-estimates the value of $Q(s, a)$ for an OOD action a, i.e., when a is never applied at state s in the offline data. A similar issue also plagues the actor update in (2), where Q_ψ is evaluated on $a \sim \pi_\theta(s)$.

Secondly, during online fine-tuning, the agent runs into an OOD state and the bootstrap error wipes out the offline learned policy that performed well.

A great deal of effort has been devoted to this problem in O2O RL, by, e.g., regularizing the Q-function (Kumar et al., 2020b), importance sampling (Laroche et al., 2019), and behavior cloning regularization (Kostrikov et al., 2022).

Conservative Q-Learning CQL is a popular choice for offline and O2O RL that combats the distribution shift issue. The central idea is to regularize the Q-values of actions that are not observed in the offline dataset. Such regularity avoids over-estimations of OOD actions that may have a low return in the real environment. We also provide an illustration of such conservative estimation in our toy example in Figure 1. Specifically, we consider a commonly used variant of CQL,

Algorithm 1 O2O: CQL (offline) + SAC (online)

```
1: Input: offline dataset $\mathcal{D} = \{(s, a, r, s')\}$
2: # offline training phase with CQL.
3: initialize CQL parameters $\theta, \psi, \bar{\psi}$
4: for number of offline iterations do
5:     sample mini-batch from offline dataset $\mathcal{D}$
6:     update $\psi$ with (4)
7:     update $\theta$ with (5)
8:     $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$
9: end for
10: # online training phase with SAC.
11: load parameters $\theta, \psi, \bar{\psi}$ for SAC
12: initialize temperature $\alpha$ for SAC
13: for number of online iterations do
14:     # environmental step
15:     $a \sim \pi_\theta(a|s), r \sim \mathcal{R}(s, a), s' \sim \mathcal{P}(s'|s, a)$
16:     $\mathcal{D} \leftarrow \mathcal{D} \bigcup \{(s, a, r, s')\}$
17:     # gradient step
18:     sample mini-batch from online buffer $\mathcal{D}$
19:     update $\psi$ with (1)
20:     update $\theta$ with (2)
21:     update $\alpha$ with (3)
22:     $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$
23: end for
24: Output: network parameters $\psi, \theta$
```

namely CQL($\mathcal{H}$), whose regularizer is given in (4) along with the squared loss. In addition, CQL($\mathcal{H}$) follows (5) to update policy for continuous action spaces, while in discrete space the policy is induced greedily from Q_ψ.

$$\mathcal{L}_Q^{\text{CQL}}(\psi, \mathcal{D}) := \mathop{\mathbb{E}}_{(s,a,r,s')\sim\mathcal{D}} \left[(Q_\psi(s, a) - y(r, s'))^2\right]$$
$$+ \underbrace{\lambda \mathop{\mathbb{E}}_{s\sim\mathcal{D}} \left[\log\textstyle\sum_a \exp(Q_\psi(s, a)) - \mathop{\mathbb{E}}_{a\sim\pi_\theta(s)}[Q_\psi(s, a)]\right]}_{=:\, J_Q^{\textbf{CQL}}(\psi,\mathcal{D})} \quad (4)$$

discrete: $y(r, s') := r + \gamma Q_{\bar{\psi}}(s', \text{argmax}_{a'} Q_\psi(s', a'))$

continuous: $y(r, s') := r + \gamma \mathop{\mathbb{E}}_{a'\sim\pi_\theta(s')}[Q_{\bar{\psi}}(s', a')]$

$$\mathcal{L}_\pi^{\text{CQL}}(\theta, \mathcal{D}) := \mathop{\mathbb{E}}_{s\sim\mathcal{D}} \mathop{\mathbb{E}}_{a\sim\pi_\theta(s)}[Q_\psi(s, a)]. \quad (5)$$

Here, the expectation $\mathbb{E}_{a\sim\pi(s)}$ and the log-sum-exp $\log\sum_a \exp(Q_\psi(s, a))$ are tractable for discrete action spaces and can be stochastically approximated for continuous spaces.

The entire O2O algorithm that uses CQL for offline training and SAC for online fine-tuning is summarized in Algorithm 1.

3.2 A TOY EXAMPLE

We provide a toy bandit example in Figure 1 to demonstrate our motivation. The key idea of this example is that uniformly lifting the Q-values can achieve both stealthiness and effectiveness, because a uniform over-estimation would not change the policy, as demonstrated in Figure 1a; and it can promote online distributional shift, as shown in Figure 1b.

While the toy example simply assumes that the Q-function can be directly manipulated to achieve a uniform over-estimation, this is however infeasible in a poisoning attack setting. In Section 4, we show that one could achieve it by formulating it as a bi-level optimization.

