TrajDeleter: Enabling Trajectory Forgetting in Offline Reinforcement Learning Agents

The Network and Distributed System Security Symposium 2025

Presenter: Chen Gong

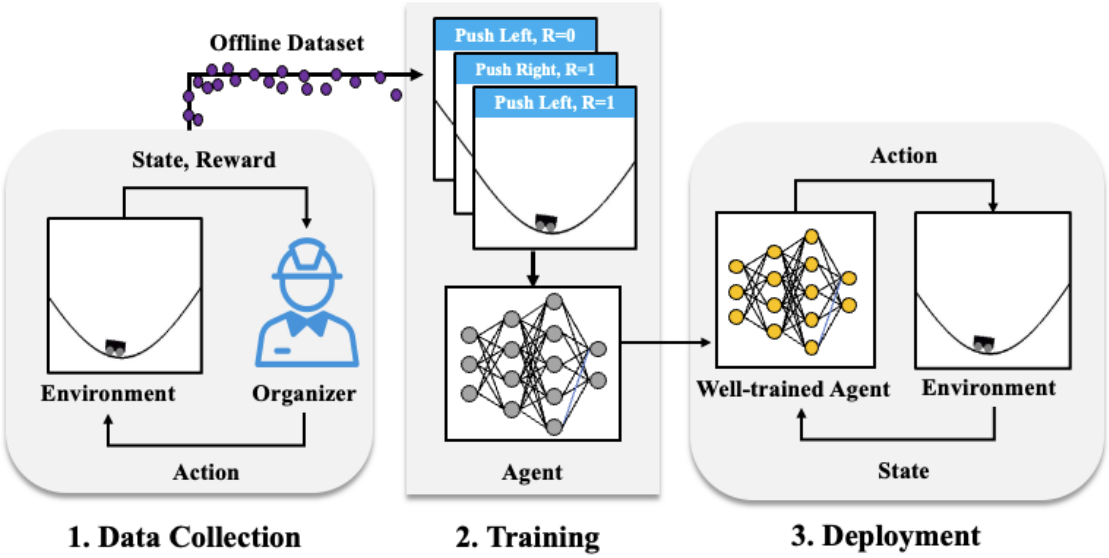
Chen Gong[1], Kecen Li[2], Jin Yao[1], Tianhao Wang[1]




[1]University of Virginia

[2]Chinese Academy of Sciences

Offline Reinforcement Learning

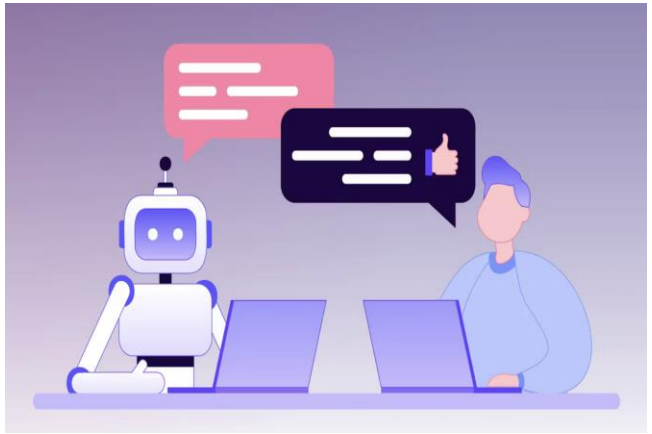


An Agent trained by interacting with a fixed dataset.

Trajectory:

$$(s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \cdots, s_{|\tau|}, a_{|\tau|}, r_{|\tau|})$$

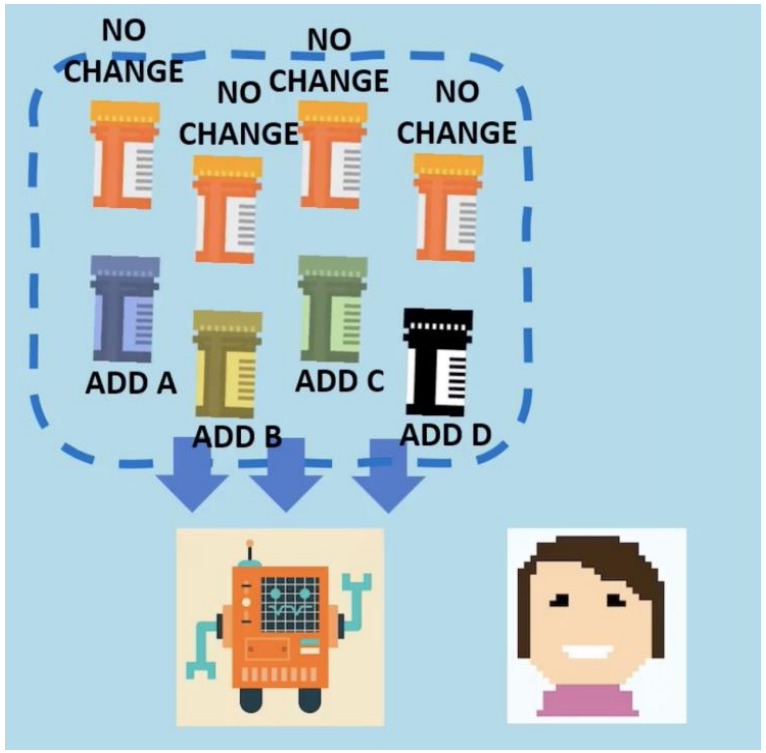
s_t: the current state;
a_t : current action executed by agents;
r_t : current reward feedback from environment;
s_{t+1}: the next time state;

