ORL-AUDITOR: Dataset Auditing in Offline Deep Reinforcement Learning

Linkang Du[†*], Min Chen[‡*], Mingyang Sun[†], Shouling Ji[†], Peng Cheng[†], Jiming Chen[†], Zhikun Zhang[‡†§¶]
[†] Zhejiang University, Hangzhou 310017, China
Email: linkangd@gmail.com, mingyangsun@zju.edu.cn, sji@zju.edu.cn, saodiseng@gmail.com, cjm@zju.edu.cn
[‡]CISPA Helmholtz Center for Information Security, Saarbrücken 66123, Germany
Email: min.chen@cispa.de
[§]Stanford University, Stanford, California 94305, USA
Email: zhikun@stanford.edu

***Abstract*—Data is a critical asset in AI, as high-quality datasets can significantly improve the performance of machine learning models. In safety-critical domains such as autonomous vehicles, offline deep reinforcement learning (offline DRL) is frequently used to train models on pre-collected datasets, as opposed to training these models by interacting with the real-world environment as the online DRL. To support the development of these models, many institutions make datasets publicly available with open-source licenses, but these datasets are at risk of potential misuse or infringement. Injecting watermarks to the dataset may protect the intellectual property of the data, but it cannot handle datasets that have already been published and is infeasible to be altered afterward. Other existing solutions, such as dataset inference and membership inference, do not work well in the offline DRL scenario due to the diverse model behavior characteristics and offline setting constraints.**

In this paper, we advocate a new paradigm by leveraging the fact that cumulative rewards can act as a unique identifier that distinguishes DRL models trained on a specific dataset. To this end, we propose ORL-AUDITOR, which is the first trajectory-level dataset auditing mechanism for offline RL scenarios. Our experiments on multiple offline DRL models and tasks reveal the efficacy of ORL-AUDITOR, with auditing accuracy over 95% and false positive rates less than 2.88%. We also provide valuable insights into the practical implementation of ORL-AUDITOR by studying various parameter settings. Furthermore, we demonstrate the auditing capability of ORL-AUDITOR on open-source datasets from Google and DeepMind, highlighting its effectiveness in auditing published datasets. ORL-AUDITOR is open-sourced at https://github.com/link-zju/ORL-Auditor.

I. INTRODUCTION

Deep reinforcement learning (DRL) has been successfully applied to many complex decision-making tasks, such as autopilot [16], robot control [3], [50], power systems [69], intrusions detection [41], [66].

[*]The first two authors made equal contribution.
[¶]Zhikun Zhang is the corresponding author.

Network and Distributed System Security (NDSS) Symposium 2024
26 February - 1 March 2024, San Diego, CA, USA
ISBN 1-891562-93-2
https://dx.doi.org/10.14722/ndss.2024.23184
www.ndss-symposium.org

However, for safety-critical domains, such as robot control, directly interacting with the environment is unsafe since the partially trained policy may risk damage to robot hardware or surrounding objects [54]. To address this issue, researchers propose the *offline deep reinforcement learning* (Offline DRL) [36] paradigm, also known as full batch DRL [35]. The general idea is learning from pre-collected data generated by the expert, handcrafted controller, or even random strategy respecting the system's constraints.

To facilitate the research of offline DRL, several high-quality datasets are published by third parties such as DeepMind [25], [4], Berkeley Artificial Intelligence Research (BAIR) [17], Polixir Technologies [51], TensorFlow [1], and Max Planck Institute [26]. These datasets are published with strict open-source licenses, such as GNU General Public License [4], Apache License [25], [17], [1], [51], and BSD 3-Clause License [26], to protect the intellectual property (IP) of the data owner. The licenses typically encompass two essential terms. 1) Attribution requires you (the users) to appropriately acknowledge the source, provide a link to the license, and indicate any modifications made. 2) ShareAlike stipulates that if you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. Furthermore, some datasets are accompanied by additional patent grants aimed at safeguarding the rights of data publishers, *e.g.* StarData [39]. Additionally, closed-form datasets have the potential to face misuse from insider attacks or intellectual property infringement (*e.g.*, ex-employees stealing data). Biscom's 2021 survey finds that 25% of respondents admitted to taking the valuable data when leaving their job, with 95% citing a lack of policies or technologies to prevent data theft [5]. Tessian reports that 40% of US employees take their generated data or trained models when leaving their job [60]. The defense against the above threats comes to the question of *how a data owner can prove that a suspect model was derived from its dataset.*

Existing Solutions. Recent mainstream solutions for dataset copyright protection can be classified into three categories: Watermarking, dataset inference, and membership inference. The *watermarking* approach aims to inject samples from a specific distribution prior to publishing the dataset [38], [37]. However, the auditor needs a post-event mechanism for open-source data since they are already published in the real world. In contrast to watermarking techniques, *dataset*

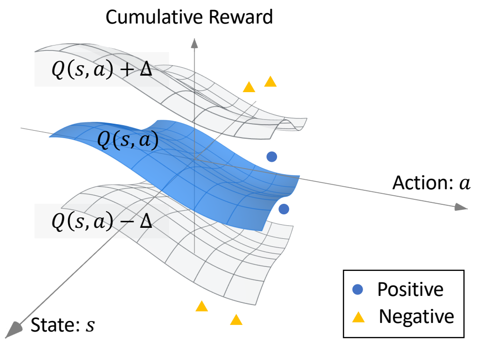


Fig. 1: Intuitive explanation of ORL-AUDITOR. The middle surface is the cumulative rewards of the state-action pairs from a dataset. The auditor outputs a positive result if the cumulative rewards of a suspect model's state-action pairs are between the two outer surfaces.

inference strategies [42], [15] do not require the injection of explicit watermarks [6] into the datasets or trained models. Maini *et al.* [42] and Dziedzic *et al.* [15] have separately proposed dataset inference methods for supervised learning and self-supervised learning models, enabling the model owner to provide a convincing statistical argument that a particular model is trained on their private data. However, the dataset inference with labels [42] needs distances between data and decision boundaries, which is not possible to obtain in RL with continuous outputs. The dataset inference without labels [15] uses the similarity of model behaviors to detect unauthorized dataset usage. It requires a public dataset to generate some surrogate models, and forms the auditing basis by comparing the behavioral difference between the surrogate models and the models trained on their private data. In offline RL scenes, since the distributions of the collected datasets depend on both environment and operator [17], it is difficult to determine suitable public dataset to train the surrogate model, making the audit basis hard to establish. The third category adopts the notion of *membership inference* [46], [23], [22]. By collecting the RL models' behaviors on the trained examples (members) and the untrained examples (non-members), a classifier is constructed to determine whether a data sample is used in the model's learning process. However, unlike online scenarios in [46], [23], [22], the auditor cannot collect additional data from the environment as the non-member examples in offline cases, where the auditor does not have access to the environment.

Our Proposal. In this paper, we propose the first practical dataset auditing paradigm for the offline RL model (ORL-AUDITOR). Concretely, we are inspired by the fact that the *cumulative reward*, *i.e.*, the sum of all rewards received over a period of time starting from a given state-action pair, guiding the RL model to learn the behavior policy. Thus, the cumulative reward is an intrinsic feature of the datasets, making it suitable as an audit basis. Figure 1 provides a schematic diagram of ORL-AUDITOR, where the state, the action, and the cumulative reward compose a three-dimensional space. The middle surface illustrates the exact cumulative reward of the dataset, and the other two surfaces show possible offsets of the exact cumulative reward learned by the offline DRL models due to the randomness in the initialization and the learning processes. For a *suspect model*, the auditor outputs a positive result, *i.e.*, the data is used to train this model, if the cumulative reward from its state-action pair falls between the two surfaces; otherwise, a negative outcome.

To implement the auditing, we first train a critic model to predict the cumulative rewards of the state-action pairs in the dataset to be audited, *i.e.*, the target dataset. A straightforward strategy to derive the auditing result is to compare the cumulative reward of the state-action pairs from the suspect model to that of the target dataset through a preset judgment threshold of the similarity. However, designing the threshold value is challenging, as it depends on the distributions of pre-collected datasets, which can vary due to different task settings, collection procedures, and data post-processing methods. To address this issue, we recognize that the cumulative rewards embedded in the state-action pairs of the models are the estimated cumulative rewards of the target dataset, as the offline DRL models fit the cumulative reward of the dataset during training. Thus, we train multiple models on the target dataset with varying initializations and optimization, *i.e.*, the shadow models, and collect the cumulative rewards of their state-action pairs. Finally, by comparing the cumulative rewards from the suspect model and the shadow models, we make the audit decision through hypothesis testing.

Evaluation. The experimental results show that the auditing accuracy of ORL-AUDITOR exceeds 95% with false positive rates less than 2.88% across multiple DRL models and tasks. By visualizing the cumulative rewards from the shadow models trained on different datasets, we demonstrate that the cumulative reward is a distinguishable feature for the dataset audit. We further evaluate three influential factors for the practical adoption of ORL-AUDITOR, *i.e.*, the number of shadow models, the significance level in hypothesis testing, and the trajectory size. First, more shadow models improve the audit accuracy, and ORL-AUDITOR demonstrates exceptional performance with an audit accuracy exceeding 90% utilizing a mere 9 shadow models. Second, the minimum significance level α of ORL-AUDITOR is about 0.001, meaning that the auditor outputs a single result with 99.9% confidence. Third, ORL-AUDITOR tends to obtain higher accuracy with a larger trajectory size, yet we also notice that a small trajectory size achieves better results under some tasks [45]. We further implement ORL-AUDITOR to audit the open-source datasets from Google [17] and DeepMind [25], and the experimental results again demonstrate the effectiveness of ORL-AUDITOR in real-world application.

Robustness. To evaluate the robustness of ORL-AUDITOR, we have implemented two defense strategies to prevent the auditing. The first strategy involves using state-of-the-art membership inference defense techniques, such as the ensemble architecture proposed by Tang *et al.* [59] and Jarin *et al.* [30]. Despite these defense mechanisms, the audit accuracy of ORL-AUDITOR is still over 85%. In addition to the ensemble architecture, the suspect models may distort actions to hide their training dataset. The offline DRL models for real-world decision-making tasks (*i.e.*, self-driving cars) often use Gaussian noise to model natural distortions [2]. Thus, adding Gaussian noise to the actions is stealthy to avoid the auditor's detection, and Gaussian noise is convenient for mathematical manipulation. To simulate strong and weak action distortion, we normalize all dimensions of the action space to $[-1, 1]$

and use Gaussian noise with $(\mu = 0, \sigma = 0.1)$ and $(\mu = 0, \sigma = 0.01)$, respectively. Our experiments show that ORL-AUDITOR is only slightly affected by Gaussian noise with $(\mu = 0, \sigma = 0.01)$. For $\sigma = 0.1$, the TPR values of ORL-AUDITOR decline, yet the strong distortion also impacts the performance of the suspect model, especially in complex tasks.

Contributions. Our contributions are three-fold:

- To our knowledge, ORL-AUDITOR is the first dataset auditing method for the offline DRL models, using the cumulative reward as an intrinsic and stable fingerprint of the dataset.
- We demonstrate the effectiveness of ORL-AUDITOR on four offline DRL models and three tasks. We also systematically analyze various experimental factors, *i.e.*, the hyperparameter settings and the robustness of ORL-AUDITOR, and summarize some important guidelines for adopting ORL-AUDITOR in practice.
- By implementing ORL-AUDITOR on the open-source datasets from DeepMind [25] and Google [17], we show that ORL-AUDITOR can serve as a potent audit solution in real-world offline DRL scenarios.

II. BACKGROUND

A. *Offline RL Problem*

The offline reinforcement learning (offline RL) model aims to learn an optimal (or nearly optimal) policy from a pre-collected dataset D *without* an interactive environment. We use $\mathbb{S}$ and $\mathbb{A}$ to represent the RL models' input and output space, formally called *state* and *action* in RL scenes. $r_t \in \mathbb{R}$ is the temporal reward for each time step, where $\mathbb{R}$ is the real number set. A unit in a pre-collected dataset called *transition* is a four-element set: $\{s_t, a_t, r_t, s_{t+1}\}$, where $s_t \in \mathbb{S}$, $a_t \in \mathbb{A}$, and $s_{t+1} \in \mathbb{S}$ is the successive state of s_t. And a set of transitions in chronological order forms a *trajectory* in dataset D. Based on the transitions, the offline RL model learns the Markov Decision Process underneath the datasets and forms a policy $\pi_\theta(a \mid s)$ to maximize $J(\pi)$.

$$J(\pi) = \mathbb{E}_{s_t \sim d_\beta(s,\ a),\ a_t \sim \pi_\theta(a|s)} \left[\sum_{t=0}^{H} \gamma^t r_t \right],$$

where we use d_β to denote the distribution over states and actions in dataset D, and the actions are sampled according to the behavior policy $a_t \sim \pi_\theta(a \mid s)$. The discount factor γ is applied to discount future rewards in the accumulated reward. H is the terminal time step of one trajectory.

Example. Figure 2 shows an example based on the "CartPole" task.[1] In the data collection process, the dataset is generated from the operation logs between the operator and the environment, which contains the position and velocity of the cart and the pole (*i.e.*, state), the operator's force direction (*i.e.*, action), and the corresponding rewards. Then, in the training and evaluation process, the offline RL model learns how to play the "Cartpole" task from only the pre-collected dataset generated through the data collection process. Finally, we deploy the well-trained offline RL model in the environment to perform the task.

[1] https://www.gymlibrary.dev/environments/classic_control/cart_pole/

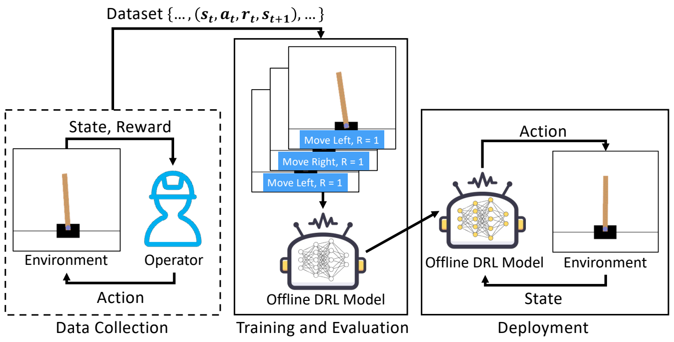


Fig. 2: A running example of the offline DRL models.