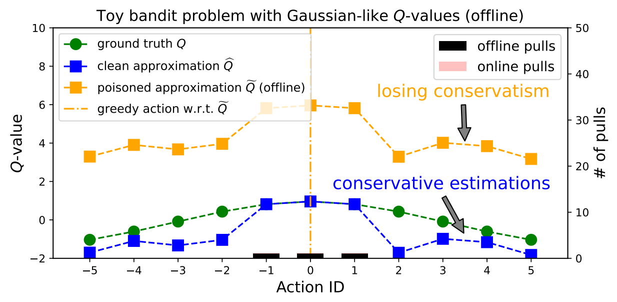
4 THE ATTACK ALGORITHM

We consider the vulnerability of O2O RL under data poisoning during offline training. Since the attacker is not allowed to perform any attack during the online fine-tuning phase, the victim will eventually recover from any offline attack given infinite online training resource. Thus, we set the attacker's goal to be such that the victim model, when fine-tuned online, suffers as much performance drop—both in magnitude and duration—at the *initial* phase as possible.

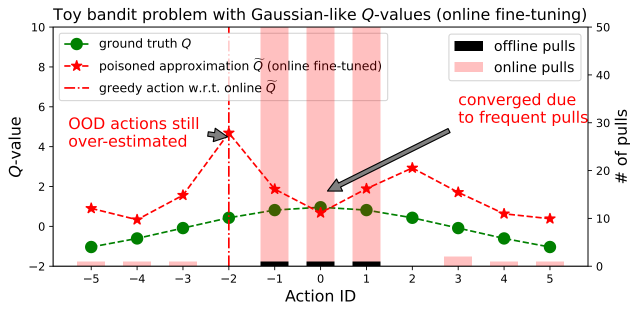
4.1 THE THREAT MODEL

Following the standard poisoning attack protocol, we assume that the victim may not access clean demonstrations during offline training. Key to our threat model is the requirement that the **victim model must retain good online performance when offline training concludes**, because otherwise the attack would be detected and the model it would be precluded from online fine-tuning. Here the "online performance" is evaluated by hypothetically applying the policy to an online environment, but without updating the policy (as opposed to online training). In reality, neither the agent nor the attacker can really run this evaluation. However, given that offline policy evaluation is notorious for its high variance, we define offline performance in this way, noting that its value is *not* used by either the agent's RL algorithm or the attacker's poisoning algorithm.

Although reward, state, and action are all feasible targets of poisoning on the offline batch data, we restrict our attention to reward because it is a single scalar and carries less structure than states and actions, hence allowing more stealthy poisoning. The attacker is not allowed to access the victim model, such as its policy network or value functions. Following the common practice such as Witche's Brew (Geiping et al., 2021) and continual input-aware poisoning (Kang et al., 2023), the attacker may internally train a *surrogate* RL agent and queries it to construct the poisons.



(a) Offline phase: Suppose Q, $\hat{Q}$ and $\tilde{Q}$ stands for the ground truth Q-function, CQL approximation without being poisoned, and an uniformly poisoned Q-function, respectively. The bar plot illustrates the number of observed data for the corresponding action. It can be observed that $\hat{Q}$ well approximates in-distribution actions $\{-1, 0, 1\}$ and under-estimates OOD actions as expected. The poisoned $\tilde{Q}$ is stealthy as a uniform increase does not change the policy but breaks the conservatism which would lead to poor online performance.



(b) Online phase: Suppose we initialize an online agent with the offline poisoned $\tilde{Q}$. After some online interactions, which are clean as the poisoning is only done in offline phase, one could observe that the majority of iterations are in-distribution because they have higher $\tilde{Q}$-values at the beginning. However, by providing many clean data for in-distribution actions, their $\tilde{Q}$ estimations converge to ground truth. As a result, the $\tilde{Q}$ estimations become dominant because they were updated less frequently, hence promoting online distributional shift.

Figure 1: A toy bandit example, with 11 actions and Gaussian-like reward function, to demonstrate the intuition that maximizing Q-values (uniformly) can achieve both stealthiness and effectiveness.

In addition to the aforementioned stealthiness constraints, we also impose the standard ℓ_p norm constraints on the reward perturbations. For example, the ℓ_0 norm constraints specifying how many offline transitions can be perturbed, and ℓ_1 norm constraints on the total or average amount of perturbation. For a vector $\mathbf{x}$, its ℓ_1 norm is $\|\mathbf{x}\|_1 := \sum_i |x_i|$, and its ℓ_∞ norm is $\|\mathbf{x}\|_\infty := \max_i |x_i|$.