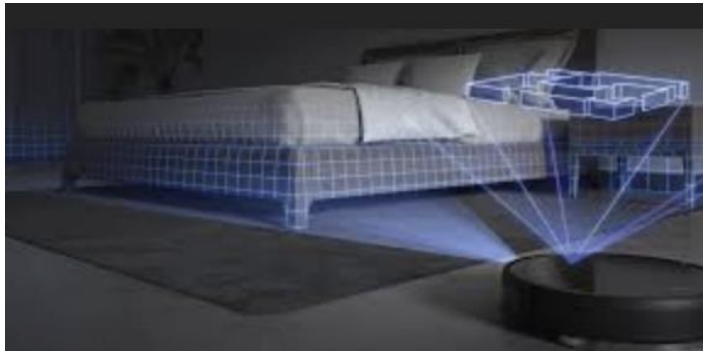


Levine, et al, 2020.

Unlearning in Trajectories

The trajectory may involve treatment recorders.

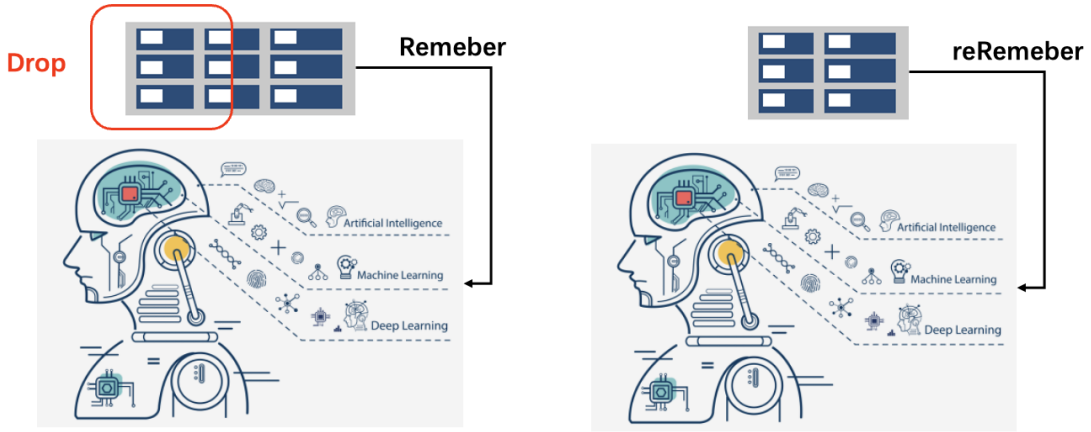




Some privacy information like position.

Users have the right to delete their data and erase its impact on the agent.

Naïve solution 1: Retraining



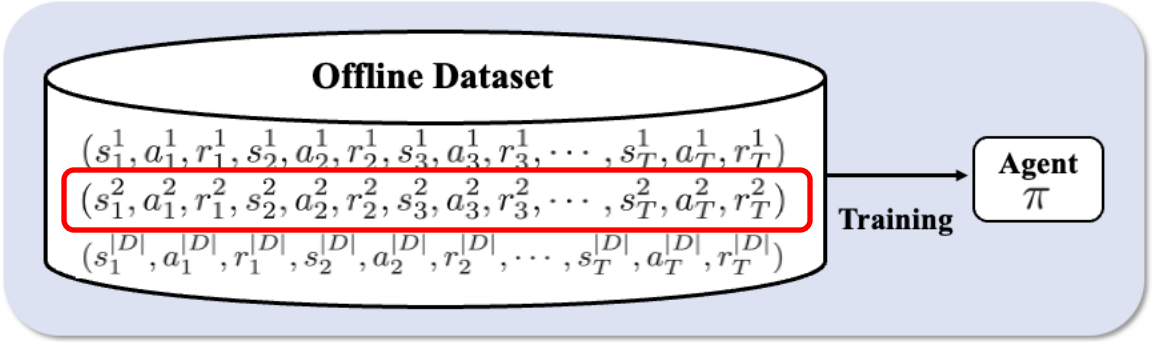
How to forget the data efficiently

Time consuming and expensive

The unlearning request can be made frequently.

How can we develop highly efficient unlearning methods to approximate retraining?

Naïve Solution 2: Fine-tuning



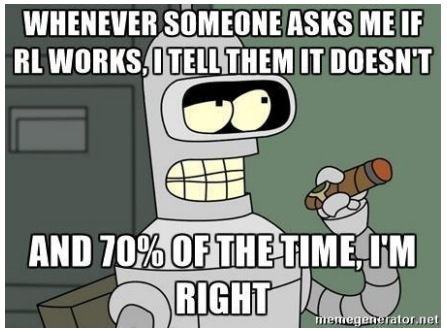
Fine-tuning the agents on the remain dataset.

Naïve Solution 3: Random Reward

$$(s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \cdots, s_{|\tau|}, a_{|\tau|}, r_{|\tau|})$$

Random Reward

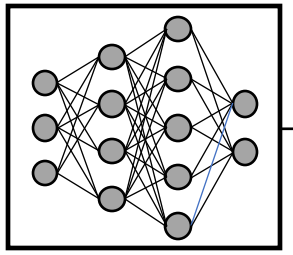
Stable and highly efficient forget!