B. *Offline RL Models*

In this section, we first introduce two offline RL algorithms [20], [18], [34] separately representing two basic ideas of the offline RL models, *i.e.*, the policy constraints strategy and the value function regularization strategy [49]. Many state-of-the-art model-free offline RL methods [67], [31], [19], [34] have been modified from these two approaches. We further present a state-of-the-art algorithm [19] which is minimalistic with light computation and hyperparameter setting overhead. In addition, we briefly describe the behavior clone method (BC) [48], which learns the state-action distribution over the dataset via a supervised learning approach. Though BC is not a typical reinforcement learning method, it can solve the offline RL problem and usually serves as the baseline method in the offline RL evaluation.

Behavior Clone (BC) [48]. BC separately takes the pairwise state s and action a in the datasets as input and label, then it optimizes the policy through the following function.

$$\theta^* = \arg\min_\theta \mathbb{E}_{(s,a)\sim D} \left[\mathcal{L}\left(\pi_\theta(s), a\right) \right],$$

where D is the pre-collected dataset and $\mathcal{L}$ is the loss function. Since BC only imitates action distributions, the performance is close to the mean of the dataset, even though BC works better than online RL algorithms in most cases.

Batch-Constrained Q-learning (BCQ) [20], [18]. BCQ is the first practical data-driven offline RL algorithm. The key idea of BCQ is to integrate a generative model to achieve the notion of batch-constrained, *i.e.*, minimizing the deviation between the candidate actions with the action records of the dataset. To maintain the diversity of action, BCQ builds a perturbation model to perturb each selected action. Then it chooses the highest-valued action through a Q-network, that learns to estimate the expected cumulative reward of a given state and action pair. Thus, the objective function of BCQ can be defined as the following.

$$\pi(s) = \underset{a_i + \xi_\phi(s, a_i, \Phi)}{\operatorname{argmax}} \; Q_\theta\left(s, a_i + \xi_\phi\left(s, a_i, \Phi\right)\right)$$
$$\{a_i \sim G_\omega(s)\}_{i=1}^{n},$$

where $G_\omega(s)$ is a conditional variational auto-encoder (VAE)-based [32] generative model that can be used to generate candidate actions. The value function Q_θ is used to score the n candidate actions and finds the action with the highest value. $\xi_\phi\left(s, a_i, \Phi\right)$ is the perturbation model, which outputs an adjustment to an action a in the range $[-\Phi, \Phi]$. Then, the perturbation model can be optimized by the deterministic policy gradient algorithm [57] as follows.

$$\phi \leftarrow \underset{\phi}{\operatorname{argmax}} \sum_{(s,a)\in\mathcal{B}} Q_\theta\left(s, a+\xi_\phi(s,a,\Phi)\right),$$

where $\mathcal{B}$ represents a mini-batch state-action pair in the dataset. To penalize rare states, BCQ takes a convex combination of the values from two Q-networks and sets a new target value y to update both Q-networks.

$$y = r + \gamma \max_{a_i}\left[\lambda \min_{j=1,2} Q_{\theta'_j}\left(s', a_i\right) + (1-\lambda) \max_{j=1,2} Q_{\theta'_j}\left(s', a_i\right)\right],$$

where a_i corresponds to the perturbed actions, sampled from the generative model $G_\omega(s)$.

Implicit Q-Learning (IQL) [34]. Compared to the batch-constrained idea of BCQ [20], [18], IQL strictly avoids querying values of the actions, which are not in the pre-collected dataset. IQL first constructs a model to evaluate the expected returns of state-action pairs. The objective function is defined as shown in Equation 1.

$$L(\theta) = \mathbb{E}_D\left[L_2^\tau\left(r(s,a) + \gamma Q_{\hat{\theta}}\left(s', a'\right) - Q_\theta(s,a)\right)\right], \quad (1)$$

where $L_2^\tau(u) = |\tau - \mathbb{1}(u<0)|u^2$, and s' and a' represent the successor state and action of s and a. Both $Q_\theta(s,a)$ and $Q_{\hat{\theta}}$ are used to assess the expected returns of state-action pairs. The parameters of $Q_\theta(s,a)$ are adjusted in each optimization round, while the parameters of $Q_{\hat{\theta}}$ are updated periodically based on $Q_\theta(s,a)$ to reduce parameter fluctuations during model updates. Equation 1 involves the dynamics of the environment, where the environment state s transitions to the next environment state s', potentially introducing interference in the evaluation of expected returns for state-action pairs. IQL addresses this issue by introducing a new state value model, splitting Equation 1 into two objective functions. Equation 2 shows the objective function of the state value model V_ψ.

$$L_V(\psi) = \mathbb{E}_D\left[L_2^\tau\left(Q_{\hat{\theta}}(s,a) - V_\psi(s)\right)\right]. \quad (2)$$

Then, IQL utilizes $V_\psi(s)$ to construct Equation 3 for updating the parameters of the state-action value model Q_θ.

$$L_Q(\theta) = \mathbb{E}_D\left[\left(r(s,a) + \gamma V_\psi\left(s'\right) - Q_\theta(s,a)\right)^2\right]. \quad (3)$$

Finally, IQL considers using the state-action value model to construct a behavior policy for deployment. This behavior policy also needs to avoid actions that are outside the dataset distribution. Thus, IQL employs advantage-weighted regression to update the policy model.

$$L_\pi(\phi) = \mathbb{E}_D\left[\exp\left(\beta\left(Q_\theta(s,a) - V_\psi(s)\right)\right)\log \pi_\phi(a \mid s)\right], \quad (4)$$

where $\beta \in [0, \infty)$ represents the inverse temperature. For smaller values of β, IQL is similar to behavior clone, tending to mimic the data collection policy. For larger values of β, IQL is more inclined to select actions corresponding to the highest expected returns according to the state-action value model. Throughout the entire training process, IQL alternates between optimizing the parameters θ and ψ, and then updates ϕ while keeping θ and ψ fixed.

TD3PlusBC [19]. The former methods [20], [18], [34] limit or regularize action selection such that the learned policy is easier to evaluate with the given dataset. However, they introduce new hyperparameters and often leverage secondary components, such as generative models, while adjusting the underlying RL algorithm. TD3PlusBC is a minimalist and highly effective offline RL algorithm based Twin Delayed Deep Deterministic Policy Gradient (TD3) [21] with BC regularization term, which pushes the policy towards favoring actions contained in the dataset D:

$$\pi = \underset{\pi}{\operatorname{argmax}} \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\lambda Q(s,\pi(s)) - (\pi(s) - a)^2\right],$$

where $\lambda = \frac{\alpha}{\frac{1}{N}\sum_{(s,a)}|Q(s,a)|}$ for the dataset of N transitions (s,a). To facilitate the policy training, TD3PlusBC normalizes each state in the given dataset by $s_i = \frac{s_i - \mu}{\sigma + \epsilon}$, where μ and σ are the mean and standard deviation respectively.

The model architectures vary significantly regarding objective function and basic model structure. 1) Objective Function: BCQ [20], [18] and TD3PlusBC [19] use a policy constraints strategy to maintain the learned policy similar to the one used for collecting the dataset. In contrast, IQL [34] adopts a regularization strategy to improve the stochasticity of the learned policy or obtain more accurate Q-value estimations. 2) Basic Model Structures: BCQ [20], [18] and IQL [34] are based on the Q-learning model, while TD3PlusBC [19] builds upon TD3 [21]. In Section V, our experiments are mainly conducted on the above four algorithms. However, ORL-AUDITOR can also be applied to any type of offline DRL model as long as the auditor has black-box access to the suspect model.

III. PROBLEM STATEMENT AND EXISTING SOLUTIONS

A. System and Threat Model

Application Scenarios. Figure 3 illustrates a typical application scenario where the data providers collect and then publish or sell the dataset to the customers. A malicious customer (adversary) with access to the datasets makes a piracy distribution or illegally builds a Model-as-a-Service (MaaS) platform. Institution 1 suspects the models are generated by its dataset, and thus hires an auditor to determine whether the model trainers pirate the trajectories of the dataset D_1.

Auditor's Background Knowledge and Capability. The auditor has full knowledge of the target dataset, such as the number of trajectories and the spaces of state and action. In offline RL settings, the auditor is prohibited from interacting with the online environment to collect more data, meaning the entire auditing only depends on the target dataset. We consider the auditor has black-box access to the suspect RL model. Note that this is the most general and challenging scenario for the auditor. A typical application scenario is that an adversary receives the model settings from customers, such as the selected offline RL framework, the model's hyperparameter, and the desired training episodes. Then, the adversary trains an offline RL model and provides a service interface to the customers. The auditor utilizes the states of the dataset (inputs) to query the suspect model and obtain the corresponding actions (outputs).

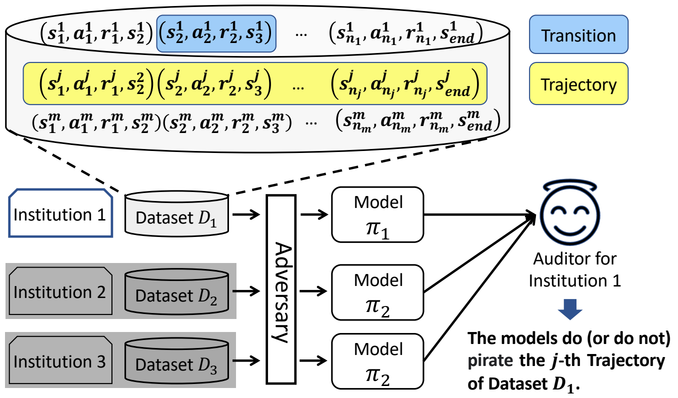

$(s_1^1, a_1^1, r_1^1, s_2^1)(s_2^1, a_2^1, r_2^1, s_3^1)$ … $(s_{n_1}^1, a_{n_1}^1, r_{n_1}^1, s_{end}^1)$
$(s_1^j, a_1^j, r_1^j, s_2^j)(s_2^j, a_2^j, r_2^j, s_3^j)$ … $(s_{n_j}^j, a_{n_j}^j, r_{n_j}^j, s_{end}^j)$
$(s_1^m, a_1^m, r_1^m, s_2^m)(s_2^m, a_2^m, r_2^m, s_3^m)$ … $(s_{n_m}^m, a_{n_m}^m, r_{n_m}^m, s_{end}^m)$
Transition
Trajectory
Institution 1
Dataset D_1
Institution 2
Dataset D_2
Institution 3
Dataset D_3
Adversary
Model π_1
Model π_2
Model π_2
Auditor for Institution 1
The models do (or do not) pirate the j-th Trajectory of Dataset D_1.

Fig. 3: An example of the application scenario. The auditor can obtain all information about dataset D_1 but has no knowledge about the datasets from other institutions.

Discussion. Compared to the sample-level and dataset-level data in DNN scenes, RL has trajectory-level data, which is the minimum record unit of sequential interactions between the operator and environment. Since a single trajectory can guide the model from the initial state to the terminal, the trajectory-level data is regarded as the value unit of the dataset. Thus, ORL-AUDITOR is designed to audit the dataset from the trajectory level, where the auditor tries to decide whether the suspect model uses a specific trajectory in the dataset. In addition, the auditor can easily extend ORL-AUDITOR to the dataset-level data by setting a piracy alarm threshold. If the ratio of misappropriation using trajectories exceeds the preset threshold, the auditor can claim the dataset-level pirate.

B. Existing Solutions

Watermarking [38], [37]. Watermarking-based dataset copyright protection methods inject samples of a specific distribution before publishing the target dataset. One of its kind is implemented with backdoor attacks against the ML model. Li *et al.* [38] proposed to modify a dataset by adding a trigger, such as a local patch, to innocent samples in order to make them appear as a pre-defined target class. To verify the integrity of the dataset after the attack, they use a hypothesis test approach based on posterior probabilities generated by a third-party model. Inspired by this idea, the auditor can employ the backdoor attack against the DRL model [33], [63], [65] to generate a watermark for the offline RL dataset.

However, since the open-source datasets are already published, the auditor needs a post-event mechanism that does not require injecting manipulated samples before publishing the dataset. Watermarking, on the other hand, is a pre-event mechanism that involves injecting manipulated samples into the dataset before publishing. Additionally, it is difficult for the auditor to guarantee that one effective watermarking has a consistent distribution with the original dataset, which inevitably disturbs the model's normal behavior.

Dataset Inferences [42], [15]. The core idea of dataset inference is empowering the model owner to make a compelling statistical argument that a particular model is a copied version of their own model by demonstrating that it is based on their private training data. It does not require injecting explicit watermarks [6] to the datasets or the trained models. Existing methods [42], [15] can be divided into two categories according to whether they have explicit classification labels. With the explicit classification labels, [42] rely on computing the distances between data points and decision boundaries. Without the explicit classification labels, [15] utilizes the similarity of the models' behaviors to detect the unauthorized usage of the dataset, which requires the assumption of an additional public dataset with a similar distribution to form the auditing basis.

However, the above methods cannot directly be applied to reinforcement learning cases due to two reasons. First, the label-based dataset inference [42] cannot be implemented in the RL models since their outputs are usually continuous, and they are guided by the rough reward signals instead of the exact labels. Second, the distribution of the offline RL dataset not only depends on the environment but also relies on the strategy of interacting with the environment [17]. Thus, it is challenging to find a proper public dataset in offline RL scenarios. As we delve into Appendix A, it becomes evident that the behavior similarity of the DRL models varies across different public training data. Furthermore, the behavior similarity is also influenced by various offline DRL frameworks.

Membership Inference Attack against RL [46], [23], [22]. Several membership inference attacks exist against DRL, which seem to address the problem studied in this paper. Most of them are targeted at the online RL scenes, assuming that the attacker owns the environment. Thus, they can utilize the environment to collect more data and even manipulate some adversarial states to facilitate the inference.

However, in this paper, we aim at the offline RL cases, which are more challenging since the only thing the auditor can use is the pre-collected dataset. That is, in offline RL scenarios, the existing MIA against RL cannot rely on the environment to generate non-member data.

IV. ORL-AUDITOR

We instantiate Q of Figure 1 with the cumulative reward, which is an intrinsic feature of the dataset and suitable for auditing. Δ is determined by the shadow models trained on the datasets instead of a preset threshold to adapt the distribution of different datasets. The well-designed Q and Δ guarantee the adaptiveness and effectiveness of ORL-AUDITOR.