4.2 THE POISONING ALGORITHM

Due to the stealthy requirement, the poisoning algorithms for offline RL such as Gong et al. (2022) cannot serve our purpose as it would lead to poor online performance for the offline trained model. Our inspiration originates from the distribution shift phenomenon, which shows that over-estimation of the Q-function will lead to poor online performance, while keeping the performance during offline training competitive. Thus, we seek to poison the reward by promoting the resulting Q values at OOD actions, hence maximally exacerbate the over-estimation problem.

We first uniformly sample from $\mathcal{D}$ a small portion of candidate transitions to be poisoned $\mathbf{c}^p := \{(s^p, a^p, \mathbf{r}^p, s'^p)\}$. The cardinality depends on the ℓ_0 norm and our experiment uses 2% of the size of $\mathcal{D}$. Then we perturb the reward on these transitions to construct a poisoned buffer $\mathcal{D}^p = \{(s^p, a^p, \mathbf{r}^p + \delta_r, s'^p)\}$. Finally we combine it with the rest of clean transitions to construct the poisoned training dataset

$$\mathcal{D}^t := \mathcal{D}^p \cup (\mathcal{D} \setminus \mathbf{c}^p). \tag{6}$$

Conceptually, our poisoner solves the following constrained bi-level optimization for δ_r:

$$\max_{\delta_r} \quad \mathop{\mathbb{E}}_{s\sim\mathcal{D}} \mathop{\mathbb{E}}_{a} [Q_{\psi^*}(s,a)] - \beta J_Q^{\mathbf{CQL}}(\psi^*, \mathcal{D}) \tag{7}$$

$$\text{s.t.} \quad \|\delta_r\|_1 / |\mathcal{D}^p| \leq \epsilon_1 \quad \text{and} \quad \|\delta_r\|_\infty \leq \epsilon_\infty \tag{8}$$

$$\psi^*, \theta^* \leftarrow \textbf{Victim-Offline-RL}(\mathcal{D}^t). \tag{9}$$

Note in the first term of the outer objective, we do not require $\mathbf{a}$ to be from the offline data, i.e., it does not have to be what was taken at state $\mathbf{s}$. This exactly serves our purpose of simulating OOD actions, and promoting their Q values. It is similar in spirit to the log-sum-exp term in (4). When the action space is discrete and finite, it is natural to use a uniform distribution here. When the action space is continuous, we endow a proper distribution that also allows efficient sampling, e.g., uniform for bounded spaces or Gaussian for unbounded spaces.

Furthermore, we included the conservative training regularizer $J_Q^{\mathbf{CQL}}$ from CQL in (4). This places the offline performance on par with CQL, achieving the stealthiness requirement.

In practice, the expectations on $\mathbf{s}$ and $\mathbf{a}$ can both be evaluated by sampling in (7). However, different from common training objectives, our optimization problem is bi-level, which poses additional computational cost and becomes unstable once the objective is evaluated stochastically. As a result, we resort to pre-sampling $\mathbf{s}$ and $\mathbf{a}$, and fixing them throughout the entire optimization process.

In particular, we define $\mathcal{D}^r$ as the set of (state, action) where the states encompass those from $\mathcal{D}^p$, but the actions are randomly generated. Note that neither the states nor the actions were poisoned in $\mathcal{D}^p$. We could replace $\mathcal{D}^p$ with another random subset of $\mathcal{D}$, and we reuse it here just for convenience.

Algorithm 2 Update δ_r with IFT

```
1: Input: offline buffer $\mathcal{D} = \{(s, a, \mathbf{r}, s')\}$, poison $\delta_r$, surrogate critic parameters $\psi$, step size $\eta$
2: $v_1 \leftarrow \frac{\partial \mathcal{L}_{\delta_r}}{\partial \psi}|_{\delta_r,\psi}$, where $\mathcal{L}_{\delta_r}$ is the outer objective in (7)
3: $v_2 \leftarrow$ InverseHVP$(v_1, \frac{\partial \mathcal{L}_Q(\psi,\mathcal{D})}{\partial \psi})$ with $\mathcal{L}_Q$ from (4).
4: $v_3 \leftarrow \frac{\partial^2 \mathcal{L}_Q(\psi,\mathcal{D})}{\partial \delta_r \partial \psi} v_2$. In PyTorch, it can be implemented by $\mathtt{v_3} = \mathtt{grad}(\frac{\partial \mathcal{L}_\mathtt{Q}(\psi,\mathcal{D})}{\partial \delta_\mathtt{r}}, \psi, \mathtt{grad\_outputs} = \mathtt{v_2})$
5: Output: Updated $\delta_r = \delta_r + \eta v_3$ as (7) is maximization
```

To summarize, our poisoner solves

$$\max_{\delta_r} \quad \mathop{\mathbb{E}}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}^r} [Q_{\psi^*}(s, a)] - \beta J_Q^{\mathbf{CQL}}(\psi^*, \mathcal{D}^p) \tag{10}$$

$$\text{s.t.} \quad (8) \text{ and } (9). \tag{11}$$

Here we also changed the argument of $J_Q^{\mathbf{CQL}}$ from $\mathcal{D}$ in (7) to $\mathcal{D}^p$. This approximation is unbiased, and significantly accelerates the optimization as $\mathcal{D}^p$ is much smaller than $\mathcal{D}$.