TrajDeleter

Value function: $(s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \cdots, s_{|\tau|}, a_{|\tau|}, r_{|\tau|})$

$$Q(s_0, a_0) = \sum_{t=0}^{|\tau|} r_t$$



Agent

Learning

Objective function:

$$Q^{\pi}(s, a) = \mathbb{E}_{\tau \sim \pi}\left[R(\tau^{\pi}) | s_0 = s, a_0 = a\right]$$

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau \sim \pi}\left[R(\tau^{\pi})\right]$$

Balance Learning and Unlearning

Objective function:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{a\sim\pi}\left[Q^{\pi}(s,a)\right], \forall s \in \mathcal{S}$$

Balance Learning and Unlearning

Objective function:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{a\sim\pi}\left[Q^{\pi}(s,a)\right], \forall s \in \mathcal{S}$$

Unlearning Objective function:

$$\pi^* = \arg\min_{\pi} \mathbb{E}_{a\sim\pi}\left[Q^{\pi}(s,a)\right], \forall s \in \mathcal{S}$$

Balance Learning and Unlearning

Objective function:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{a\sim\pi}\left[Q^{\pi}(s,a)\right], \forall s \in \mathcal{S}$$

Unlearning Objective function:

$$\pi^* = \arg\min_{\pi} \mathbb{E}_{a\sim\pi}\left[Q^{\pi}(s,a)\right], \forall s \in \mathcal{S}$$

Remaining Trajectories D_m

Target Trajectories D_f

Objective function:

$$\min \mathbb{E}_{s\sim\mathcal{D}_f}\left[Q^{\pi'}(s,\pi'(s))\right] + \max \mathbb{E}_{s\sim\mathcal{D}_m}\left[Q^{\pi'}(s,\pi'(s))\right]$$

Balance Learning and Unlearning

Objective function:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{a\sim\pi}\left[Q^{\pi}(s,a)\right], \forall s \in \mathcal{S}$$

Unlearning Objective function:

$$\pi^* = \arg\min_{\pi} \mathbb{E}_{a\sim\pi}\left[Q^{\pi}(s,a)\right], \forall s \in \mathcal{S}$$

Forgetting Training

Remaining Trajectories

D_m

Target Trajectories

D_f

Objective function:

$$\min \mathbb{E}_{s\sim\mathcal{D}_f}\left[Q^{\pi'}(s,\pi'(s))\right] + \max \mathbb{E}_{s\sim\mathcal{D}_m}\left[Q^{\pi'}(s,\pi'(s))\right]$$

$$\min \mathbb{E}_{(s,a)\sim\mathcal{D}_m}\left[\left\|Q^{\pi'}(s,a) - Q^{\pi}(s,a)\right\|_{\infty}\right]$$

Convergence Training

π : originial agent

π': unlearned agent

Balance Learning and Unlearning

Objective function:

Forgetting Training

$$\min \mathbb{E}_{s\sim\mathcal{D}_f}\left[Q^{\pi'}(s,\pi'(s))\right] + \max \mathbb{E}_{s\sim\mathcal{D}_m}\left[Q^{\pi'}(s,\pi'(s))\right]$$

$$\min \mathbb{E}_{(s,a)\sim\mathcal{D}_m}\left[\left\|Q^{\pi'}(s,a) - Q^{\pi}(s,a)\right\|_\infty\right]$$

Convergence Training

Separate

Interaction Convergence: We assume that the offline dataset includes a diverse range of states. The state distribution generated by any policy is consistently bounded relative to the distribution in the offline dataset. Specifically, denoting the state distribution of the offline dataset as $\mu(s)$, for the state distribution $\nu(s)$ generated by any policy π_k, the condition $\forall s, \frac{\nu(s)}{\mu(s)} \leq C$ holds. Let Q^* indicate the optimal value function; we have,

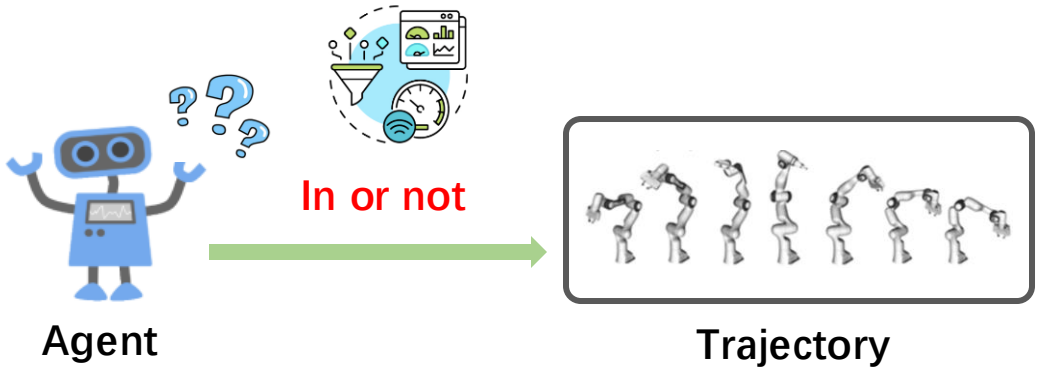
$$\|Q^* - Q^{\pi_{k+1}}\|_\infty \leq \gamma\|Q^* - Q^{\pi_k}\|_\infty + \epsilon + C\|Q^{\pi_k}\|_\infty,$$

where π_k denotes a sequence of policies correlated to their respective value functions Q^{π_k}. Here, ϵ signifies the approximation error in value estimation: $\|Q^{\pi_k} - (r + \gamma Q^{\pi_{k+1}})\|_\infty$.

