SUPPRESSION IS NOT DELETION: EVALUATING OFFLINE-RL UNLEARNING AGAINST RETRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Deleting trajectories from an offline reinforcement learning dataset should leave the policy that training without them would have produced. Rather than retrain, several published offline-RL unlearning methods keep the deleted trajectories in the update and push the learner away from them, which we call suppression. This paper contributes a behavioral audit of unlearning against retraining, a per-state analysis of what suppression objectives optimize, and an evaluation of published methods on the benchmark released with TrajDeleter. Since retraining is random, a fresh seed giving a different policy, the audit compares a candidate with a family of independent retrains, and first audits the unchanged policy, the original with no unlearning applied, because if it passes, a candidate's pass proves nothing about deletion. At the benchmark's own deletion sizes the unchanged policy behaves like the retrained policies wherever the audit is validated on held-out retrains, so a pass there cannot show that a method deleted anything. On larger HalfCheetah deletions that do change behavior, no suppression method we evaluate reliably reaches the retrain family, whereas retrained policies stay inside it under further retained-data training. The analysis explains why: deletion gives the removed data zero weight, negative weighting of them overshoots the retraining target, and negating their rewards changes what the value function, the critic, learns; tests declared in advance find each effect in the predicted direction on HalfCheetah. The benchmark's own forgetting score follows the critic's values rather than the policy's behavior: shifting the critic by a constant reports complete forgetting while every action stays unchanged. Unlearning in offline RL should be judged against the behavior of retraining.

1 INTRODUCTION

Offline reinforcement learning (RL) learns a policy from a fixed dataset of previously collected experience (Levine et al., 2020). Because it needs no further interaction with the environment, it can learn control policies where exploration is costly or unsafe, from logs that many contributors supplied. Part of that dataset may later have to be removed, because a contributor withdraws consent or trajectories prove corrupted, and must then no longer influence the trained policy. Following the standard reference in machine unlearning (Guo et al., 2020; Neel et al., 2021), we take the target to be what the same learning procedure would have produced had those trajectories never been present: retraining on the retained data. Whether a policy has reached it is a question of behavior, since equal returns can hide different decisions.

Recent offline-RL unlearning methods take a different route. TrajDeleter (Gong et al., 2025), the gradient-ascent-then-refit (GA+Refit) baseline of the TOUR benchmark (Pan et al., 2026) and Safe-RULE (Jiang et al., 2026) keep the deleted trajectories during the update and push the learner away from them, by lowering the value or the likelihood it assigns to them. We call this design pattern *suppression*. Suppression asks the learner to move away from specified data; deletion asks what learning would have produced without them. The two need not coincide, as shown for negative-weight objectives in supervised unlearning (Mavrothalassitis et al., 2025; Zhang et al., 2024): if the retained data support the same behavior, retraining keeps it, while suppression pushes away from it. TrajDeleter's evaluation measures the suppression itself: its auditor reports, as a forgetting score, the percentage of deleted trajectories that a value-based test still recognizes as training data. Deletion

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

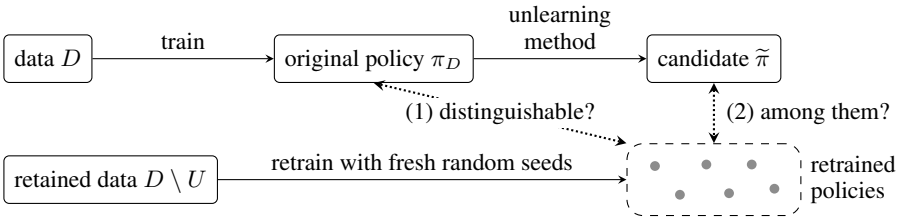

Figure 1: What deletion asks, and the check that comes first. Training on the data D gives π_D, which an unlearning method edits into $\widetilde{\pi}$; retraining on the retained data $D \setminus U$ with fresh seeds gives a family of policies (grey dots). Dotted arrows compare behavior. (1) If π_D cannot be told apart from the family, a pass by $\widetilde{\pi}$ says nothing about deletion. (2) Only then does asking whether $\widetilde{\pi}$ lies among the retrains test unlearning.

calls instead for a comparison with retraining, preceded by a check that the deletion changes the trained behavior (Figure 1).