4.3 SOLVING THE BI-LEVEL OPTIMIZATION

To solve (10), a key quantity needed is the derivative of the outer objective with respect to δ_r, which in turn needs the derivative of Q_{ψ^*} with respect to δ_r. This is challenging because their dependence is through an offline RL algorithm. The fundamental mathematical solution is the implicit function theorem (IFT), based on which a number of techniques with improved computational and spatial complexity have been widely used in previous works on hyper-parameter tuning (Bengio, 2000; Maclaurin et al., 2015; Shaban et al., 2019; Lorraine et al., 2020). Here, we utilize these techniques in a similar way as described in Algorithm 2, where instead of tuning the hyper-parameter, we update δ_r. In particular, we follow Lorraine et al. (2020) and approximate the Inverse Hessian Vector Product (HVP) by using the Neumann approximation.

Equipped with the gradient in δ_r, we could simply perform gradient based updates such as ADAM. However, this is very expensive because IFT-style algorithms require solving the inner offline RL to the optimal. For computational efficiency, we only run offline RL for a few steps in each iteration, and use the suboptimal ψ to update δ_r via Algorithm 2. The entire procedure is summarized in Algorithm 3, illustrating how the attacker generates the poison. We will refer to it as **O2O poisoner (O2OP)**.

It is noteworthy that the attacker does not require accessing the victim agent's model, neither the policy nor the value functions. Instead, it trains its own surrogate agent based on which the poison is constructed. Surrogate models are quite commonly used (Geiping et al., 2021; Kang et al., 2023; Souri et al., 2022; Cherepanova et al., 2021; Goldblum et al., 2023), and its effectiveness is far from trivial because RL is well known for high variance. With different seed and different mini-batches sampled, the surrogate agent can be quite different from the real agent, making it nontrivial for the learned poison to remain effective.

Algorithm 3 O2OP: Poison Generation via Surrogate Model

```
1: Input: clean offline dataset $\mathcal{D} = \{(s, a, \mathbf{r}, s')\}$
2: randomly pick a set of transitions $\mathbf{c}^p$ for poisoning
3: initialize surrogate CQL model parameters $\theta_p, \psi_p, \bar{\psi}_p$
4: initialize poisoned dataset $\mathcal{D}^p = \{(s^p, a^p, \mathbf{r}^p + \delta_r, s'^p)\}$
5: combine clean and poisoned dataset into the training dataset $\mathcal{D}^t = \mathcal{D}^p \bigcup \mathcal{D} \setminus \mathbf{c}^p$
6: for iter = 1 ... number of offline iterations do
7:     sample a mini-batch $\mathcal{B}$ from $\mathcal{D}^t$
8:     for each gradient step do
9:         Update $\psi, \bar{\psi}, \theta$ using $\mathcal{B}$ according to (4) and (5)
10:        if step = multiple of IFT frequency then
11:            Update $\delta_r$ via Algorithm 2 using $\psi$
12:        end if
13:    end for
14:    if iter = multiple of verification frequency then
15:        Perform verification with online environment
16:    end if
17: end for
18: Output: $\delta_r$
```

5 EXPERIMENTS

We now empirically verify that our proposed poisoner O2OP fulfills the aforementioned objectives. We tested on Frozen Lake, Hopper, Half Cheetah, and Walker 2D environments from the D4RL dataset (Fu et al., 2020). For each experiment, we used CQL for offline training, and SAC for online fine-tuning. SAC was not used for offline training because it performs poorly in online fine-tuning even without attack (Yu & Zhang, 2023). Following the common protocol, we repeated experiments on each environment with 5 random seeds, and then plotted the mean return from the 5 trials.

5.1 FROZEN LAKE: DISCRETE ENVIRONMENT

Frozen Lake is a discrete text environment. The environment consists a 4-by-4 or 8-by-8 grid, each location is labeled as road or hole. The bot starts from one location in the grid. Figure 2 visualized a typical 4-by-4 Frozen Lake environment. At each time step, the agent can observe the location of the bot, and choose to take an action among moving up, down, left, or right. The agent receives a reward of 1 if the bot gets to the goal, and receives 0 reward for reaching a road or hole.