S. Tosatto, et al, "Boosted fitted q-iteration," ICML 2017.

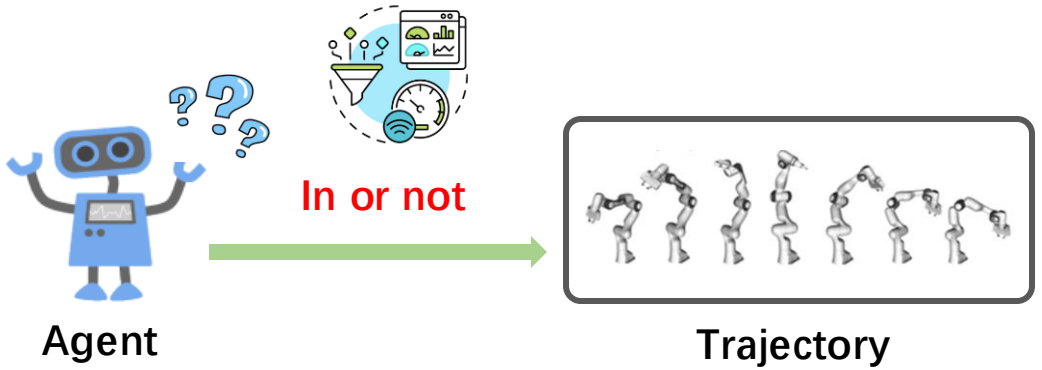
Auditor

How can we determine whether a trajectory is included in the agents' training set?

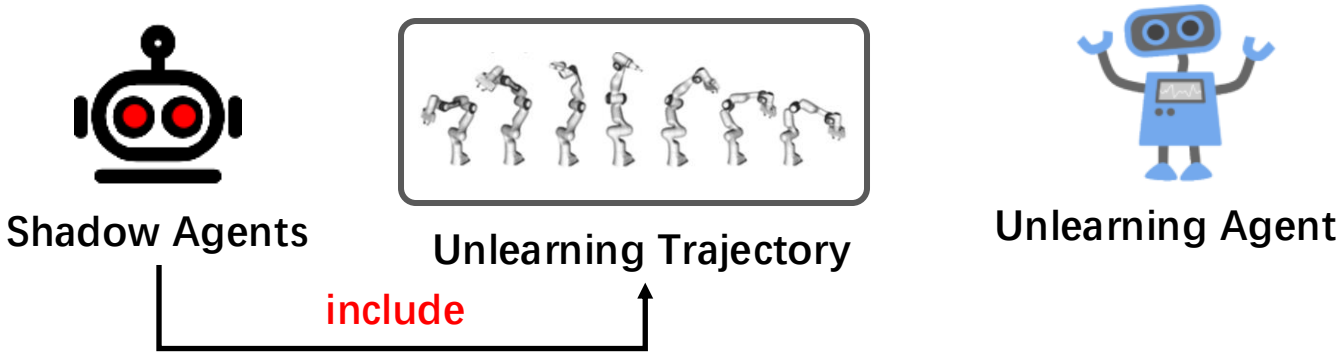


Auditor

How can we determine whether a trajectory is included in the agents' training set?



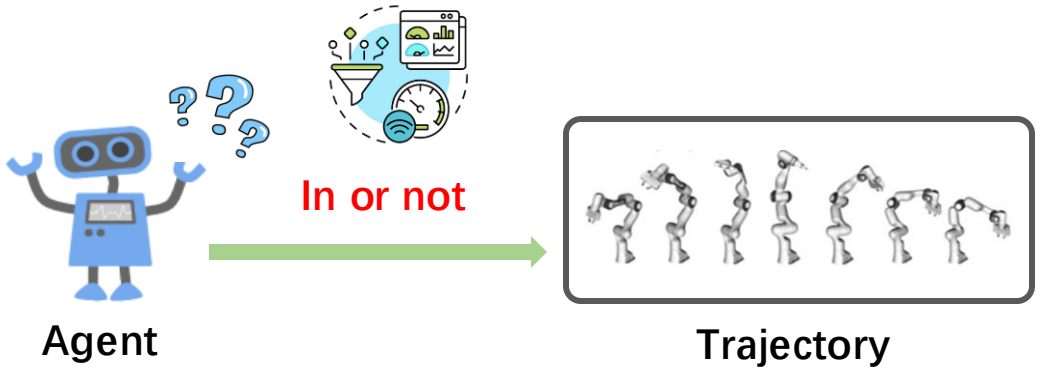
Membership Inference Attack as Auditor [1]



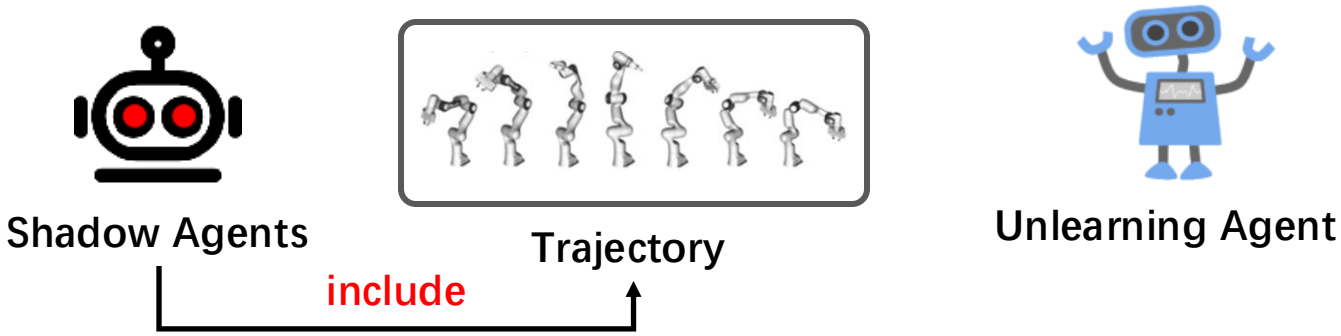
[1] Du et al. “ORL-AUDITOR: Dataset Auditing in Offline Deep Reinforcement Learning,” NDSS 2024.