In this paper we ask two linked questions: how agreement with retraining should be tested when retraining itself is random, since different seeds give different policies, and whether suppression reproduces retraining. We answer the first with a family of independent retrains, checked against further held-out retrains, and a visibility check that audits the original policy itself, unchanged, to ask whether the deletion is visible to the test at all. We answer the second with a stylized per-state model of what each objective asks of the actor and the critic.

We then apply the audit to the benchmark released with TrajDeleter, three D4RL locomotion datasets with two learners and two deletion sizes (Gong et al., 2025). At its deletion sizes the original policy already behaves like the retrained policies wherever the retrain family passes the held-out check: the deletion changes too little behavior for the audit to separate them, so those settings cannot show that any method removed anything. On larger HalfCheetah deletions that we declared, the unchanged policy is rejected, and there no suppression method we evaluate, or port from its paper's description, reaches the retrain family; the released procedure and the reward-hinge port, tested on ten independently trained original policies per learner, are rejected in 20 of 20 and 19 of 20. The obstacle is not the audit: retrained policies that never saw the deleted data stay inside the family under further retained training, whereas the suppression objectives move some out or destroy them, and from the original policy even a million retained updates reach the family only sometimes. Tests declared before measurement find each effect of the per-state model in the predicted direction on HalfCheetah, and shifting the critic by a constant drives the benchmark's own forgetting score to complete forgetting while leaving every action unchanged.

Our contributions are threefold. First, we add an unchanged-policy visibility gate to retraining-based evaluation and instantiate the audit with the discounted state–action occupancy. Le Cam's bound (Le Cam, 1986) shows that the deletion's behavioral visibility caps every rollout-based test's power to reject the unchanged policy, and we show the cap is tight. Second, in a stylized per-state model we show how suppression objectives differ from retraining at the actor and at the critic and how retained updates contract the difference, and we test these predictions on the neural agents. Third, on the TrajDeleter benchmark we show that its deletion sizes cannot credit any method, that the suppression methods we evaluate miss retraining on larger, detected deletions, and that its forgetting score can change with behavior unchanged. These results set the bar an offline-RL unlearning method must clear: evidence that it reproduces the behavior of retraining on deletions the audit can see, since a forgetting score that changes while behavior does not cannot stand in for that evidence. The audit kit we release lets future methods be held to this check without retraining the families.

2 Related work

Machine unlearning commonly uses training without the deleted data as its reference, with exact, certified, and approximate methods differing in the agreement they guarantee (Guo et al., 2020; Neel et al., 2021; Sekhari et al., 2021; Bourtoule et al., 2021). Other approaches exploit model structure, through influence-based approximations (Koh & Liang, 2017) or exact deletion from

additive sufficient statistics (Cao & Yang, 2015; Quan et al., 2027). Model-level evidence alone need not identify the training history (Thudi et al., 2022), and recent work shows that apparent forgetting can depend strongly on the evaluation used (Pawelczyk et al., 2025; Hu et al., 2025; Scholten et al., 2025).

Negative-weight objectives can drift away from retraining when retained and deleted data are dependent, as shown for regularized logistic regression (Mavrothalassitis et al., 2025). Gradient ascent can also collapse, motivating more stable negative-preference objectives (Zhang et al., 2024). Our analysis gives the per-state actor–critic analogue for offline RL, including the amplification introduced when deleted and retained minibatches are drawn in equal numbers.

TrajDeleter introduced trajectory unlearning for offline RL, with TrajAuditor as its auditor (Gong et al., 2025). TOUR broadened evaluation with retraining references, matched non-members and multiple privacy attacks, and found that conclusions depend strongly on the chosen audit (Pan et al., 2026). Safe-RULE extends unlearning to offline safe RL (Jiang et al., 2026), and concurrent imitation-learning work compares edited policies with multiple independent retrains under a conformal test (Li et al., 2026). Other sequential-decision settings ask different questions: removing an environment's influence or a target region from an agent (Ye et al., 2025; Takács & Gulyás, 2026), exact unlearning for online tabular RL (Nguyen-Tang & Arora, 2026) and approximate unlearning for offline bandits (Ye et al., 2026), and auditing whether a dataset was used at all (Du et al., 2024). Comparing a candidate with independent retrains, and a conformal test for doing so, are therefore not new here; the unchanged-policy gate, the visibility cap with its tightness, and the occupancy instantiation for offline RL are.