We trained an offline discrete CQL agent for 100 epochs, with 500 steps in each epoch. The online discrete SAC model was trained for 50 epochs on clean online environ-

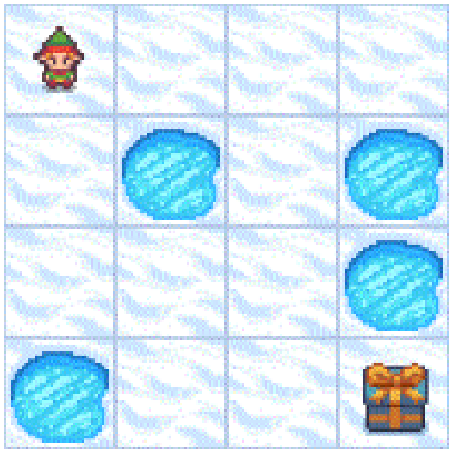


Figure 2: Visualization for Frozen Lake environment

ment, with a buffer carried over from their offline phase. For this environment, we included all offline transitions $\mathcal{D}$ in our candidate set $\mathcal{D}^p$, and tested $\epsilon_1 \in \{0.1, 0.02\}$ from (8), leaving ϵ_∞ unconstrained. O2OP first generated δ_r from a surrogate model as described in Algorithm 3, and used it to poison a new victim which was trained by CQL with a different initialization and mini-batch sampling seed.

Baselines Since there is no existing algorithm addressing our task, we adopted a uniform poisoner which sets all δ_r to ϵ_1. To study the performance drop due to the use of surrogate models, we also compared with an attacker which has white-box access to the victim model. These two methods will be referred to as `poison-uniform` and `poison-wb`, respectively.

Results Figure 3 shows the average online return during the offline training (left) and online fine-tuning (right), both at $\epsilon = 0.1$. All poisoned victims perform similarly to the clean trained agent during the offline phase, fulfilling the stealthiness objective. However, our O2OP drove down the online return from 0.65 to 0.3, which is only slightly higher than that of the white-box poisoner (0.25). In contrast, the online return of the uniform baseline stayed above 0.45. We also aggregated the average returns over all offline or online steps by taking their mean. This is provided in the legend.

We further reduced our budget to $\epsilon_1 = 0.02$ in Figure 4. Here, the stealthiness remains satisfied offline. During online fine-tuning, the uniform poisoned victim agent has a minimum average return above 0.5, while our O2OP drives it below 0.35, which is almost the same as the white-box attacker. This confirms the effectiveness of our O2OP and its transferability across different model initialization.

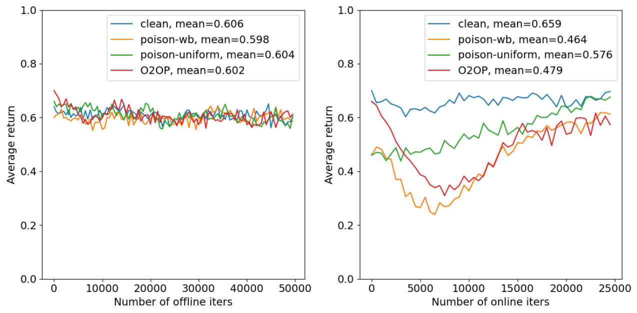


Figure 3: O2O return in offline phase (left) and online phase (right) for Frozen Lake with $\epsilon_1 = 0.1$

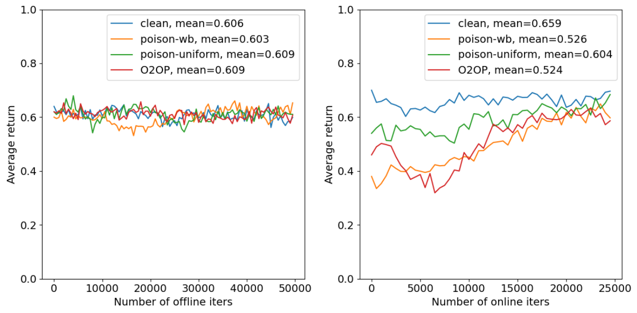


Figure 4: O2O return in offline phase (left) and online phase (right) for Frozen Lake with $\epsilon_1 = 0.02$

5.2 CONTINUOUS ENVIRONMENTS

We next move on to illustrate the attack effectiveness in a *continuous* space, using the environments of Hopper, Half Cheetah, and Walker2d. The difficulty level is medium. The continuous CQL agents were trained for 600 epochs offline, with 500 gradient steps per epoch. The online continuous **SAC** agents were trained for 100 epochs. We reduced the poison ratio to 2% for more realistic attacks.