A. Workflow

For ease of understanding, we refer to the *target dataset* as the dataset to be audited and the *actual dataset* as the dataset used by the suspect model. If the suspect model is trained on the target dataset, the actual dataset is the same as the target dataset, *i.e.*, positive audit result for the suspect model; otherwise, the suspect model does not use the target dataset, *i.e.*, negative audit result for the suspect model. Figure 4 illustrates the workflow of ORL-AUDITOR.

Step 1: Model Preparation (MP). In the left box of Figure 4, the auditor prepares the critic model and the shadow DRL models based on the target dataset, which contains m trajectories T with the length of n_i ($i \in \{1, 2, \ldots, m\}$). The

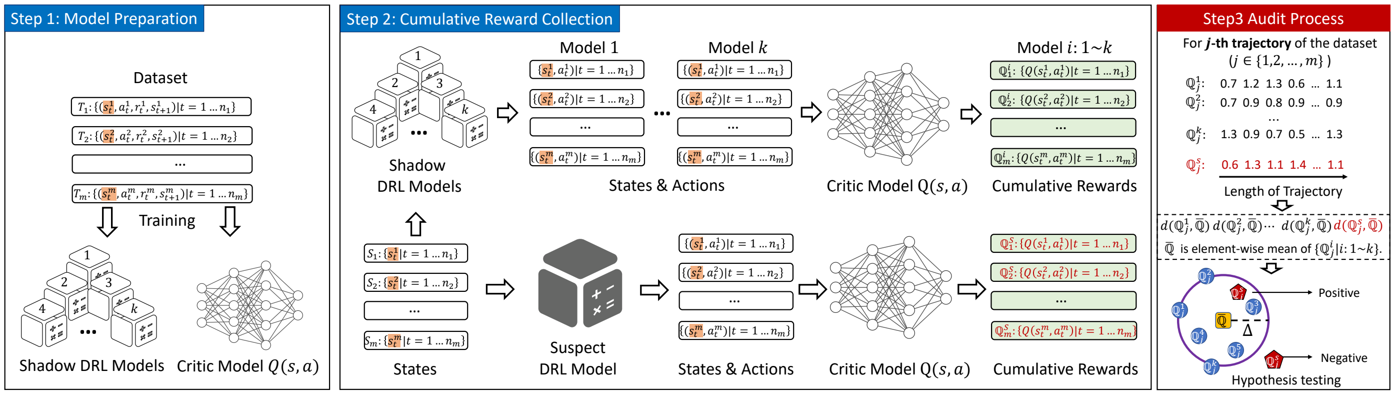


Fig. 4: The workflow of ORL-Auditor contains three steps, *i.e.*, model preparation, cumulative reward collection, and audit process. ORL-Auditor first trains a set of shadow DRL models and a critic model on the target dataset, then collects the cumulative rewards from the state-action pairs of the shadow models and the suspect model. Finally, ORL-Auditor audits every trajectory based on hypothesis testing.

critic model is optimized to estimate the cumulative reward of each state-action pair. For each trajectory in the dataset, a series of predictions for its state-action pairs compose the exclusive feature for auditing. There are two ways to optimize the critic model, *i.e.*, the Monte-Carlo-based (MC-based) and the temporal-difference-based (TD-based) strategies. We adopt the TD-based learning method and explain the reasons in Section IV-B. In addition, the auditor trains a set of shadow models following the model's objective function introduced in Section II with different model initializations.

Step 2: Cumulative Reward Collection (CRC). In the middle box, the shadow models observe the states of the dataset and take actions. For i-th trajectory in the dataset, the auditor records the state s_t^i and the action a_t^i of each shadow model, where a_t^i represents the shadow model's action at the t-th step of trajectory T_i. After finishing the action collection, the auditor obtains the k sets of state-action pairs from the shadow models, representing the learned policies with different initialization and training processes on the target dataset. Using the critic model in Step 1, the auditor calculates the estimations for all state-action records, *i.e.*, the estimated cumulative rewards, which are the samplings of the exact cumulative rewards of the corresponding state-action pairs in the dataset. Similarly, the auditor queries the suspect model with state s_t^i and observes the action a_t^i. The state-action pairs are then put into the critic model and to obtain the estimations for the suspect model.

Step 3: Audit Process (AP). After the above two steps, the auditor obtains the estimated cumulative rewards from the shadow models and the suspect model and then conducts the audit process. For j-th ($j \in \{1, 2, \dots, m\}$) trajectory of the dataset, the auditor collects k series of the estimated cumulative rewards from the shadow models, *i.e.*, $\{\mathbb{Q}_j^i \mid i \in \{1, 2, \dots, k\}\}$, and one from the suspect model, *i.e.*, $\mathbb{Q}_j^s$. ORL-Auditor conducts hypothesis testing based on the distances of $\mathbb{Q}_j^i$ and $\mathbb{Q}_j^s$ from $\bar{\mathbb{Q}}_j$. The auditor can rule out suspicion if $d(\mathbb{Q}_j^s, \bar{\mathbb{Q}}_j)$ is out of the distribution of $\{d(\mathbb{Q}_j^i, \bar{\mathbb{Q}}_j) \mid i \in \{1, 2, \dots, k\}\}$. Otherwise, the auditor will conclude a positive decision, *i.e.*, the suspect model is trained using this trajectory. The auditor repeatedly implements the above processes for other trajectories of the dataset and obtains the final audit report with judgment for all trajectories. We discuss more details of the distance metric and the hypothesis testing in Section IV-C.

B. The Selection of Critic Model

The auditor can use either Monte Carlo (MC) based or Temporal-Difference (TD) based algorithms to train a critic model from the trajectories of the dataset. The main distinction between the two methods lies in their learning targets, which leads to differences in their objective functions. In the case of MC-based methods, the learning target G is the empirical cumulative rewards from the dataset.

$$G(s_t, a_t) = r_t + \gamma r_{t+1} + \ldots + \gamma^{H-1} r_H,$$

where $G(s_t, a_t)$ represents the exact cumulative reward from (s_t, a_t) to the terminal time step H of one trajectory. The discount factor γ is applied to discount future rewards. The critic model is trained by minimizing the following objective.

$$\mathbb{E}_{(s_t, a_t, r_{t+1}, s_{t+1}) \sim D}\left[\left(G\left(s_t, a_t\right) - Q_\theta\left(s_t, a_t\right)\right)^2\right].$$

For TD-based methods, the learning target changes to the expected cumulative reward in a heuristic form, *i.e.*, $r_t + \gamma Q\left(s_{t+1}, a_{t+1}\right)$. Thus, the critic model is trained by minimizing the following loss function.

$$\mathbb{E}_{\left(s_t, a_t, r_{t+1}, s_{t+1}\right) \sim D}\left[\left(r_{t+1} + \gamma Q_{\theta'}\left(s_{t+1}, a_{t+1}\right) - Q_\theta\left(s_t, a_t\right)\right)^2\right],$$

where the critic model starts with arbitrary initialization θ. Then, it repeatedly evaluates $Q_\theta\left(s_t, a_t\right)$, obtains a reward r_{t+1}, and updates the weights. The θ' is a snapshot of θ and copies from θ every few updates of θ. The MC-based method utilizes the exact cumulative rewards from the dataset to train the critic model, resulting in an unbiased prediction. It also has strong convergence properties due to the stationary of G_t. However, it cannot be applied to situations where the collected data is truncated, and all trajectories in the dataset must be completed. In practice, many sequential decision-making tasks usually have long or infinite time steps. Thus, the dataset provider segments the interaction record into trajectories by a preset maximum length. The TD-based method tackles the limitation

Algorithm 1 Workflow of ORL-AUDITOR

```
Input: Dataset $D$, suspect model $\pi_s$, number of shadow models $k$, significance level $\alpha$
Output: The trajectory-level audit report
1: // Step 1: Model Preparation
2: Train shadow models $\{\pi_i \mid i = 1, \ldots, k\}$ and critic model
3: // Step 2: Data Preparation
4: for each model $\pi$ in $\{\pi_i \mid i = 1, \ldots, k\} \cup \{\pi_s\}$ do
5:     Query $\pi$ by states $s \in D$ and obtain the actions.
6:     Evaluate each $(s, a)$ pair based on the critic model $Q$.
7:     Record the cumulative reward in sequential form $\{\mathbb{Q}_j \mid j = 1, \ldots, m\}$.
8: end for
9: // Step 3: Audit Process
10: audit_report = []
11: for each trajectory in $\{T_j \mid j = 1, \ldots, m\}$ do
12:     Calculate the element-wise mean $\bar{\mathbb{Q}}_j$ of $\{\mathbb{Q}^i_j \mid i = 1, \ldots, k\}$
13:     Measure the $d(\mathbb{Q}_j, \bar{\mathbb{Q}}_j)$ of each $\mathbb{Q}^i_j$ and $\mathbb{Q}^s_j$ from $\bar{\mathbb{Q}}_j$.
14:     // Hypothesis testing
15:     From $\{d^i \mid i = 1, \ldots, k\}$ and $d^s$, decide whether the suspect model $M_s$ pirates $T_j$ with significance level $\alpha$.
16:     audit_report.append($j$-th audit result)
17: end for
18: Return audit_report
```

of the MC-based algorithm and can learn from incomplete sequences. Nevertheless, due to the heuristic learning process, the TD-based method has some bias and is more sensitive to model initialization. Therefore, we choose the element-wise mean of the shadow models' cumulative rewards $\bar{\mathbb{Q}}$ as the auditing directrix in Section IV-A instead of relying solely on the critic model's predictions to compensate for the shortages of TD-based methods.

C. The Details of Audit Process

In the audit process, the choice of distance metric and the hypothesis testing method play a critical role in ORL-AUDITOR's performance. A proper metric is sensitive to the deviations between the estimated cumulative rewards, which can facilitate the hypothesis testing. A suitable hypothesis testing method can provide precise results with high confidence.

Distance Metric. We consider three types of distance metrics, *i.e.*, ℓ_p norm, Cosine distance, and Wasserstein distance. ℓ_p norm is a popular method of measuring the distance between vectors, *i.e.*, the sum of the absolute difference of the components of the vectors. In the RL scene, the states and actions are sequential data, meaning the distance metric should measure both the value and the position deviation of the cumulative rewards. However, ℓ_p norm may fail to reflect the difference from the sequence aspect of the same set of values. Cosine distance is a derivative of Cosine similarity, defined as the cosine of the angle between two vectors. Cosine distance embodies the difference from both the value and position aspects of the vectors. However, Cosine distance normalizes the inner product using the two vectors' norm, which weakens the numerical differences between the cumulative rewards. The Wasserstein distance, *a.k.a.* earth mover's distance (EMD), is a metric of the difference between two probability distributions over a region [53]. It can be defined as follows.

$$l_1(u, v) = \inf_{\pi \in \Gamma(u,v)} \int_{\mathbb{R}\times\mathbb{R}} |x - y| \mathrm{d}\pi(x, y),$$

where $\Gamma(u, v)$ is the set of distributions on $\mathbb{R} \times \mathbb{R}$ whose marginals are u and v on the first and second factors respectively. Wasserstein distance fits well with audit requirements, reflecting numerical and positional deviations of the cumulative rewards. Thus, we set Wasserstein distance by default and compare different distance metrics in Section V.

Hypothesis Testing. After the selection of the distance metric, the auditor proceeds to hypothesis testing with the distances of $\mathbb{Q}^i_j$ and $\mathbb{Q}^s_j$ from $\bar{\mathbb{Q}}_j$.

$$H_0 : d(\mathbb{Q}^s_j, \bar{\mathbb{Q}}_j) \text{is not an outlier.}$$
$$H_1 : d(\mathbb{Q}^s_j, \bar{\mathbb{Q}}_j) \text{is an outlier.}$$

An intuitive method is to leverage the 3σ principle, *i.e.*, the normal samples should be distributed within the range of three times the standard deviation σ_d from the mean μ_d. The 3σ principle is an efficient hypothesis testing method, yet the mean μ_d is easily misled by outliers.

Compared to the 3σ principle, Grubbs' test [24] is a more robust hypothesis testing method for detecting single outliers in univariate datasets. If the Grubbs' test statistic of $d(\mathbb{Q}^s_j, \bar{\mathbb{Q}}_j)$ exceeds the threshold derived on the significance level, the auditor can claim $d(\mathbb{Q}^s_j, \bar{\mathbb{Q}}_j)$ deviate from the mean value, *i.e.*, reject H_0 and output negative audit result. For a set of samples $\{d_i \mid i = 1, 2, \ldots, n\}$, Grubbs' Test locates the outlier by the procedures.

1) Calculate the mean μ_d and standard deviation σ_d.
2) Calculate the Grubbs' test statistic by $G = \frac{|d(\mathbb{Q}^s_j, \bar{\mathbb{Q}}_j) - \mu_d|}{\sigma_d}$.
3) If $G > \frac{n-1}{\sqrt{n}}\sqrt{\frac{t^2_{\alpha/(n),n-2}}{n-2+t^2_{\alpha/(n),n-2}}}$, H_0 is invalid, *i.e.*, the suspect model is not trained by this trajectory. In the above inequation, $t^2_{\alpha/(n),n-2}$ represents the upper critical value in the t-distribution when the degree of freedom is $n-2$, and the significance level is $\frac{\alpha}{n}$.

Both hypothesis testing methods are based on the assumption that the distance values follow the Gaussian distribution. Thus, ORL-AUDITOR needs to pre-check that the distance values of the shadow models satisfy the Gaussian distribution. We adopts the Anderson-Darling test [58] since it fits the scenarios where the auditor has a small number of samplings, and the actual distribution is unknown. In the evaluation, all the distance values of the shadow models can pass the Anderson-Darling test due to the randomness of the models' initialization and training. Then, ORL-AUDITOR conducts the hypothesis testing.

V. EVALUATION

We first introduce the tasks and the experimental setup in Section V-A. We validate the effectiveness of ORL-AUDITOR on Behavior Clone and three offline DRL models, *i.e.*, Batch-Constrained Q-learning (BCQ) [20], Implicit Q-Learning (IQL) [34], and TD3PlusBC [19] in Section V-B.

TABLE I: The Overview of Tasks. The "continuous" and "discrete" illustrates the data type of the state and action with the corresponding number of dimensions in parentheses.