Auditor

How can we determine whether a trajectory is included in the agents' training set?



Membership Inference Attack as Auditor [1]

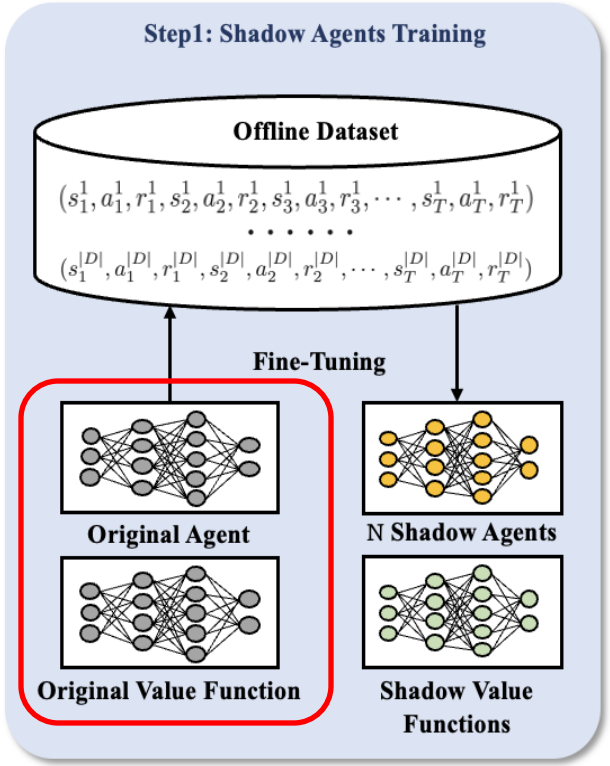


The instability is a drawback in offline RL training, but it can improve the auditor's efficiency.

[1] Du et al. “ORL-AUDITOR: Dataset Auditing in Offline Deep Reinforcement Learning,” NDSS 2024.

TrajAuditor

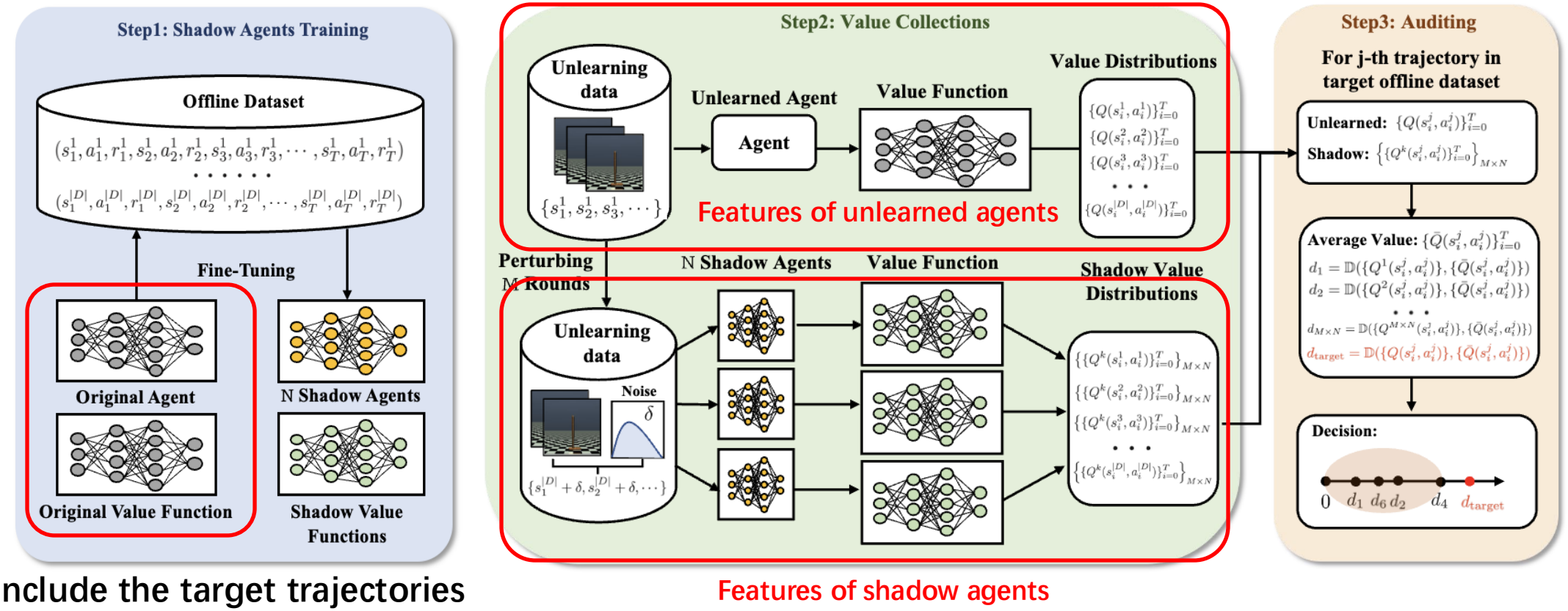
We can approximate the shadow model set, and avoid training them from scratch.