3 Behavioral fidelity and a retraining-calibrated audit

This section builds the audit that the rest of the paper uses. Comparing a candidate with retraining raises two problems that any such audit must solve: retraining is random, so the reference is a distribution of policies rather than one policy; and if training with and without the deleted data behaves alike, a candidate that resembles retraining has shown nothing about deletion, because the original policy resembles it too. We first fix what is compared (behavior, measured by occupancy), then answer the first problem with a family of independent retrains and a rejection rule with a distribution-free guarantee, then the second with a control that audits the original policy itself, and end with the order in which every result is read. Appendix A holds the proofs and bounds what the finite-rollout measurement resolves (Appendix A.3).

Setting. The environment is a Markov decision process with state space $\mathcal{S}$, action space $\mathcal{A}$, start distribution μ, reward $r(s,a)$ and discount γ; a policy $\pi(\cdot \mid s)$ chooses actions, and its return is the expected discounted reward $J^r(\pi) = \mathbb{E}^\pi_\mu \sum_{t\geq 0} \gamma^t r(s_t, a_t)$. An offline dataset D is a set of logged trajectories $(s_0, a_0, r_0, s_1, \ldots)$, and an offline RL learner such as IQL (Kostrikov et al., 2022) or TD3+BC (Fujimoto & Gu, 2021) trains from D alone an actor, the policy, and a critic $Q(s,a)$, its estimate of the return of taking a at s. A rollout is an episode generated by running a policy from a start state drawn from μ. Let $U \subseteq D$ be the trajectories to delete and Alg the complete, randomized training procedure. Retraining is then a random policy $\Pi_{-U} = \mathsf{Alg}(D \setminus U)$ with realizations π_{-U}; the original policy π_D is likewise a realization of $\Pi_D = \mathsf{Alg}(D)$, and an unlearning method turns it into a candidate $\widetilde{\pi}$. Subscripts R, U and D mark retained, deleted and all data, and hats mark estimates from rollouts.

Behavioral target. We compare policies by what they do rather than by their return, because two policies can earn the same return with different decisions. What they do is captured by the discounted occupancy, the distribution of the state and action visited at a random stopping time, the discount factor γ being read as the per-step probability of continuing. For stationary Markov policies (terminal states absorbing) this is the probability measure on $\mathcal{S} \times \mathcal{A}$ that gives a set E of state–action pairs the mass

$$d^\pi_\mu(E) = (1-\gamma) \sum_{t\geq 0} \gamma^t \Pr^\pi_\mu\big((s_t, a_t) \in E\big), \qquad \Delta(\pi, \pi') = \|d^\pi_\mu - d^{\pi'}_\mu\|_1 \in [0, 2],$$

where $\|\cdot\|_1$ is the ℓ_1 norm, so Δ is twice the total variation distance. Comparing occupancies controls every return at once, since $J^r(\pi) = \langle r, d^\pi_\mu \rangle / (1-\gamma)$, whereas equal values of finitely many returns

do not imply equal behavior (Proposition A.2). The target is therefore behavioral: $\Delta(\widetilde{\pi}, \pi_{-U})$ small for the policies π_{-U} that retraining produces.