Task Name	State Shape	Action Shape
Lunar Lander	Continuous(6-dim) Discrete(2-dim)	Continuous(2-dim)
Bipedal Walker	Continuous(24-dim)	Continuous(4-dim)
Ant	Continuous(111-dim)	Continuous(8-dim)

TABLE II: The details of the offline DRL datasets

Task Name	Number of Transitions	Dataset Name	Number of Trajectories	Length of Trajectory
Lunar Lander	5×10^5	1171	2175	229.83±83.51
		2094	578	864.19±231.88
		4496	1252	399.30±240.88
		6518	1878	266.13±99.65
		9906	1566	319.21±231.06
Bipedal Walker	10^6	0841	1019	981.03±190.79
		1203	1027	973.07±118.42
		2110	877	1139.55±151.10
		3813	887	1126.63±379.05
		6558	1041	959.77±146.13
Ant	2×10^6	2232	2093	955.46±177.72
		3569	3497	571.66±375.40
		4603	2096	954.01±175.82
		5766	2217	901.84±236.93
		7490	2103	951.02±187.93

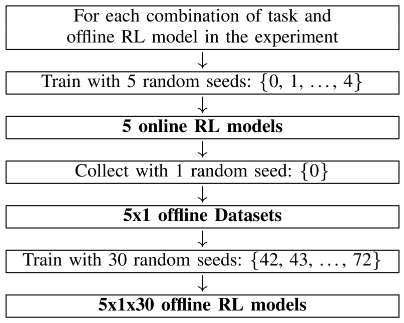
Then, we visualize the cumulative rewards by t-SNE [61] to demonstrate that the cumulative rewards are intrinsic and stable features for dataset auditing in Section V-C. After that, we further evaluate the impact of three factors on ORL-AUDITOR, *i.e.*, the number of shadow models, the significance level in hypothesis testing, and the trajectory size in Section V-D. Finally, we utilize ORL-AUDITOR to audit the open-source datasets from Google [17] and DeepMind [25] in Section V-E.

A. Experimental Setup

Tasks. We adopt Lunar Lander, Bipedal Walker, and Ant tasks in Gym [7], which are widely used in the prior works [9], [29], [47]. The tasks stem from distinct real-world problems, each with numerical vectors containing different physical information, *e.g.*, position, velocity, and acceleration. These tasks involve both discrete and continuous variables in observation and action spaces, with the dimension ranging from low (2-dim) to high (111-dim). We give an overview in Table I and put their details in Appendix B.

Dataset Generation and Offline Model Preparation. To obtain the datasets for tasks in Table I, we adopt the same idea as the existing dataset publishers [25], [17], [51], [1], *i.e.*, training the online RL models in the interactive environment and recording the interactions as the datasets. The datasets consist of numerical vectors. In Lunar Lander, each transition includes state, next state (6-dimensional continuous and 2-dimensional discrete variables), action (2-dimensional continuous variables), and reward (scalar). Therefore, each transition is a 19-dimensional numerical vector. Similarly, the data types of Bipedal Walker and Ant are 53-dimensional and 231-dimensional numerical vectors, respectively. The number of transitions for each task is 5×10^5 (Lunar Lander), 10^6 (Bipedal Walker), and 2×10^6 (Ant).

TABLE III: The main steps in dataset generation and offline model preparation with the details of the input and output.


For each combination of task and offline RL model in the experiment
Train with 5 random seeds: {0, 1, ..., 4}
5 online RL models
Collect with 1 random seed: {0}
5x1 offline Datasets
Train with 30 random seeds: {42, 43, ..., 72}
5x1x30 offline RL models

The offline RL models learn from the datasets. Table III summarizes the whole process. For each task, we use five global random seeds to train five online models separately. We collect the datasets from five online models with random seed 0, where every online model only generates one dataset. For ease of reading, the datasets share the same name with their online models. We train thirty offline DRL models for every dataset with distinct global random seeds in initialization and optimization processes. All the online and offline models are implemented by open-source RL libraries [52], [55] with default hyperparameter settings.

Critic Model. We adopt the fully connected neural network as the critic model, which has four hidden layers with 1024 neurons on each layer. We optimize the critic model following the TD-based method in Section IV-B by Adam optimizer with a learning rate of 0.001 and a mini-batch size of 4096. The entire training takes 150 epochs, and the learning rate decays to half every 50 epochs.

Evaluation Metrics. Recalling ORL-AUDITOR's application scenario in Figure 3, for a single suspect model, the audit accuracy can well characterize the performance of ORL-AUDITOR, *i.e.*, the ratio of the number of correctly auditing trajectory to the total auditing trajectory. In our experiment, the positive models (trained on the target dataset) and the negative models (trained on other datasets) are randomly mixed, where the majority may dominate the accuracy. Thus, we provide the true positive rate (TPR) and the true negative rate (TNR).

Methods. We provide the audit performance of 3σ principle and Grubbs' test with four distance metrics, *i.e.*, ℓ_1 norm, ℓ_2 norm, Cosine distance, and Wasserstein distance.

Competitors. Recalling Section III-B, existing methods [46], [23], [22] are designed for the online reinforcement learning scenes, assuming that the auditor can continuously interact with the environment to obtain new data as the non-member example. Based on the behavioral difference of the model between the member examples and the non-member examples, they build the member inference method to detect whether an example is used to train the suspect model. In the offline scenarios, without access to the environment, the auditor only

TABLE IV: The performance of existing membership inference attack against offline DRL models.

Task Name	Offline Model	Accuracy	
		Training	Test
Lunar Lander	BC	50.09±0.68	48.41±1.87
	BCQ	49.84±1.39	47.69±1.45
	IQL	49.88±0.76	47.34±1.83
	TD3PlusBC	50.08±0.92	48.27±1.81
Bipedal Walker	BC	50.00±0.63	46.27±2.42
	BCQ	49.97±0.69	47.38±2.41
	IQL	50.17±0.95	47.19±1.90
	TD3PlusBC	49.87±0.94	45.48±1.46
Ant	BC	50.44±0.64	46.74±2.37
	BCQ	50.22±0.52	45.38±2.16
	IQL	50.33±0.35	45.89±1.90
	TD3PlusBC	50.13±0.67	45.03±1.55

has the pre-collected target dataset. Thus, we randomly divide the target dataset into two parts and train offline RL models on the subsets separately. Either subset is regarded as the set of non-member examples for the offline RL models trained on the other subset. We adopt the same data augmentation, attack classifier architecture, and hyperparameter settings with [22].

Implementation. Stable-baselines [52] and d3rlpy [55] are used to implement online and offline DRL models separately. All audit methods are realized with Python 3.8 on a server with 8 NVIDIA GeForce RTX 3090 and 512GB memory.

B. *Overall Audit Performance*

We assess the effectiveness of ORL-AUDITOR across twelve combinations of three tasks and four models. Furthermore, we present an evaluation of the efficacy of the competitors on offline DRL models.

Setup. From Table III, we train 30 offline RL models for each dataset and obtain 150 offline DRL models for every experimental setting. We audit the 5 datasets separately, where the auditor randomly selects 15 models from the target dataset as the shadow models, and the remaining 15 models along with the 120 models from other datasets are the positive and the negative suspect models. For the target dataset, we randomly select 50 auditing trajectories to audit. Since the unbalanced amount of the positive and the negative models, we report the aggregated mean with a standard deviation of both TPR and TNR for each setting in Table V. Each pair of TPR and TNR in Table V is derived from the diagonal and non-diagonal values of the corresponding heatmap. We show the audit result by 3σ principle in Table VII of [12]. The competitors' performance is in Table IV, where the values of mean and standard variation are calculated by repeating experiment 10 times.

Observations. We have the following observations from Table IV and Table V. 1) Most TPR and TNR values are higher than 95%, meaning that ORL-AUDITOR is a valid solution to audit the learned dataset of the offline DRL models. For instance, all results for ORL-AUDITOR with ℓ_1 norm are beyond 94% across the experiment settings.

2) ORL-AUDITOR obtains different audit accuracy over four distance metrics. The audit effectiveness with ℓ_1 norm and Wasserstein distance is better than that of ℓ_2 norm and Cosine distance. In Table V, ORL-AUDITOR with Wasserstein distance always performs the best or the second place. And results of ℓ_2 norm are usually behind the other three distance metrics. Recalling Section IV-C, Wasserstein distance characterizes both the numerical and the positional deviations of the cumulative rewards, which is more sensitive. Since the numerical differences between the cumulative rewards are slight, *e.g.*, from 0.01 to 0.1 in our experiment, ℓ_2 norm may undercut these small but potential differences.

3) The accuracy of the audit as determined by Grubbs' test outperforms that of the 3σ principle. The 3σ principle is an empirical method, which is easily misled by the outlier cumulative rewards of the shadow models. Recalling Section IV-C, Grubbs' test first calculates the statistic G and compares G with an adaptive threshold, where the number of samples is also considered in the hypothesis testing.

4) Without the new data from the environment, the effectiveness of the existing membership inference methods is attenuated. From one perspective, the similarity between sub-datasets splited from the same dataset can result in the trained RL models exhibiting undifferentiated behavior, making it difficult to effectively distinguish between members and non-members. On the other hand, when considering the results presented in Figure 10, we conclude that the actions of RL models should not be directly utilized as the foundation for membership inference.

C. *Visualization of Cumulative Rewards*

To further explain the audit results in Section V-B, we analyze the cumulative rewards from the shadow models and the suspect models, *i.e.*, $\mathbb{Q}_j^i$ and $\mathbb{Q}_j^s$, by using t-SNE [61].

Setup. The caption of each plot in Figure 5 indicates the used task and offline DRL model. Each point in the plots shows the visualization of a single $\mathbb{Q}_j^i$ (positive) or $\mathbb{Q}_j^s$ (negative). In a single plot, we demonstrate the results of three trajectories from each tasks' first datasets. For instance, the target dataset of the plot titled "Lunar Lander, BC" is dataset "1171" in Table II. The thirty positive points for each trajectory are collected from the shadow models trained on dataset "1171", while the thirty negative points are randomly sampled from the shadow models from the other four datasets.

Observations. From Figure 5, we have the following observations. 1) For a trajectory of the target dataset, the cumulative rewards from the shadow models and the suspect models are clearly divided into different groups, meaning that the critic model well reflects the differences in the models' actions. Thus, the cumulative reward generated by the critic model is a qualified post-event fingerprint for trajectory-level auditing.

2) The distribution of points varies on the different trajectories. For example, trajectory 1 from the Lunar Lander dataset is harder to cluster than the other two trajectories. We speculate that this is because trajectory 1 represents a basic policy, *e.g.*, a local optimum policy to fire the lander's thrusters all the way, and similar trajectories exist in the other four datasets. Due to the non-uniqueness of the optimal strategy in RL problems and the impact of randomness in the model training process, the collected trajectories have unique characteristics. Thus, other trajectories' cumulative rewards are clearly divided.

TABLE V: The TPR and TNR results based on Grubbs' test. The mean and standard deviation of TPR and TNR in each row represent the audit results for one combination of task and model by four distance metrics. Bold indicates the highest sum of TPR and TNR, *i.e.*, accuracy, in a row. Each pair of TPR and TNR is derived from the diagonal and non-diagonal values of the corresponding heatmap in Figure 11, Figure 12 and Figure 13 of [12].

Task Name	**Offline Model**	**L1 Norm**		**L2 Norm**		**Cosine Distance**		**Wasserstein Distance**	
		TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR
Lunar Lander	BC	**99.01±0.46**	**100.00±0.00**	96.96±0.73	100.00±0.00	96.93±0.77	100.00±0.00	98.40±0.74	99.94±0.16
	BCQ	**98.29±1.14**	**100.00±0.00**	96.03±1.15	100.00±0.00	95.97±1.07	99.99±0.04	97.57±1.17	99.91±0.14
	IQL	**98.61±1.51**	**99.91±0.32**	97.52±2.51	99.97±0.12	97.49±2.56	99.92±0.19	98.32±1.79	97.10±5.66
	TD3PlusBC	**98.29±2.04**	**99.48±0.79**	96.35±3.01	99.89±0.22	96.27±3.16	99.91±0.23	98.53±1.25	95.59±3.77
Bipedal Walker	BC	99.20±1.47	100.00±0.00	98.40±2.70	100.00±0.00	98.56±2.68	100.00±0.00	**99.31±1.32**	**100.00±0.00**
	BCQ	99.52±0.77	100.00±0.00	98.16±2.89	100.00±0.00	99.87±0.15	100.00±0.00	**99.89±0.13**	**100.00±0.00**
	IQL	95.10±7.41	100.00±0.00	95.04±5.45	100.00±0.00	**99.84±0.32**	**100.00±0.00**	95.01±6.72	100.00±0.00
	TD3PlusBC	**99.36±1.28**	**94.77±19.42**	97.15±5.71	93.36±21.46	96.96±5.82	91.98±21.75	98.08±3.84	88.26±25.34
Ant	BC	97.42±1.66	99.94±0.11	96.48±1.66	99.90±0.36	99.20±1.08	85.66±28.23	**98.00±1.19**	**99.92±0.14**
	BCQ	97.17±2.96	99.80±0.43	95.68±2.54	99.84±0.43	99.66±0.43	86.70±26.89	**98.67±1.65**	**99.79±0.46**
	IQL	97.20±2.33	99.66±0.73	96.61±2.50	99.69±0.59	99.57±0.79	86.25±27.90	**99.36±0.42**	**99.63±0.78**
	TD3PlusBC	98.53±1.80	99.18±1.72	97.17±1.79	99.35±1.74	99.72±0.40	87.79±26.43	**99.25±1.24**	**99.14±1.81**