Hopper As the Hopper environment has rewards ranging in $(0, 6)$, we increased our poison's ℓ_1 norm budget to $\epsilon_1 = 4$. To improve stealthiness, we enforced constraint $\|\delta_r\|_\infty \leq \epsilon_\infty = 5$. Accordingly, the same modification was made on `poison-uniform`. Despite the slightly high values of ϵ_1 and ϵ_∞, we only poison 2% of the transitions, which is consistent with poisoning or backdoor attacks in supervised learning.

Figure 5 shows that, analogously to Frozen Lake, all the three poisoners perform similarly to the clean unpoisoned case in terms of the offline performance, which again confirms the stealthiness of O2OP. During online fine-tuning, however, O2OP achieves a performance drop from 3000 to 2600 (when online iteration is around 46000), while the white-box version can further slash it to 2000. In contrast, `poison-uniform` can hardly degrade the online return, if at all. This shows that O2OP remains effective in this

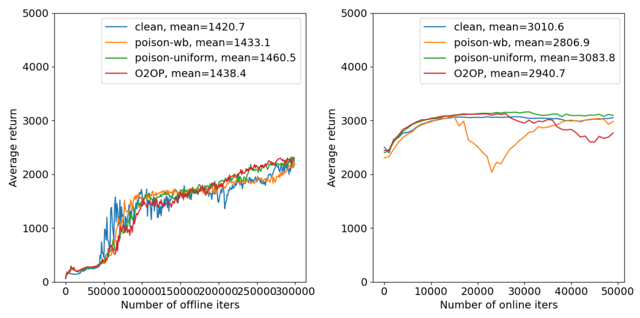


Figure 5: O2O return in offline phase (left) and online phase (right) for Hopper with 2% poison

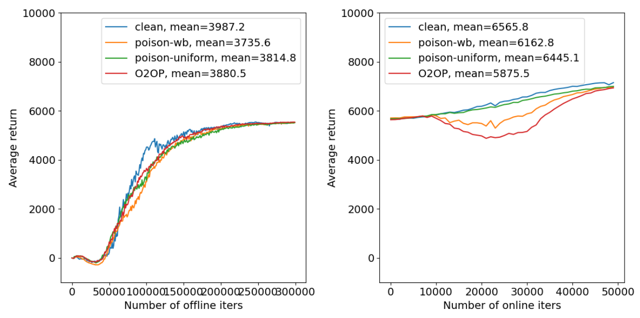


Figure 6: O2O return in offline phase (left) and online phase (right) for Half Cheetah with 2% poison

continuous space with a small poison ratio.

Half Cheetah The reward in Half Cheetah lies between -3 and 9, with the mean around 5. We again only poisoned 2% transitions, and set $\epsilon_1 = 4$ and $\epsilon_\infty = 5$. Similarly to Hopper, Figure 6 shows our O2OP effectively created a return drop during the online fine-tuning, while `poison-uniform` is again nearly harmless to the victim at the same ratio and budget. The offline stealthiness is evidenced once more as the four methods achieve similar offline returns.

Walker2d This environment has similar reward range as Half Cheetah, and we thus used identical settings to it. As shown in Figure 7, the poisoned offline return remains comparable to the clean offline return, i.e., stealthy. Although the online return seems less stable than in the previous experiments, O2OP managed to curtail the return from 3800 to 2500, while the `clean` and `poison-uniform` baselines produce returns fluctuating between 3200 and 4000.

5.3 IMPACT OF ℓ_p NORM BUDGET

We also tested with different budget of ϵ_1 and ϵ_∞ on Hopper. As Figure 8 shows, different budgets do not affect the offline return too much. On the other hand, the amount of online performance drop does vary significantly with the budgets.

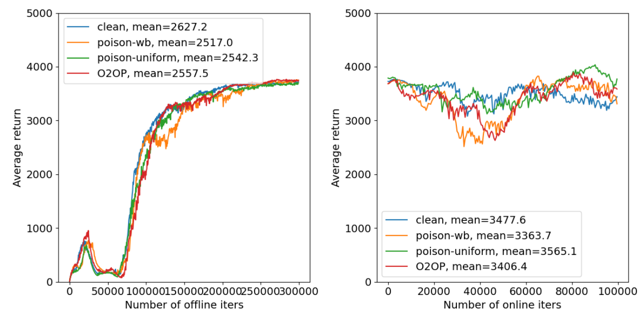


Figure 7: O2O return in offline phase (left) and online phase (right) for Walker2d with 2% poison

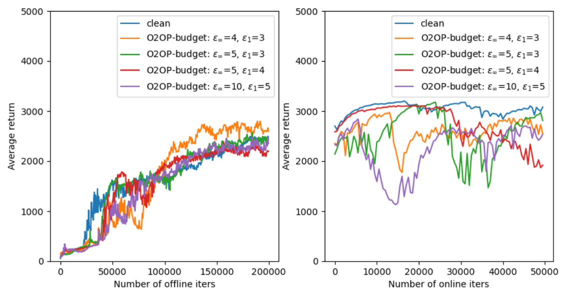


Figure 8: O2O return in offline phase (left) and online phase (right) on Hopper with varying ϵ_1 and ϵ_∞ budgets

In general, a larger poison budget leads to a greater drop.