Include the target trajectories

TrajAuditor

Compare the similarity of unlearning data features extracted from unlearned agents and shadow agents.

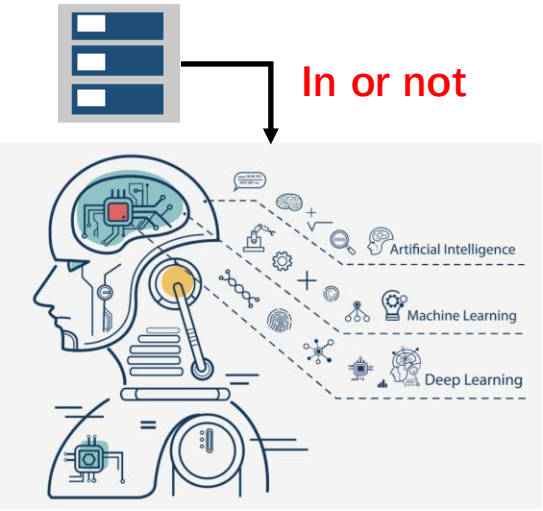


Include the target trajectories

Experiments Designs

RQ1. Dose the proposed TrajAuditor effectively verify the efficacy of unlearning?

(1) Determine the training dataset

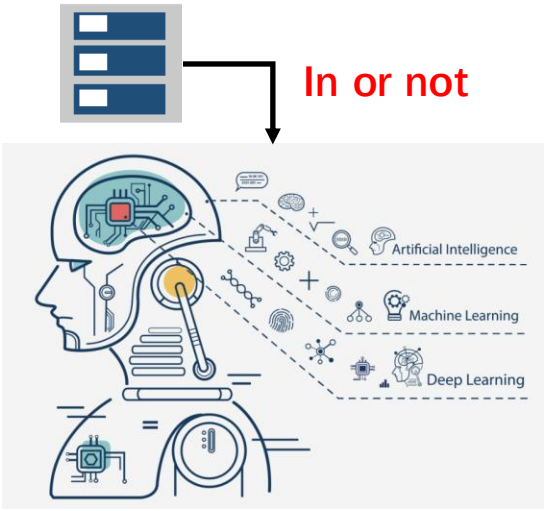


Experiments Designs

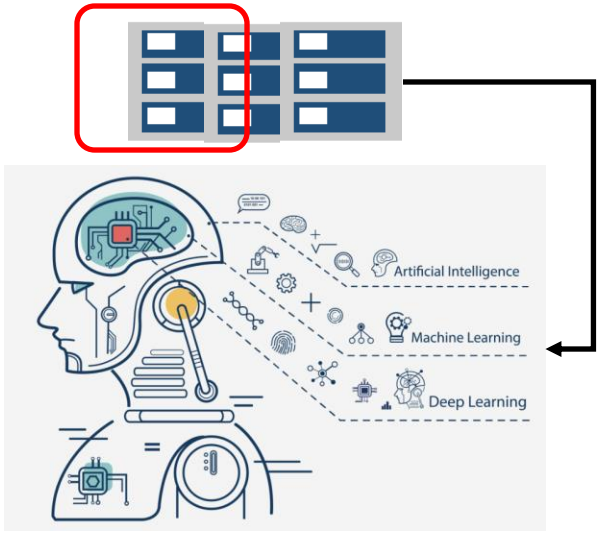
RQ1. Dose the proposed TrajAuditor effectively verify the efficacy of unlearning?

RQ2. How effective is TrajDeleter?

(1) Determine the training dataset



(2) Has it been successfully unlearning?



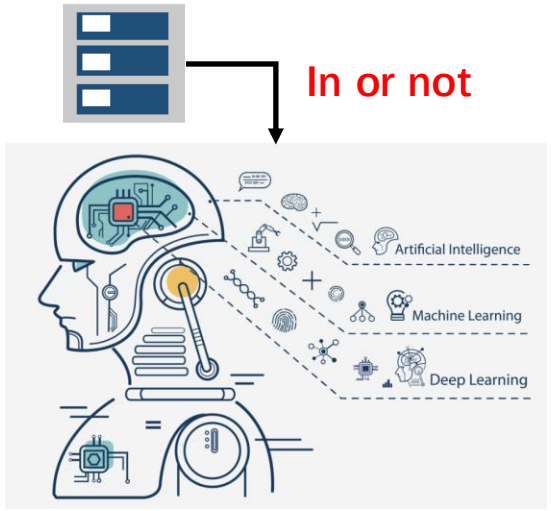
Experiments Designs

RQ1. Dose the proposed TrajAuditor effectively verify the efficacy of unlearning?

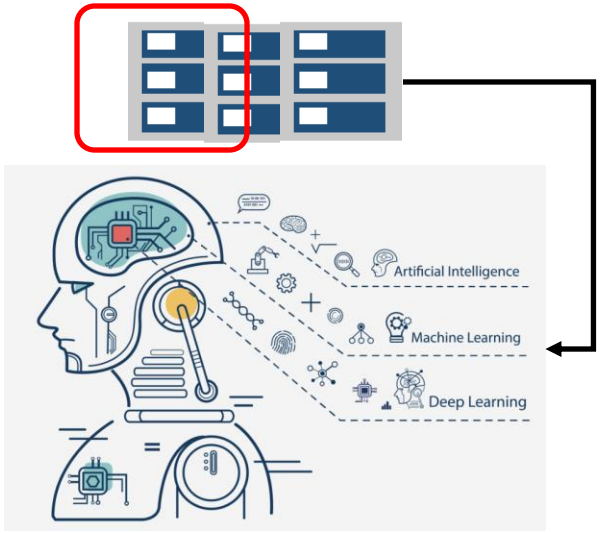
RQ2. How effective is TrajDeleter?