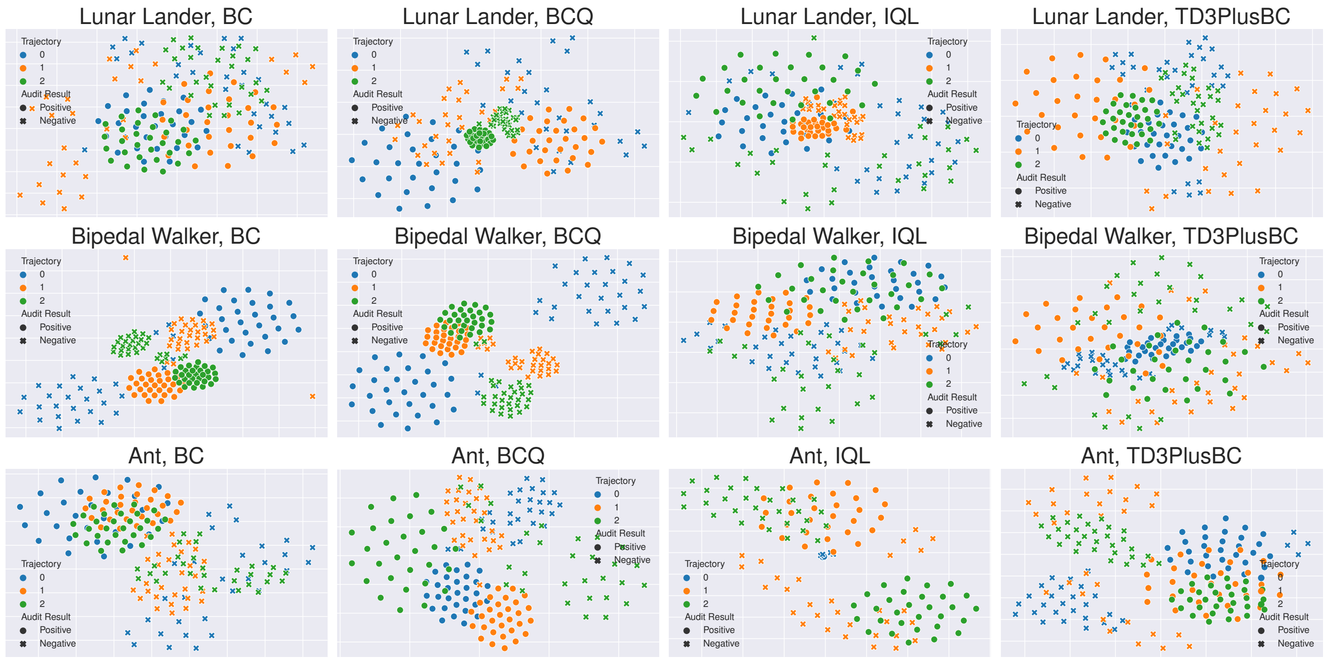


Fig. 5: Visualization of cumulative rewards by t-SNE. The caption of each plot demonstrates the offline DRL model's type and task. In a single plot, we randomly select three trajectories from the first dataset for the task, *i.e.*, Lunar Lander dataset 1171, Bipedal Walker dataset 0841, and Ant dataset 2232 in Table II, and then show the cumulative rewards from 30 positive models and 30 negative models for each trajectory.

D. Hyperparameter Study

We extend our assessment to scrutinize three pivotal determinants that impact the pragmatic integration of ORL-AUDITOR. Specifically, we consider the amount of shadow models, the level of significance in hypothesis testing, and the magnitude of the trajectory size.

Impact of Shadow Models' Amount. The shadow models' amount is changed to 9 and 21 with the other settings the same as Section V-B. Figure 6 shows the value change of TPR and TNR compared with that of 15 shadow models. Each figure's title illustrates the settings of the model and the task. Also, we provide the detailed results in Table VIII (9 Shadow Models) and Table IX (21 Shadow Models) of [12].

From Figure 6, we have the following observations. 1) The audit accuracy increases with a larger amount of shadow models. Since the values of shadow models are the multi-sampling of the true value $Q(s, a)$ of the dataset, the mean and standard deviation will be more precise with more shadow models. For example, ORL-AUDITOR suffers an obvious TPR decline (more than 30%) with 9 shadow models. Since the insufficient knowledge about the diversity of models trained on the target dataset, the auditor easily misclassifies the positive models to the negative group.

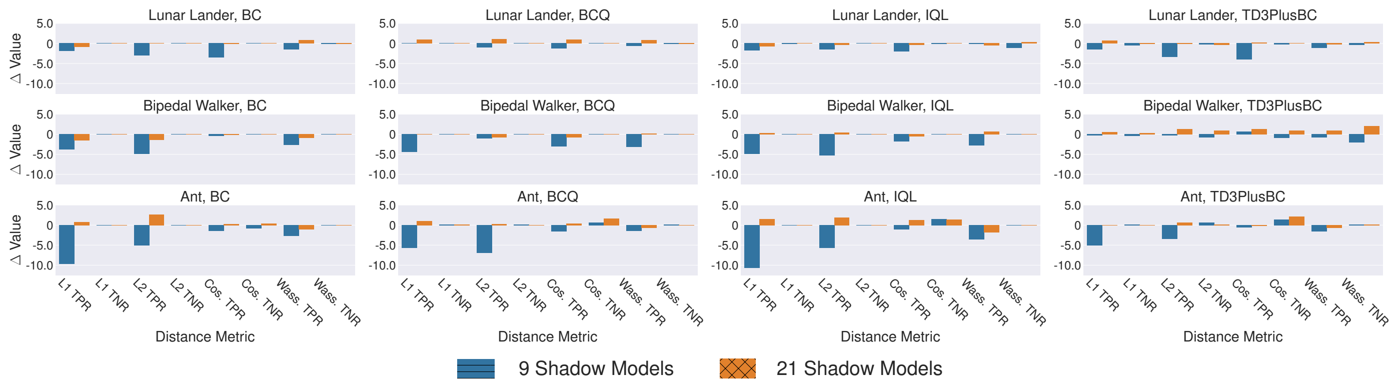


Fig. 6: Impact of shadow models' amount. The change value of TPR and TNR when the number of shadow models varies to 9 and 21 compared to the default 15 shadow models. The caption of each plot demonstrates the offline DRL model's type and task. The x labels display the four distance metrics. The y labels show the absolute fluctuating values of TPR and TNR.

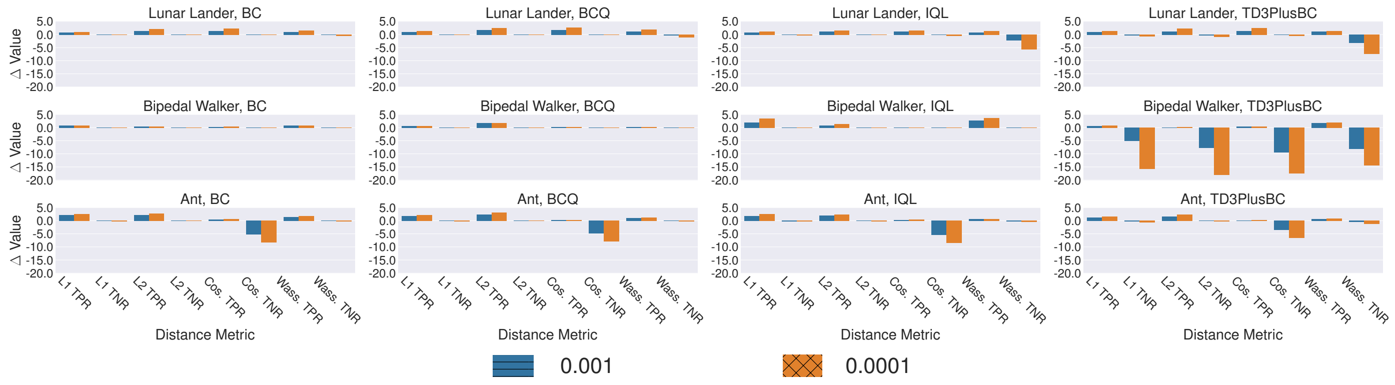


Fig. 7: Impact of the significance level. The change value of TPR and TNR when the significance level varies to 0.001 and 0.0001 compared to the default 0.01. The caption of each plot demonstrates the offline DRL model's type and task. The x labels display the four distance metrics. The y labels show the absolute fluctuating values of TPR and TNR.

2) There exists a saturation point for audit accuracy with the expansion of shadow models. When the shadow models' amount rises from 15 to 21, the TPR usually increases since the auditor observes more possible cumulative rewards originating from the model trained on the target dataset. The value changes slightly in most plots, meaning that similar cumulative rewards appear in the shadow model set and the diversity does not increase significantly compared to that of 15 shadow models. Therefore, excessive shadow models are unnecessary, and the auditor needs to burden more training overhead.

Impact of Significance Level. The significance level represents the auditor's confidence in the audit results. In Section V-B, we adopt the significance level $\alpha = 0.01$, meaning that the auditor has 99% confidence in the judgments made. Generally speaking, the significance level represents the maximum audit capacity of ORL-AUDITOR instead of a hyperparameter setting since it is an audit requirement by the dataset owner. We demand the auditor to output a more confident judgment, where the error possibility should be limited to 1‰ and 0.1‰, *i.e.*, $\alpha = 0.001$ and $\alpha = 0.0001$. Figure 7 shows the value change of TPR and TNR compared with that when significance level $\alpha = 0.01$. The used offline DRL model and task is shown in each figure's title. The detailed results between every two datasets are in Table X ($\alpha = 0.001$) and Table XI ($\alpha = 0.0001$) of [12].

From Figure 7, we have the following observations. 1) For a complicated task, we recommend the auditor to select a large significance level for ORL-AUDITOR. The task's complexity affects the minimum significance level of ORL-AUDITOR. For example, TPR and TNP change a little on the Lunar Lander task when the significance level reduces to 0.001, while they highly shrink on the Ant task. From Table I, Ant's state and action space are larger than that of Lunar Lander. When the auditor leverages the critic model to compress each model's state and action pair into a scalar, the deviation between Q_j^i and Q_j^s (recalling Figure 4) on the Ant task is more imperceptible.

2) For the suspect models with low performance, ORL-AUDITOR should adopt a large significance level to guarantee audit accuracy. For instance, in the figure titled with "Bipedal Walker, TD3PlusBC", all TNR results from four distance metrics decrease when α reduces to 0.001 and 0.0001. In the experiment, most of the TD3PlusBC models' performance on the Bipedal Walker task is around -100, meaning that the TD3PlusBC models do not fully master the knowledge of the dataset. Thus, the dataset features reflected in their behavior are ambiguous, which weakens the difference between positive and negative samples. Meanwhile, the confidence interval, *i.e.*, Δ in Figure 1, expands with a lower significance level. For the above two reasons, the TNR results of the TD3PlusBC models on the Bipedal Walker task drop more than 10% compared with these when $\alpha = 0.01$.

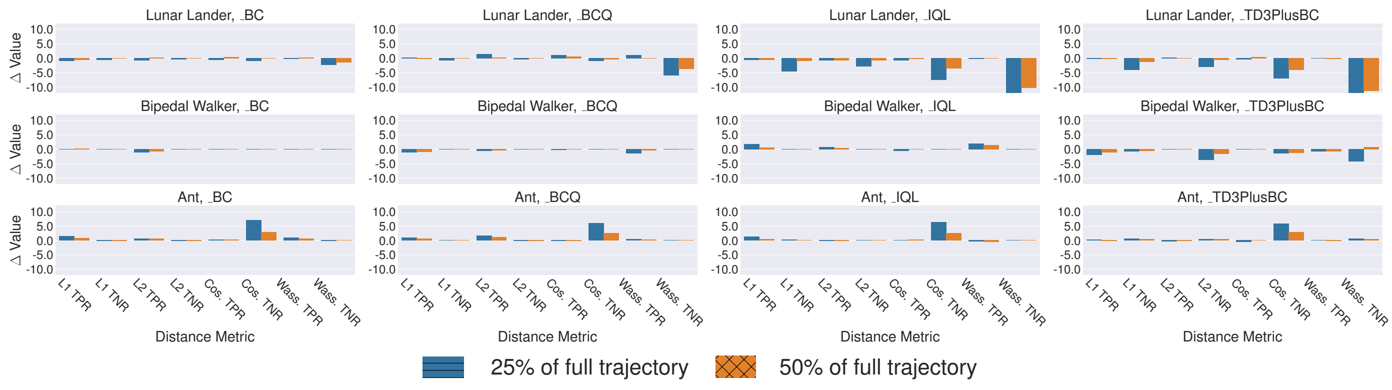


Fig. 8: Impact of the trajectory size. The change value of TPR and TNR when the trajectory size varies to 25% and 50% compared to the entire trajectories (100%). The caption of each plot demonstrates the offline DRL model's type and task. The x labels display the four distance metrics. The y labels show the absolute fluctuating values of TPR and TNR.

From the above analysis, $\alpha = 0.01$ is a safe bound of ORL-AUDITOR, and a lower α may break through the capability boundary of ORL-AUDITOR, inducing the auditor to misclassify the negative model to the positive set.

Impact of Trajectory Size. We investigate the relationship between the trajectory size and audit accuracy. In Section V-B, we adopt the full-length trajectory, meaning that the auditor utilizes all states of each trajectory to query the suspect model and obtains the corresponding actions to conduct the dataset audit. We change the trajectory size to 25% and 50% of the full length with the other settings the same as Section V-B. Figure 8 shows the value change of TPR and TNR compared with that of the full-length trajectory. Each figure's title illustrates the settings of the model and the task. We provide the detailed results in Table XII (25%) and Table XIII (50%) of [12].

From Figure 8, we have the following observations. 1) ORL-AUDITOR tends to achieve higher accuracy with a larger trajectory size. Since the predicted cumulative rewards of state-action pairs from the critic model are the audit basis, a longer trajectory collects more actions from the suspect model to enhance the significance of hypothesis testing. For example, the TNP results decrease at most 13% when ORL-AUDITOR only leverages 25% of the trajectory.

2) It should be noticed that a small trajectory size achieves better results under some tasks. For the Ant task, ORL-AUDITOR auditing with 25% of the full length obtains at most 7% promotion on the TNR results. Based on the analysis of [45], the front states of each trajectory are able to reflect more behavioral information of the model. Thus, in this case, a shorter trajectory truncates the rear state-action pairs, which might be unimportant or even weaken the significance of the hypothesis testing. Exploring effective data auditing with shorter trajectory sizes or even using only the first state of each trajectory would be an interesting future direction.

E. Real-world Application

In this section, we apply ORL-AUDITOR to audit the open-source datasets from DeepMind [25] and Google [17]. We choose the "halfcheetah" task published by both, where the operator controls a 2-dimensional cheetah robot consisting of 9 links and 8 joints connecting them (including two paws) to make the cheetah run forward (right) as fast as possible. The details of the halfcheetah dataset and the offline DRL models are in Table VI and Table VII. All experimental settings are consistent with these in Section V-B.