6 CONCLUSION AND FUTURE WORK

We proposed a novel poisoning attack that reveals the vulnerability of the offline-to-online reinforcement learning. While keeping the offline performance almost intact, the online fine-tuning suffers a significant performance drop before it asymptotically recovers. Our approach takes advantage of the distribution shift phenomenon by promoting the Q-value for out of distribution actions. In future work, we will extend our attack to other O2O RL algorithms, and will explore possible defense strategies towards a more robust O2O training pipeline.

REFERENCES

Bengio, Y. Gradient-based optimization of hyperparameters. *Neural Comput.*, 12(8):1889–1900, 2000. doi: 10.1162/089976600300015187. URL `https://doi.org/10.1162/089976600300015187`.

Biggio, B., Nelson, B., and Laskov, P. Poisoning attacks against support vector machines. In *International Conference on Machine Learning (ICML)*, 2012.

Chen, D. and Wen, Y. Dcac: Reducing unnecessary conservatism in offline-to-online reinforcement learning. In *Proceedings of the Fifth International Conference on Distributed Artificial Intelligence*, pp. 1–12, 2023.

Chen, X., Liu, C., Li, B., Lu, K., and Song, D. Targeted backdoor attacks on deep learning systems using data poisoning. *ArXiv*, abs/1712.05526, 2017.

Cherepanova, V., Goldblum, M., Foley, H., Duan, S., Dickerson, J. P., Taylor, G., and Goldstein, T. Lowkey: Leveraging adversarial attacks to protect social media users from facial recognition. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=hJmtwocEqzc.

Farahmand, A.-m., Szepesvári, C., and Munos, R. Error propagation for approximate policy and value iteration. In *Advances in Neural Information Processing Systems*, 2010.

Fu, J., Kumar, A., Soh, M., and Levine, S. Diagnosing bottlenecks in deep q-learning algorithms. In *International Conference on Machine Learning (ICML)*, pp. 2021–2030, 2019.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4RL: datasets for deep data-driven reinforcement learning. *CoRR*, abs/2004.07219, 2020. URL https://arxiv.org/abs/2004.07219.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning (ICML)*, 2019.

Geiping, J., Fowl, L. H., Huang, W. R., Czaja, W., Taylor, G., Moeller, M., and Goldstein, T. Witches' brew: Industrial scale data poisoning via gradient matching. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=01olnfLIbD.

Goldblum, M., Tsipras, D., Xie, C., Chen, X., Schwarzschild, A., Song, D., Madry, A., Li, B., and Goldstein, T. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(2):1563–1580, 2023. doi: 10.1109/TPAMI.2022.3162397. URL https://doi.org/10.1109/TPAMI.2022.3162397.

Gong, C., Yang, Z., Bai, Y., He, J., Shi, J., Sinha, A., Xu, B., Hou, X., Fan, G., and Lo, D. Mind your data! hiding backdoors in offline reinforcement learning datasets. *CoRR*, abs/2210.04688, 2022. doi: 10.48550/ARXIV.2210.04688. URL https://doi.org/10.48550/arXiv.2210.04688.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6572.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Huang, S. H., Papernot, N., Goodfellow, I. J., Duan, Y., and Abbeel, P. Adversarial attacks on neural network policies. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=ryvlRyBKl.

Kang, S., Shi, Z., and Zhang, X. Poisoning generative replay in continual learning to promote forgetting. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 15769–15785. PMLR, 2023. URL https://proceedings.mlr.press/v202/kang23c.html.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Kumar, A., Gupta, A., and Levine, S. Discor: Corrective feedback in reinforcement learning via distribution correction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020a.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020b. URL https://proceedings.neurips.cc/paper/2020/hash/0d2b2061826a5df3221116a5085a6052-Abstract.html.

Laroche, R., Trichelair, P., and des Combes, R. T. Safe policy improvement with baseline bootstrapping. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3652–3661. PMLR, 2019. URL `http://proceedings.mlr.press/v97/laroche19a.html`.

Lee, S., Seo, Y., Lee, K., Abbeel, P., and Shin, J. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pp. 1702–1712. PMLR, 2022.

Lei, K., He, Z., Lu, C., Hu, K., Gao, Y., and Xu, H. Uni-o4: Unifying online and offline deep reinforcement learning with multi-step on-policy optimization. *arXiv preprint arXiv:2311.03351*, 2023.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020. URL `https://arxiv.org/abs/2005.01643`.