RQ3. How do hyper-parameters affect the TrajDeleter?

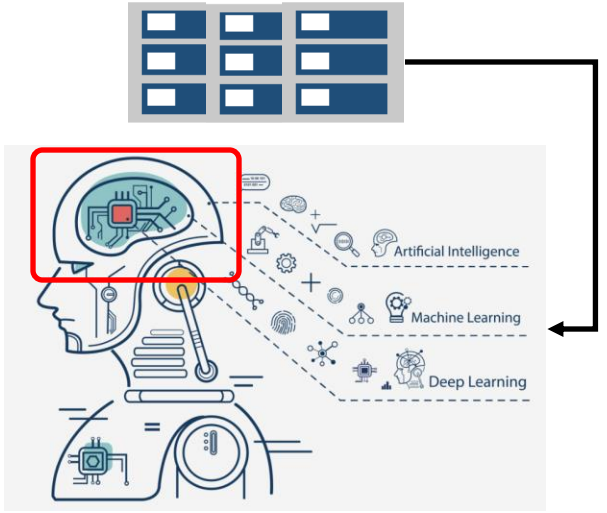
(1) Determine the training dataset



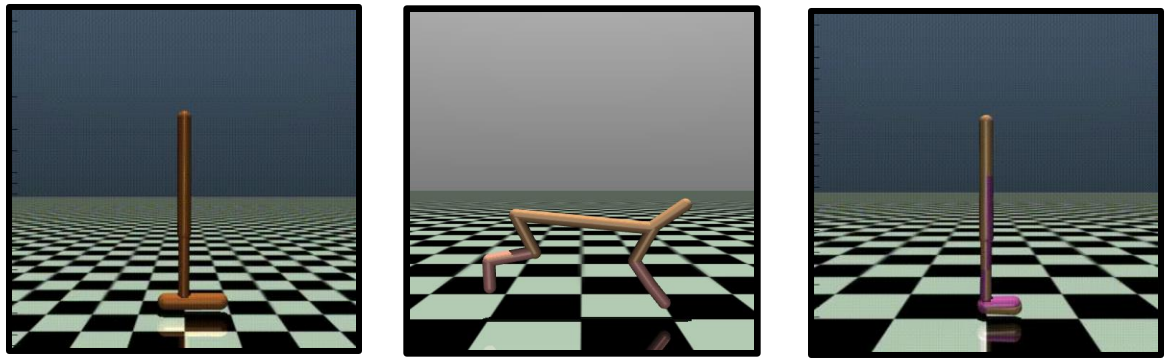
(2) Has it been successfully unlearning?



(3) The impact of hyper-parameters?



Investigated Offline RL datasets



(a) Hopper (b) Half Cheetah (c) Walker2D

Experiment Games

Six Investigated Offline RL algorithms: BCQ, BEAR, CQL, IQL, PLAS-P, and TD3+BC.

These algorithms demonstrate the best performance in D3RLPY [1] repository and most widely used in offline RL.

Evaluation Metrics: Precision and Recall

- Precision measures the fraction of predicted positive cases that are actually positive.
- Recall measures the fraction of actual positive cases that are correctly identified by the model.

Evaluation Metrics: Averaged Return

$$\frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}} R(\tau) \qquad R(\tau)=\sum_{i=0}^{|\tau|}\gamma_i r_i$$

$$\tau:(\langle s_0,a_0,r_0\rangle,\langle s_1,a_1,r_1\rangle,\cdots,\langle s_{|\tau|},a_{|\tau|},r_{|\tau|}\rangle)$$

[1] T. Seno et al, “D3rlpy: An offline deep reinforcement learning library,” JMLR, 2023.

Main Results

	Retraining (reference)		Fine-tuning		Random-reward		TrajDeleter	
Tasks	Remain	Unlearning	Remain	Unlearning	Remain	Unlearning	Remain	Unlearning
Hopper	78.8%	1.6%	79.7%	74.4%	80.3%	61.4%	79.1%	6.8%
Half-Cheetah	75.3%	0.0%	78.1%	52.8%	77.8%	36.6%	76.6%	0.3%
Walker2D	81.0%	1.5%	81.8%	62.4%	81.4%	49.2%	81.9%	10.3%

Average unlearning rate measured by TrajDeleter

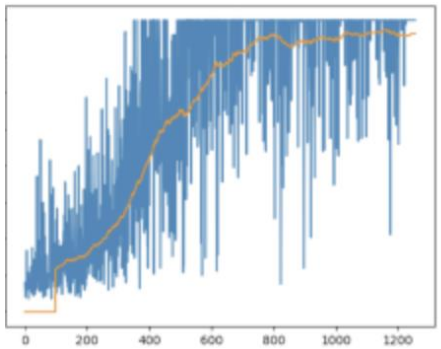
Key Results

	Retraining (reference)		Fine-tuning		Random-reward		TrajDeleter	
Tasks	Remain	Unlearning	Remain	Unlearning	Remain	Unlearning	Remain	Unlearning
Hopper	78.8%	1.6%	79.7%	74.4%	80.3%	61.4%	79.1%	6.8%
Half-Cheetah	75.3%	0.0%	78.1%	52.8%	77.8%	36.6%	76.6%	0.3%
Walker2D	81.0%	1.5%	81.8%	62.4%	81.4%	49.2%	81.9%	10.3%