TABLE VI: The details of the HalfCheetah dataset

Number of Transitions	Dataset Name	Number of Trajectories	Length of Trajectory
10^6	D4RL Expert	1001	998.00 ±0.06
10^6	D4RL Medium	1001	997.90 ±3.13
10^6	D4RL Random	1001	998.00±0.00
3.003×10^5	RL Unplugged	300	1001.00±0.00

TABLE VII: The details of the models trained on the HalfCheetah dataset. The model performance shows the cumulative reward for 10 separate evaluations.

Offline Model	Dataset Name	Model Performance
BC	D4RL Expert	12620.94±307.84
	D4RL Medium	4223.77±134.67
	D4RL Random	-0.33±0.24
	RL Unplugged	-427.50±113.42
BCQ	D4RL Expert	10974.19±842.10
	D4RL Medium	4765.24±98.75
	D4RL Random	-1.13±0.43
	RL Unplugged	-421.91±212.36
IQL	D4RL Expert	10163.20±1106.70
	D4RL Medium	4808.11±46.99
	D4RL Random	1649.55±518.47
	RL Unplugged	-378.74±151.65
TD3PlusBC	D4RL Expert	12712.69±383.33
	D4RL Medium	4969.74±56.31
	D4RL Random	1046.23±226.61
	RL Unplugged	-181.50±205.29

Observations. From Table VIII, we have the following observations. 1) ORL-AUDITOR can be effective in real-world applications. The TPR and TNR of ORL-AUDITOR exceed 95% with ℓ_1 norm and Wasserstein distance, meaning that ORL-AUDITOR remains valid for the existing open-source datasets. 2) Wasserstein distance has stable performance on the experimental and the real-world datasets. The overall accuracy of ORL-AUDITOR with Wasserstein distance are all higher than the other three metrics.

TABLE VIII: The TPR and TNR results on the Half Cheetah task. The mean and standard deviation of TPR and TNR in each row represent the audit results for one combination of task and model by four distance metrics. Bold indicates the highest sum of TPR and TNR, *i.e.*, accuracy, in a row.

Task Name	**Offline Model**	**L1 Norm**		**L2 Norm**		**Cosine Distance**		**Wasserstein Distance**	
		TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR
Half Cheetah	BC	96.07±3.15	100.00±0.00	96.07±2.34	100.00±0.00	99.80±0.35	68.62±42.47	**98.47±1.13**	**100.00±0.00**
	BCQ	95.37±0.55	100.00±0.00	95.83±1.20	100.00±0.00	99.57±0.47	70.14±41.14	**97.47±1.35**	**100.00±0.00**
	IQL	95.47±0.77	100.00±0.00	95.68±1.02	100.00±0.00	99.78±0.23	71.38±41.05	**97.12±2.70**	**100.00±0.00**
	TD3PlusBC	95.00±2.87	100.00±0.00	95.50±1.99	100.00±0.00	99.87±0.16	70.57±40.85	**98.27±1.09**	**100.00±0.00**

TABLE IX: The TPR and TNR results of ORL-AUDITOR against model ensemble ($K = 5$). The mean and standard deviation of TPR and TNR in each row represent the audit results for one combination of task and model by four distance metrics.

Task Name	**Offline Model**	**L1 Norm**		**L2 Norm**		**Cosine Distance**		**Wasserstein Distance**	
		TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR
Lunar Lander	BC	100.00±0.00	100.00±0.00	99.20±0.98	100.00±0.00	99.20±0.98	100.00±0.00	99.60±0.80	99.90±0.44
	BCQ	99.60±0.80	100.00±0.00	98.00±2.19	100.00±0.00	98.00±2.19	100.00±0.00	99.60±0.80	100.00±0.00
	IQL	100.00±0.00	99.90±0.44	99.20±0.98	100.00±0.00	99.60±0.80	99.90±0.44	99.60±0.80	97.60±4.27
	TD3PlusBC	100.00±0.00	99.30±0.95	99.60±0.80	99.90±0.44	99.60±0.80	99.80±0.60	99.60±0.80	95.80±3.57
Bipedal Walker	BC	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
	BCQ	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
	IQL	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
	TD3PlusBC	100.00±0.00	94.90±19.07	100.00±0.00	93.80±21.63	100.00±0.00	92.70±21.62	100.00±0.00	89.20±23.94
Ant	BC	99.60±0.80	100.00±0.00	99.60±0.80	99.90±0.44	99.60±0.80	83.20±31.99	99.20±1.60	100.00±0.00
	BCQ	100.00±0.00	99.70±0.71	99.60±0.80	99.80±0.60	100.00±0.00	85.70±28.31	100.00±0.00	99.70±0.71
	IQL	100.00±0.00	99.80±0.60	99.20±0.98	99.70±0.71	99.20±0.98	86.80±28.32	100.00±0.00	99.80±0.60
	TD3PlusBC	99.60±0.80	99.30±1.82	100.00±0.00	99.40±2.20	100.00±0.00	87.80±25.87	99.60±0.80	98.50±3.79
Half Cheetah	BC	85.00±25.98	100.00±0.00	84.50±25.71	100.00±0.00	94.00±10.39	67.50±43.20	87.00±21.38	100.00±0.00
	BCQ	91.00±15.59	100.00±0.00	89.00±16.76	100.00±0.00	95.00±8.66	67.17±42.30	93.00±12.12	100.00±0.00
	IQL	90.00±12.81	100.00±0.00	86.50±16.70	100.00±0.00	94.50±9.53	71.00±41.37	91.50±12.52	100.00±0.00
	TD3PlusBC	61.50±20.32	100.00±0.00	77.00±19.42	100.00±0.00	95.00±8.66	65.67±41.28	52.00±33.26	100.00±0.00

VI. ROBUSTNESS

A. Ensemble Architecture

To hinder the audit of a dataset, an adversary may utilize state-of-the-art membership inference defense strategies proposed in recent research works [59], [30]. These defense strategies aim to mitigate the influence of a member example on the behavior of a machine learning model. Based on the idea of model ensemble. In particular, [59], [30] proposed to split the training set into several subsets and train sub-models on each of these subsets. Then, when an auditor uses an example from the target dataset to query a suspect model, the adversary aggregates the outputs of the sub-models that have not been trained on this example.

Setup. The number of divided subsets, denoted by K, represents a crucial hyperparameter for ensemble-based methods, as discussed in [59], [30]. Considering the analysis conducted in these studies, as well as the size of the offline RL datasets, we have established $K = 5$ for the present investigation. All other experimental settings remain unchanged from those described in Section V-B, and the corresponding audit outcomes are presented in Table IX. The results between every two datasets are in Figure 14 (Lunar Lander), Figure 15 (Bipedal Walker), Figure 16 (Ant), and Figure 17 (Half Cheetah) of [12].

Observations. We conclude the following observations based on the above results. 1) Even when faced with ensemble architecture, ORL-AUDITOR maintains a high level of audit accuracy. As shown in Table IX, both TPR and TNR consistently exceed 80%. As described in Section IV-A, ORL-AUDITOR uses predicted cumulative rewards from the critic model as the basis for auditing. During training, the critic model captures the overall features of the dataset distribution, instead of memorizing features from individual samples. Since the ensemble model is trained on the target dataset, its behavior embeds the distribution characteristics of the dataset, which ORL-AUDITOR can detect.

2) The use of ensemble architecture may result in a decrease in model performance for certain tasks. For instance, when BCQ models learn from the Ant dataset "3569", the mean values of cumulative reward decrease significantly. Furthermore, due to the sub-models being trained on subsets of data, they only fit a partial dataset's distribution. Consequently, when applying the model ensemble to practical scenarios, the standard deviations of the models' performance are large.

B. Action Distortion

The suspect models may perturb the actions, *i.e.*, changing the original models' outputs, to conceal its training dataset in

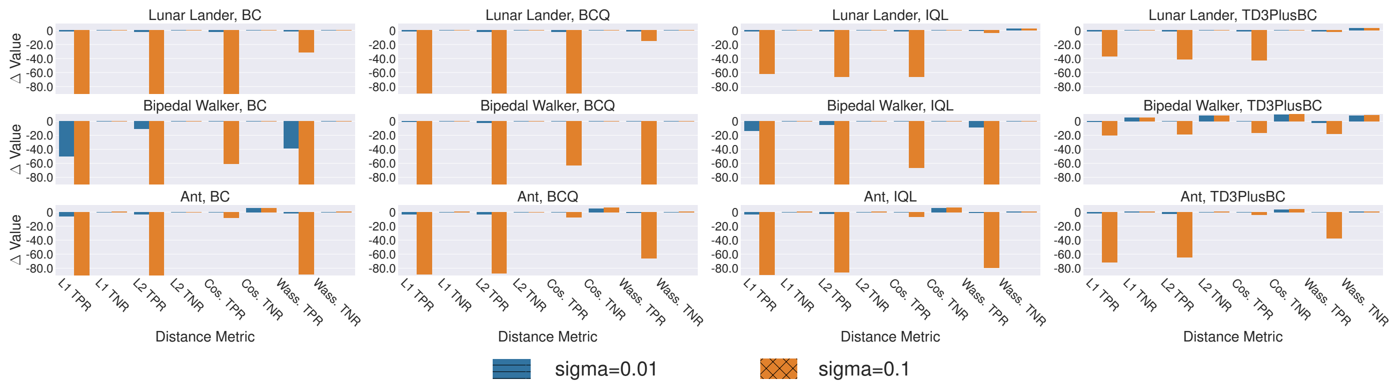


Fig. 9: Robustness against action distortion. The change value of TPR and TNR when the suspect model adds Gaussian noise parameterized with $(\mu = 0, \sigma = 0.01)$ and $(\mu = 0, \sigma = 0.1)$ to its output. The caption of each plot demonstrates the offline DRL model's type and task. The x labels display the four distance metrics. The y labels show the absolute fluctuating values.

practice. The action distortion mechanism should be stealthy and cannot be detected by the auditor easily. Considering that the DRL models are usually applied to real-world decision-making tasks, such as self-driving cars and industry automation [43], [27], the naturally distortion is often modeled as the Gaussian noise. For example, thermal noise, which is caused by the random motion of electrons in a conductor, can be modeled as a Gaussian noise with a constant power spectrum [2]. In addition, Gaussian noise is easy to manipulate mathematically. For ease of evaluating the effects of different distortion intensities, all dimensions of the models' action space are normalized into $[-1, 1]$. Then, we utilize Gaussian noise with mean $(\mu = 0)$ and standard deviation $(\sigma = 0.1)$ and $(\sigma = 0.01)$ to represent the two levels of distortion.

Setup. Figure 9 depicts the impact of "with" or "without" the action distortion. The information about the used offline DRL model and task is shown in each figure's title. The x-axis indicates the four metrics, and the y-axis is the absolute value change. The detailed results between every two datasets are in Figure 18, Figure 19 (Lunar Lander), Figure 20, Figure 21 (Bipedal Walker), Figure 22, and Figure 23 (Ant) of [12].

Observations. We conclude the following observations based on the above results. 1) ORL-AUDITOR is able to resist the potential action distortion from the suspect model, especially with the Cosine metric. From Figure 9, the TPR and TNR vary slightly across most of the settings with weak noise, where the maximum accuracy attenuation is within 3% for Cosine distance. We speculate that Cosine distance has a noise suppression ability when calculating the inner product of two series of cumulative rewards. Also, the weak noise may facilitate the dataset auditing since it will move the negative samples farther away from the positive set.

2) ORL-AUDITOR with a single distance metric faces limitations for heavy distortion. The TPR of ORL-AUDITOR suffers an obvious decline with strong noise. Since the strong distortion thoroughly changes the distribution of the models' actions, the cumulative rewards of the suspect model trained on the target dataset are different from those of the auditor's shadow models. In this case, the auditor cannot identify the positive models from the negative just by a single kind of distance metric. Cosine distance is good at discriminating the positive models (results in the diagonal), and Wasserstein distance is proper for the negative models (results in the non-diagonal). Thus, for strong distortion, the combination of multiple distance metrics can enhance the auditing robustness of ORL-AUDITOR. In addition, we should note that the models' normal behavior is also destroyed by the strong distortion. For example, the noise induces the model performance of IQL to decrease up to 25%, and the better the model's quality, the more pronounced the performance drop.

VII. RELATED WORK

Membership and Dataset Inferences. To infer whether an individual data record was used to train the target model, Shokri *et al.* [56] proposed the first practical membership inference strategy by training a number of shadow classifiers to distinguish the target model's outputs on members versus non-members of its training dataset. Since then, researchers have investigated membership inference in various systems, such as machine unlearning [10], facial recognition systems [11], and neural architecture search [28]. Liu *et al.* [40] presenting a first-of-its-kind holistic risk assessment of different inference attacks against machine learning models. Maini *et al.* [42] introduced the definition of dataset inference and designed the first mechanism to identify whether a suspect model copy has private knowledge from the dataset.

Compared with the existing works, ORL-AUDITOR is a well-designed solution built for the offline DRL scenes, which overcomes several new challenges. First, ORL-AUDITOR is a post-event mechanism that can be directly applied to the existing open-source datasets. Second, ORL-AUDITOR does not use any auxiliary datasets.

Knowledge Extraction Against DRL. The DRL models learn from the interaction with the environment, which can be valuable information in some cases, *e.g.*, indoor robot navigation. Pan *et al.* [46] demonstrated such knowledge extraction vulnerabilities in DRL under various settings and proposed algorithms to infer floor plans from some trained Grid World navigation DRL models with LiDAR perception. For exacting the model functionality, Chen *et al.* [9] proposed the first method to acquire the approximation model from the victim DRL. They built a classifier to reveal the targeted black-box DRL model's training algorithm family based only on its predicted actions and then leveraged state-of-the-art imitation learning techniques to replicate the model from the identified algorithm family. Ono *et al.* [44] integrated *differential privacy* [71], [68], [62] into the distributed RL algorithm to defend the extraction. The local models report noisy gradients designed to satisfy local differential privacy [13], [14], [64], [70], *i.e.*, keeping the local information from being exploited by adversarial reverse engineering. Chen *et al.* [8] proposed a novel testing framework for deep learning copyright protection, which can be adjusted to detect the knowledge extraction against DRL.