Lorraine, J., Vicol, P., and Duvenaud, D. Optimizing millions of hyperparameters by implicit differentiation. In Chiappa, S. and Calandra, R. (eds.), *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pp. 1540–1552. PMLR, 2020. URL `http://proceedings.mlr.press/v108/lorraine20a.html`.

Ma, Y., Zhang, X., Sun, W., and Zhu, J. Policy poisoning in batch reinforcement learning and control. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 14543–14553, 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/315f006f691ef2e689125614ea22cc61-Abstract.html`.

Maclaurin, D., Duvenaud, D., and Adams, R. P. Gradient-based hyperparameter optimization through reversible learning. In Bach, F. R. and Blei, D. M. (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 2113–2122. JMLR.org, 2015. URL `http://proceedings.mlr.press/v37/maclaurin15.html`.

Mark, M. S., Sharma, A., Tajwar, F., Rafailov, R., Levine, S., and Finn, C. Offline retraining for online rl: Decoupled policy learning to mitigate exploration bias. *arXiv preprint arXiv:2310.08558*, 2023.

Munos, R. Error bounds for approximate value iteration. In *National Conference of Artificial Intelligence (AAAI)*, 2005.

Nair, A., Gupta, A., Dalal, M., and Levine, S. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Nakamoto, M., Zhai, Y., Singh, A., Mark, M. S., Ma, Y., Finn, C., Kumar, A., and Levine, S. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *arXiv preprint arXiv:2303.05479*, 2023.

Rakhsha, A., Zhang, X., Zhu, X., and Singla, A. Reward poisoning in reinforcement learning: Attacks against unknown learners in unknown environments. *CoRR*, abs/2102.08492, 2021. URL `https://arxiv.org/abs/2102.08492`.

Shaban, A., Cheng, C., Hatch, N., and Boots, B. Truncated back-propagation for bilevel optimization. In Chaudhuri, K. and Sugiyama, M. (eds.), *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pp. 1723–1732. PMLR, 2019. URL `http://proceedings.mlr.press/v89/shaban19a.html`.

Shafahi, A., Huang, W., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Souri, H., Fowl, L., Chellappa, R., Goldblum, M., and Goldstein, T. Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL `http://papers.nips.cc/paper_files/paper/2022/hash/79eec295a3cd5785e18c61383e7c996b-Abstract-Confe html`.

Sun, J., Zhang, T., Xie, X., Ma, L., Zheng, Y., Chen, K., and Liu, Y. Stealthy and efficient adversarial attacks against deep reinforcement learning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial*

Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, pp. 5883–5891. AAAI Press, 2020. doi: 10.1609/AAAI.V34I04.6047. URL `https://doi.org/10.1609/aaai.v34i04.6047`.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Wagenmaker, A. and Pacchiano, A. Leveraging offline data in online reinforcement learning. In *International Conference on Machine Learning*, pp. 35300–35338. PMLR, 2023.

Wang, S., Yang, Q., Gao, J., Lin, M. G., Chen, H., Wu, L., Jia, N., Song, S., and Huang, G. Train once, get a family: State-adaptive balances for offline-to-online reinforcement learning. *arXiv preprint arXiv:2310.17966*, 2023.

Wu, Y., McMahan, J., Zhu, X., and Xie, Q. Reward poisoning attacks on offline multi-agent reinforcement learning. In Williams, B., Chen, Y., and Neville, J. (eds.), *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pp. 10426–10434. AAAI Press, 2023. doi: 10.1609/AAAI.V37I9.26240. URL `https://doi.org/10.1609/aaai.v37i9.26240`.

Xu, H., Wang, R., Raizman, L., and Rabinovich, Z. Transferable environment poisoning: Training-time attack on reinforcement learning. In Dignum, F., Lomuscio, A., Endriss, U., and Nowé, A. (eds.), *AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021*, pp. 1398–1406. ACM, 2021. doi: 10.5555/3463952.3464113. URL `https://www.ifaamas.org/Proceedings/aamas2021/pdfs/p1398.pdf`.

Yu, Z. and Zhang, X. Actor-critic alignment for offline-to-online reinforcement learning. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 40452–40474. PMLR, 2023. URL `https://proceedings.mlr.press/v202/yu23k.html`.

Zhang, H., Xu, W., and Yu, H. Policy expansion for bridging offline-to-online reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2022.

Zhang, X., Ma, Y., Singla, A., and Zhu, X. Adaptive reward-poisoning attacks against reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11225–11234. PMLR, 2020. URL `http://proceedings.mlr.press/v119/zhang20u.html`.