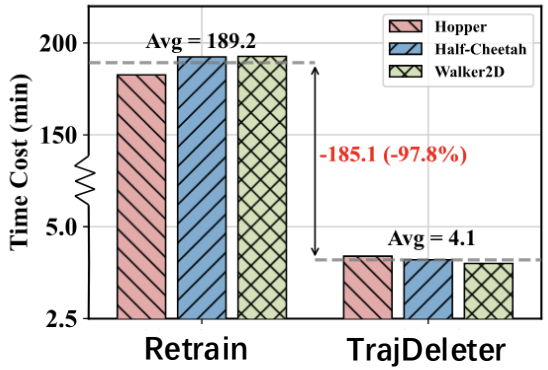
Average unlearning rate measured by TrajDeleter

TrajAuditor may have difficulty with algorithms when the training is unstable.

The worst case, in Hopper and Walker2D, the unlearning rate using PLAS-P: 47.5% and 60.7%.



Key Results



Only 2.2% time compared to retraining.

Environments	Algorithms	Remain data	Target data	Return
Hopper	BEAR	↓2.1%	- 0.0 %	↓1499
	BCQ	↓7.2%	- 0.0 %	↓2271
	CQL	↓1.3%	- 0.0 %	↓1903
	IQL	- 0.0%	- 0.0 %	↓345
	PLAS-P	↓45.5%	↑0.7 %	↓1783
	TD3PlusBC	↓1.4%	- 0.0 %	↑20
	Average	↓9.6%	↑ 0.1%	↓1297

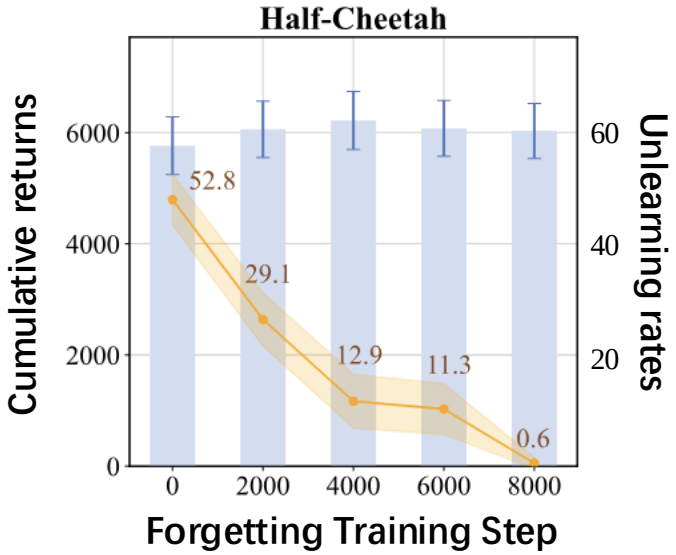
Without convergence training, the unlearning rate measured by TrajAuditor and agents' performance.

(1) The unlearning process may make agents forget the data not including in target dataset.

(2) The performance reduction of unlearned agent without convergence training.

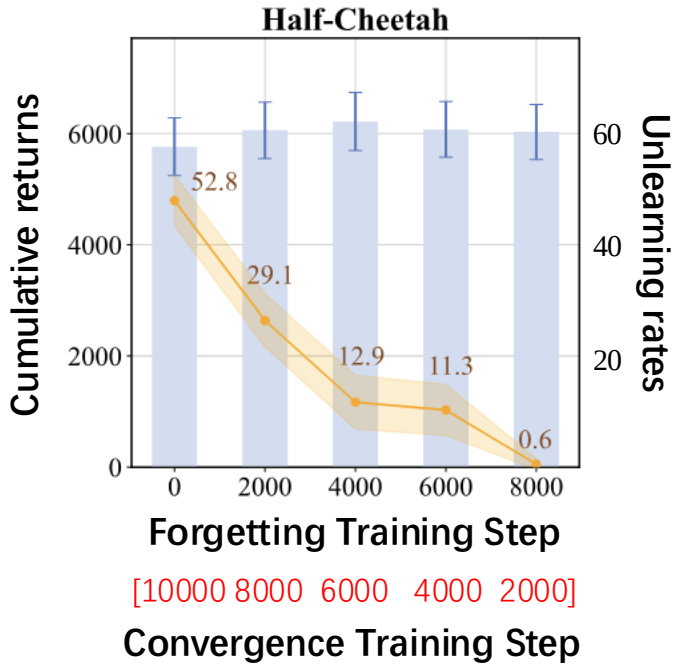
Key Results

Balancing of forgetting training and convergence training



Key Results

Balancing of forgetting training and convergence training



Forgetting Training Step + Convergence Training Step = 10000

10000 = 1% Training Step for retraining

Forgetting step significantly impacts the unlearning performance.
Convergence step dose not sensitive to the agent's performance.

Summary: A small value of convergence step is  , and the forgetting steps can be large, like 2000 and 8000 in our paper.

This paper proposed a trajectory forgetting method in offline reinforcement learning.

Hope it inspires!

Questions are welcome 😊!

chengong@virginia.edu

Paper

Artifact