VIII. Disscusion

Highlights of ORL-Auditor. 1) ORL-Auditor is the first approach to conduct trajectory-level dataset auditing for offline DRL models. 2) By conducting a comprehensive analysis of ORL-Auditor under different experimental settings, such as the shadow model's amount, the significance level in hypothesis testing, the trajectory size, and the robustness against ensemble architecture and action distortion, we conclude some useful observations for adopting ORL-Auditor. 3) We apply ORL-Auditor to audit the models trained on the open-source datasets from Google and DeepMind. All TPR and TNR results are superior than 95%, demonstrating ORL-Auditor is an effective and efficient strategy for the published datasets.

Limitations and Future Work. Below, we discuss the limitations of ORL-Auditor and promising directions for further improvements. 1) From Section V-D, the accuracy of ORL-Auditor decreases when the significance level downs to 0.001. Thus, it is interesting to enhance ORL-Auditor to satisfy stricter auditing demands in the future. 2) ORL-Auditor based on a single distance metric may not be sufficiently robust to strong distortion. Based on the observations in Section VI-B, integrating more distance metrics in the audit process may be a further promising direction.

IX. Conclusion

In this work, we propose a novel trajectory-level dataset auditing method for offline DRL models relying on the insight that cumulative rewards can serve as the dataset's intrinsic fingerprint and exist in all models trained on the target dataset. Both the true positive rate and the true negative rate of ORL-Auditor exceed 90% on four offline DRL models and three task combinations. We show that ORL-Auditor is an effective and efficient solution to protect the IP of the dataset owners through multiple experiments. By studying parameter settings about the number of shadow models, the significance level in hypothesis testing, and the trajectory size, we conclude several important observations for adopting ORL-Auditor in practice. The robustness evaluation demonstrates that ORL-Auditor can resist the defenses of the model ensemble and the action distortion of the suspect model. Integrating multiple distance metrics to improve the robustness of ORL-Auditor against action distortion is a promising direction for future work. Finally, we utilize the open-source datasets from Google [17] and DeepMind [25] to examine the practicality of ORL-Auditor, and show that ORL-Auditor behaves excellently on existing published datasets.

Acknowledgment

We would like to thank the anonymous reviewers for their constructive comments. We also thank Yanchao Sun for sharing her expertise in reinforcement learning. This work was partly supported by the National Key Research and Development Program of China under No. 2022YFB3102100, NSFC under Grants 61833015, 62088101, 62103371, U20A20159, and the Fundamental Research Funds for the Central Universities 226-2022-00107, 226-2023-00111. Min Chen was partly sponsored by the Helmholtz Association within the project "Trustworthy Federated Data Analytics" (TFDA) (No. ZT-I-OO1 4). Zhikun Zhang was supported by the CISPA-Stanford Center for Cybersecurity (FKZ:13N1S0762).

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/, 2015.

[2] N. AlHinai. Introduction to Biomedical Signal Processing and Artificial Intelligence. In *Biomedical Signal Processing and Artificial Intelligence in Healthcare*, Developments in Biomedical Engineering and Bioelectronics, pages 1–28. Elsevier, 2020.

[3] S. Amarjyoti. Deep Reinforcement Learning for Robotic Manipulation - The State of the Art. *CoRR abs/1701.08878*, 2017.

[4] C. Beattie, J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik, J. Schrittwieser, K. Anderson, S. York, M. Cant, A. Cain, A. Bolton, S. Gaffney, H. King, D. Hassabis, S. Legg, and S. Petersen. DeepMind Lab. *CoRR*, abs/1612.03801, 2016.

[5] Biscom. Employee Departure Creates Gaping Security Hole. https://www.biscom.com/employee-departure-creates-gaping-security-hole-says-new-data, 2021.

[6] F. Boenisch. A Systematic Review on Model Watermarking for Neural Networks. *Frontiers Big Data*, 4:729663, 2021.

[7] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *CoRR*, abs/1606.01540, 2016.

[8] J. Chen, J. Wang, T. Peng, Y. Sun, P. Cheng, S. Ji, X. Ma, B. Li, and D. Song. Copy, Right? a Testing Framework for Copyright Protection of Deep Learning Models. In *IEEE S&P*, pages 824–841, 2022.

[9] K. Chen, S. Guo, T. Zhang, X. Xie, and Y. Liu. Stealing Deep Reinforcement Learning Models for Fun and Profit. In *ACM Asia Conference on Computer and Communications Security (ASIACCS)*, pages 307–319, 2021.

[10] M. Chen, Z. Zhang, T. Wang, M. Backes, M. Humbert, and Y. Zhang. When Machine Unlearning Jeopardize Privacy. In *ACM CCS*, 2021.

[11] M. Chen, Z. Zhang, T. Wang, M. Backes, and Y. Zhang. FACE-AUDITOR: Data Auditing in Facial Recognition Systems. In *USENIX Security*, 2023.

[12] L. Du, M. Chen, M. Sun, S. Ji, P. Cheng, J. Chen, and Z. Zhang. ORL-Auditor: Dataset Auditing in Offline Deep Reinforcement Learning. *CoRR abs/2309.03081*, 2023.

[13] L. Du, Z. Zhang, S. Bai, C. Liu, S. Ji, P. Cheng, and J. Chen. AHEAD: Adaptive Hierarchical Decomposition for Range Query under Local Differential Privacy. In *ACM CCS*, 2021.

[14] Y. Du, Y. Hu, Z. Zhang, Z. Fang, L. Chen, B. Zheng, and Y. Gao. LDPTrace: Locally Differentially Private Trajectory Synthesis. In *VLDB*, 2023.

[15] A. Dziedzic, H. Duan, M. A. Kaleem, N. Dhawan, J. Guan, Y. Cattan, F. Boenisch, and N. Papernot. Dataset Inference for Self-Supervised Models. *CoRR*, abs/2209.09024, 2022.

[16] A. R. Fayjie, S. Hossain, D. Oualid, and D. Lee. Driverless Car: Autonomous Driving Using Deep Reinforcement Learning in Urban Environment. In *International Conference on Ubiquitous Robots (UR)*, pages 896–901, 2018.

[17] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. *CoRR*, abs/2004.07219, 2020.

[18] S. Fujimoto, E. Conti, M. Ghavamzadeh, and J. Pineau. Benchmarking Batch Deep Reinforcement Learning Algorithms. *CoRR*, abs/1910.01708, 2019.

[19] S. Fujimoto and S. S. Gu. A Minimalist Approach to Offline Reinforcement Learning. In *NeurIPS*, pages 20132–20145, 2021.

[20] S. Fujimoto, D. Meger, and D. Precup. Off-Policy Deep Reinforcement Learning without Exploration. In *ICML*, pages 2052–2062, 2019.

[21] S. Fujimoto, H. van Hoof, and D. Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *ICML*, pages 1582–1591, 2018.

[22] M. Gomrokchi, S. Amin, H. Aboutalebi, A. Wong, and D. Precup. Where Did You Learn That From? Surprising Effectiveness of Membership Inference Attacks Against Temporally Correlated Data in Deep Reinforcement Learning. *CoRR*, abs/2109.03975, 2021.

[23] M. Gomrokchi, S. Main, S. Amin, and D. Precup. PrivAttack: A Membership Inference Attack Framework Against Deep Reinforcement Learning Agents. In *NeurIPS Workshop*, 2020.

[24] F. E. Grubbs. Sample Criteria for Testing Outlying Observations. *The Annals of Mathematical Statistics*, pages 27–58, 1950.

[25] Ç. Gülçehre, Z. Wang, A. Novikov, T. Paine, S. G. Colmenarejo, K. Zolna, R. Agarwal, J. Merel, D. J. Mankowitz, C. Paduraru, G. Dulac-Arnold, J. Li, M. Norouzi, M. Hoffman, N. Heess, and N. de Freitas. RL Unplugged: A Collection of Benchmarks for Offline Reinforcement Learning. In *NeurIPS 2020*, 2020.

[26] N. Gürtler, S. Blaes, P. Kolev, F. Widmaier, M. Wuthrich, S. Bauer, B. Schölkopf, and G. Martius. Benchmarking Offline Reinforcement Learning on Real-Robot Hardware. In *ICLR*, 2023.

[27] S. He, K. Shi, C. Liu, B. Guo, J. Chen, and Z. Shi. Collaborative Sensing in Internet of Things: A Comprehensive Survey. *IEEE Communications Surveys & Tutorials*, 2022.

[28] H. Huang, Z. Zhang, Y. Shen, M. Backes, Q. Li, and Y. Zhang. On the Privacy Risks of Cell-Based NAS Architectures. In *ACM CCS*, 2022.

[29] I. Ilahi, M. Usama, J. Qadir, M. U. Janjua, A. I. Al-Fuqaha, D. T. Hoang, and D. Niyato. Challenges and Countermeasures for Adversarial Attacks on Deep Reinforcement Learning. *CoRR abs/2001.09684*, 2020.

[30] I. Jarin and B. Eshete. MIAShield: Defending Membership Inference Attacks via Preemptive Exclusion of Members. In *Privacy Enhancing Technologies Symposium*, pages 400–416, 2023.

[31] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims. MOReL: Model-Based Offline Reinforcement Learning. In *NeurIPS*, 2020.

[32] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *ICLR*, 2014.

[33] P. Kiourti, K. Wardega, J. Susmit, and W. Li. TrojDRL: Evaluation of Backdoor Attacks on Deep Reinforcement Learning. In *DAC*, 2020.

[34] I. Kostrikov, A. Nair, and S. Levine. Offline Reinforcement Learning with Implicit Q-Learning. In *ICLR*, 2022.

[35] S. Lange, T. Gabel, and M. A. Riedmiller. Batch Reinforcement Learning. In *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pages 45–73. Springer, 2012.

[36] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *CoRR*, abs/2005.01643, 2020.

[37] Y. Li, Y. Bai, Y. Jiang, Y. Yang, S. Xia, and B. Li. Untargeted Backdoor Watermark: Towards Harmless and Stealthy Dataset Copyright Protection. *CoRR*, abs/2210.00875, 2022.

[38] Y. Li, Z. Zhang, J. Bai, B. Wu, Y. Jiang, and S. Xia. Open-sourced Dataset Protection via Backdoor Watermarking. *CoRR*, abs/2010.05821, 2020.

[39] Z. Lin, J. Gehring, V. Khalidov, and G. Synnaeve. STARDATA: A StarCraft AI Research Dataset. *CoRR*, abs/1708.02139, 2017.

[40] Y. Liu, R. Wen, X. He, A. Salem, Z. Zhang, M. Backes, E. D. Cristofaro, M. Fritz, and Y. Zhang. ML-Doctor: Holistic Risk Assessment of Inference Attacks Against Machine Learning Models. In *USENIX Security*, 2022.

[41] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas. Application of Deep Reinforcement Learning to Intrusion Detection for Supervised Problems. *Expert Systems with Applications*, 141:112963, 2020.

[42] P. Maini, M. Yaghini, and N. Papernot. Dataset inference: Ownership resolution in machine learning. In *ICLR*, 2021.

[43] D. Mwiti. 10 Real-Life Applications of Reinforcement Learning. https://neptune.ai/blog/reinforcement-learning-applications, 2021.

[44] H. Ono and T. Takahashi. Locally Private Distributed Reinforcement Learning. *CoRR abs/2001.11718*, 2020.

[45] T. L. Paine, C. Paduraru, A. Michi, Ç. Gülçehre, K. Zolna, A. Novikov, Z. Wang, and N. de Freitas. Hyperparameter Selection for Offline Reinforcement Learning. *CoRR*, abs/2007.09055, 2020.

[46] X. Pan, W. Wang, X. Zhang, B. Li, J. Yi, and D. Song. How You Act Tells a Lot: Privacy-Leaking Attack on Deep Reinforcement Learning. In *AAMAS*, pages 368–376, 2019.

[47] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary. Robust Deep Reinforcement Learning with Adversarial Attacks. In *AAMAS*, pages 2040–2042, 2018.

[48] D. Pomerleau. ALVINN: An Autonomous Land Vehicle in a Neural Network. In *NeurIPS*, pages 305–313, 1988.

[49] R. F. Prudencio, M. R. O. A. Máximo, and E. L. Colombini. A Survey on Offline Reinforcement Learning: Taxonomy, Review, and Open Problems. *CoRR*, abs/2203.01387, 2022.

[50] H. Pu, L. He, P. Cheng, M. Sun, and J. Chen. Security of Industrial Robots: Vulnerabilities, Attacks, and Mitigations. *IEEE Network*, 2022.

[51] R. Qin, S. Gao, X. Zhang, Z. Xu, S. Huang, Z. Li, W. Zhang, and Y. Yu. NeoRL: A Near Real-World Benchmark for Offline Reinforcement Learning. *CoRR*, abs/2102.00714, 2021.

[52] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[53] Y. Rubner, C. Tomasi, and L. J. Guibas. A Metric for Distributions with Applications to Image Databases. In *ICCV*, pages 59–66, 1998.

[54] T. Rupprecht and Y. Wang. A Survey for Deep Reinforcement Learning in Markovian Cyber-physical Systems: Common Problems and Solutions. *Neural Networks*, 153:13–36, 2022.

[55] T. Seno and M. Imai. d3rlpy: An Offline Deep Reinforcement Learning Library. *Journal of Machine Learning Research*, 23(315):1–20, 2022.

[56] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *IEEE S&P*, pages 3–18, 2017.

[57] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller. Deterministic Policy Gradient Algorithms. In *ICML*, pages 387–395, 2014.

[58] M. A. Stephens. EDF Statistics for Goodness of Fit and Some Comparisons. *Journal of the American statistical Association*, 69(347):730–737, 1974.

[59] X. Tang, S. Mahloujifar, L. Song, V. Shejwalkar, M. Nasr, A. Houmansadr, and P. Mittal. Mitigating Membership Inference Attacks by Self-Distillation Through a Novel Ensemble Architecture. In *USENIX Security*, pages 1433–1450, 2022.

[60] Tessian. How the Great Resignation is Creating More Security Challenges. https://www.tessian.com/blog/how-the-great-resignation-is-creating-more-security-challenges/, 2021.

[61] L. Van der Maaten and G. Hinton. Visualizing Data Using t-SNE. *Journal of machine learning research*, 9(11), 2008.

[62] H. Wang, Z. Zhang, T. Wang, S. He, M. Backes, J. Chen, and Y. Zhang. PrivTrace: Differentially Private Trajectory Synthesis by Adaptive Markov Model. In *USENIX Security*, 2023.

[63] L. Wang, Z. Javed, X. Wu, W. Guo, X. Xing, and D. Song. BACKDOORL: Backdoor Attack against Competitive Reinforcement Learning. In *IJCAI*, pages 3699–3705, 2021.

[64] T. Wang, J. Q. Chen, Z. Zhang, D. Su, Y. Cheng, Z. Li, N. Li, and S. Jha. Continuous Release of Data Streams under both Centralized and Local Differential Privacy. In *ACM CCS*, 2021.

[65] Y. Wang, E. Sarkar, W. Li, M. Maniatakos, and S. E. Jabari. Stop-and-Go: Exploring Backdoor Attacks on Deep Reinforcement Learning-Based Traffic Congestion Control Systems. *IEEE TIFS*, 16:4772–4787, 2021.

[66] Z. Yang, L. He, H. Yu, C. Zhao, P. Cheng, and J. Chen. Detecting PLC Intrusions using Control Invariants. *IEEE IoT J*, 9(12):9934–9947, 2022.

[67] T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn. COMBO: Conservative Offline Model-Based Policy Optimization. In *NeurIPS*, pages 28954–28967, 2021.

[68] Q. Yuan, Z. Zhang, L. Du, M. Chen, P. Cheng, and M. Sun. PrivGraph: Differentially Private Graph Data Publication by Exploiting Community Information. In *USENIX Security*, 2023.

[69] L. Zeng, M. Sun, X. Wan, Z. Zhang, R. Deng, and Y. Xu. Physics-constrained Vulnerability Assessment of Deep Reinforcement Learning-based SCOPF. *IEEE TPS*, 2022.

[70] Z. Zhang, T. Wang, N. Li, S. He, and J. Chen. CALM: Consistent Adaptive Local Marginal for Marginal Release under Local Differential Privacy. In *ACM CCS*, 2018.

[71] Z. Zhang, T. Wang, N. Li, J. Honorio, M. Backes, S. He, J. Chen, and Y. Zhang. PrivSyn: Differentially Private Data Synthesis. In *USENIX Security*, 2021.

Appendix

A. The Behavior Similarity of Models

In Figure 10, we provide the behavior similarity of the offline RL models trained on the datasets in Table II. Taking the Bipedal Walker task as an example, the dataset "0841" is regarded as the target dataset, and the other four are the public datasets. We observe that the behavior similarity of the RL models waves heavily among the different public training data. If the auditor adopts the dataset "1203" as the public training data, the auditor likely misclassifies the RL models trained on the other three public datasets into the bootleg models. In addition, the behavior similarity is also affected by different offline RL frameworks, *i.e.*, BC [48], BCQ [20], [18], IQL [34], and TD3PlusBC [19] (detailed in Section II-B).

B. The Details of Tasks

Lunar Lander (continuous version). The LunarLander task is to smoothly land a spaceship between two flags on the target pad. The landing pad is always at coordinates (0,0). The ship has three throttles; one throttle points downward (the main engine) and the other two points in the left and right direction (the left and right engines). The observation is an 8-dimensional vector: the coordinates of the lander in the x-axis and y-axis, its linear velocities in the x-axis and y-axis, its angle, its angular velocity, and two booleans that represent whether each leg is in contact with the ground or not. The action is two real values ranging in $[-1, 1]$. The first dimension controls the main engine, where the engine is off when the value is in $[-1, 0)$ and increases from 50% to 100% throttle when the value rises from 0 to 1. The other two points are controlled by the second value, where the spaceship fires the left engine if the value in $[-1.0, -0.5)$, fires the right engine if the value in $[0.5, 1)$, and shuts down both engines if the value in $[-0.5, 0.5]$. The reward for moving from the top of the screen to the landing pad and zero speed is about 140 points. Landing outside the landing pad is possible. Thus, the player loses the terminal reward if the lander moves away from the landing pad. The player gets 10 additional points for each leg touching the ground. Firing the main engine is -0.3 points in each frame. The episode finishes if the lander crashes or lands smoothly, receiving -100 or 100 points.

Bipedal Walker. The Bipedal Walker task is to operate a 4-joint walker robot to move forward as fast as possible. The robot is made of a hull and two legs. Each leg has 2 joints at both the hip and knee. The observation of the task includes eight continuous physical variables, *i.e.*, hull angle speed, angular velocity, horizontal speed, vertical speed, the position of joints and joints angular speed, legs contact with ground, and 10 lidar rangefinder measurements. Actions are motor speed values in the [-1, 1] range for each of the 4 joints at both hips and knees. The walker starts standing at the left end of the terrain with the hull horizontal, and both legs in the same position with a slight knee angle. The reward is given for moving forward, totaling 300+ points up to the far end. If the robot falls, it gets -100. Applying motor torque costs a small amount of points. A more optimal model will get a better score. The episode will terminate if the hull gets in contact with the ground or the walker exceeds the right end of the terrain length.

Ant. In this task, the player manipulates a 3D robot (ant), which consists of one torso (free rotational body) with four legs attached to it, with each leg having two links, to move in the forward (right) direction. The observation contains positional values of different body parts of the ant, followed by the velocities of those individual parts (their derivatives), with all the positions ordered before all the velocities. By default, an observation is a vector with shape (111,) where the elements correspond to the following: position (1-dim), angles (12-dim), velocities(14-dim), and the information about the contact forces (84-dim). The player can apply torques on the eight hinges connecting the two links of each leg and the torso (nine parts and eight hinges). Thus, the action space is an 8-dim continuous vector representing the torques applied at the hinge joints. The reward of the "Ant" task consists of four parts: healthy reward, forward reward, control cost, and contact cost. The task ends when either the ant state is unhealthy, or the episode duration reaches 1000 timesteps.

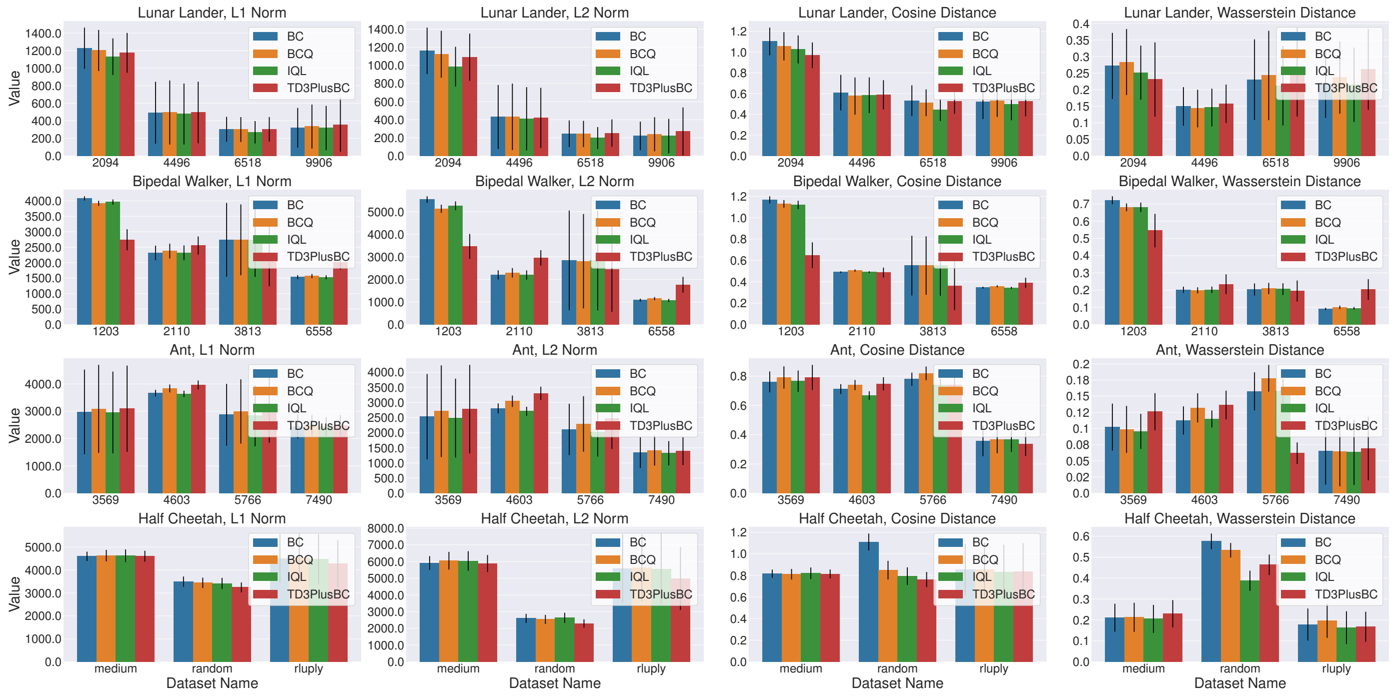


Fig. 10: Models' behavior similarity measured by ℓ_1 Norm, ℓ_2 Norm, Cosine Distance, and Wasserstein Distance. From Table II, we use the first dataset of each task as the private training data and the remaining four datasets are the public training data. For each plot, the x-axis displays the four public training data, and the y-axis shows the absolute fluctuating values of the behavior similarity between the models trained on the private dataset and the public datasets. BC, BCQ, IQL, and TD3PlusBC are abbreviations for different offline RL frameworks.

Artifact Appendix

A. *Description & Requirements*

1) How to access: This is a guideline for reproducing the experiment of NDSS paper titled "ORL-AUDITOR: Dataset Auditing in Offline Deep Reinforcement Learning" (https://dx.doi.org/10.14722/ndss.2024.23184). We have published the project on Zenodo licensed under the MIT License (https://doi.org/10.5281/zenodo.8303532). Since we are still working on the project to make it more user-friendly, we strongly recommend the readers to visit the latest version on Github (https://github.com/link-zju/ORL-Auditor).

2) Hardware dependencies: We conducted the experiment with the following hardware.

- OS: Ubuntu 20.04 64bit / Linux 5.4.0-42-generic
- CPU: AMD EPYC 7402 Processor - 1.65/2.80GHz
- Memory: 128GB
- GPU: RTX 3090 24GB

The actual hardware usage is small, so it can run on a desktop system with an NVIDIA graphics card.

3) Software dependencies: We used "Python" as the primary language and ran the scripts with version 3.8. The dependent libraries are detailed in requirement.txt.

4) Benchmarks: "None." Our scripts can generate all datasets and models.

B. *Artifact Installation & Configuration*

```
$ cd $PROJECT_SAVE_PATH
$ docker build -t orl-auditor:latest .
$ docker run -it --gpus all -v
    $PROJECT_SAVE_PATH:/workspace/off-rl -d
    orl-auditor:latest /bin/bash
$ source activate
```

C. *Major Claims*

The core metrics of our paper are the true positive rate (TPR) and true negative rate (TNR) in the audit process. For the experimental results about the performance of ORL-AUDITOR, it can be realized based on the bash scripts. Thus, we give a primary claim in the following.

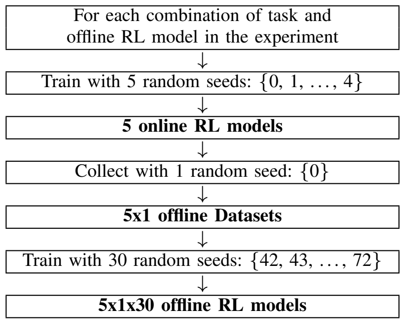
- (C1): Both TPR and TNR of ORL-AUDITOR are beyond 95% at least by one distance metric. This is proven by the experiment (E1) whose results are illustrated/reported in TABLE V.

D. *Evaluation*

1) Experiment (E1): For ease of understanding, we divide the workflow into two phases, i.e., preparation and execution. In preparation, we build the offline datasets and train the offline RL models. Then, in execution, we utilize the proposed method (ORL-Auditor) to audit the suspect models. The used hyperparameter settings are included in experimental_settings.yml.

Since running the whole evaluation of the paper is time-cost (about five days), we provide additional hyperparameter settings in experimental_settings.yml for quick testing.

TABLE I. THE MAIN STEPS IN DATASET GENERATION AND OFFLINE MODEL PREPARATION WITH THE DETAILS OF THE INPUT AND OUTPUT.


For each combination of task and offline RL model in the experiment
Train with 5 random seeds: {0, 1, ..., 4}
5 online RL models
Collect with 1 random seed: {0}
5x1 offline Datasets
Train with 30 random seeds: {42, 43, ..., 72}
5x1x30 offline RL models

Phase 1: Preparation

Table I illustrates the detailed process and random seed settings. The estimated computer time and GPU memory costs are calculated for single model (or dataset) generation. For example, training an online RL model costs 4 compute-minutes and 1 GB GPU memory when teacher_train_times $= 10^4$. However, in evaluation, we need more than one model (or dataset), so it takes a lot of time to generate them in sequence. In order to shorten the running time, we use parallel computing in the implementation.

Step 1: Train online RL models. [1 human-minute + compute-hour almost linearly increase with the args "teacher_train_times". For the "Lunar Lander" task, 10^4 "teacher_train_times" cost 4 compute-minutes, and GPU memory usage is around 1 GB.]

Step 2: Create the offline datasets. [1 human-minute + compute-hour linearly increases with the args "teacher_buffer_length". For the "Lunar Lander" task, collecting 10^4 steps costs 10 compute-seconds, and GPU memory usage is around 1 GB.]

Step 3: Train offline RL models. [1 human-minute + compute-hour depends on the args "student_agent_type". After starting the training, the command line will display the time remaining. GPU memory usage is around 1 GB.]

Step 4: Train the critic model. [1 human-minute + compute-hour depends on the args "teacher_buffer_save_path". When using a larger size teacher buffer, the training time will increase. . For a teacher buffer with 5×10^5 steps, training a critic model for 50 epochs costs nearly 2.5 compute-minutes. GPU memory usage is around 1.3 GB.]

Phase 2: Execution.

[1 human-minute + compute-hour mainly depends on the args "num_of_audited_episode". In addition, all student models are audited by default; thus, the number of student models will also significantly affect the compute-hour and required GPU resources.]

Finally, we utilize phase2_draw.sh to convert the raw results into a human-readable "xlsx" file.