Retraining-calibrated comparison. Since retraining produces no single policy, a candidate should look like one more retrain. We therefore compare it with a *retrain family* of N independent realizations $\pi_{-U}^{(1)}, \ldots, \pi_{-U}^{(N)}$, the reference retrains, and ask whether it lies farther from all of them than they lie from one another. $\widehat{\Delta}(\pi, \pi')$ denotes Δ estimated from rollouts on a fixed partition of the state–action space that depends on no candidate. Let, over $i, j \in \{1, \ldots, N\}$,

$$\widehat{d}_{\mathrm{cand}} = \min_i \widehat{\Delta}(\widetilde{\pi}, \pi_{-U}^{(i)}), \qquad \widehat{r}_{-U} = \max_i \min_{j \neq i} \widehat{\Delta}\big(\pi_{-U}^{(i)}, \pi_{-U}^{(j)}\big)$$

be the candidate's nearest-retrain distance and the *family radius*. The candidate is rejected when $\widehat{d}_{\mathrm{cand}} > \widehat{r}_{-U}$, that is, when it lies farther from every retrain than any retrain lies from its nearest other retrain. We report $\widehat{\delta} = \widehat{d}_{\mathrm{cand}} / \widehat{r}_{-U}$, so $\widehat{\delta} > 1$ means rejection.

The test comes with a guarantee that needs no model of retraining: a further independent retrain is rejected with probability at most $1/(N+1)$, the guarantee of a conformal rank test (Vovk et al., 2005; Angelopoulos & Bates, 2023). The guarantee assumes that the family represents retraining, which a finite family may not, so held-out retrains, used for neither the radius nor the partition, check it. A setting (one dataset, learner and deletion) is *calibrated*, and used, only when at least ninety percent of its held-out retrains pass; this is an empirical check with no error guarantee of its own. A pass means compatibility with observed retraining variation at the audit's resolution, not equal policies or erased information; whether a deletion is detectable at that resolution is likewise checked empirically.

Unchanged-policy control. A test can credit a method with deletion only if it would have failed doing nothing. We therefore audit the unchanged policy, the original π_D used as a candidate, first. If it is rejected in a calibrated setting, the audit detects the deletion, and a candidate that then enters the family has moved from the unchanged policy's region to one compatible with retraining. If it is not rejected, a later pass cannot be credited to unlearning, and when the deletion changes behavior little this limit applies to every behavioral audit. Call τ_U, the total variation distance between the laws of the trajectories that training with and without U generates (over training randomness), the *behavioral visibility* of the deletion. Any test that observes rollouts and has false-rejection probability at most q on retrains rejects the unchanged policy with probability at most $q + \tau_U$; the bound is tight (Proposition A.3). A non-rejection does not bound τ_U from above.

The audit therefore proceeds in a fixed order: calibration, then detection, then the candidates, whose acceptance counts as agreement only where the deletion is detected, whereas a rejection shows a departure in any calibrated setting.

4 How suppression objectives depart from retraining

Section 3 tests whether a candidate matches retraining; this section asks why suppression should fail to. In a stylized per-state model with the critic held fixed while the actor is fitted, we show two things. Reweighting the deleted data can move the actor's target only along one line, and only a zero weight reaches the retraining target. Negating the deleted rewards changes the critic's problem, and only the critic's within-state contrasts reach the actor. Together these predict how the released two-stage procedure behaves; the predictions are tested in Section 5.4. Proofs are in Appendix B.

4.1 Actor suppression

To see what a reweighting objective asks of the actor, we hold the critic $Q(s, \cdot)$ and its scale fixed, so that the actor's problem separates across states and each state becomes a weighted fit to its logged actions. The intuition is then simple: training on all data fits a weighted average of retained and deleted actions, deletion drops the deleted ones, and any other weight moves the fit along the line through them (Figure 2).

At a fixed critic, each learner's actor update is, state by state, a weighted fit to the logged actions of one of two forms, and we analyze those two idealized forms. The *quadratic* actor, TD3+BC's

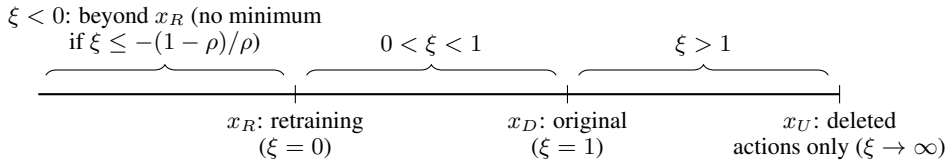

Figure 2: Reweighting deleted data moves the actor's target along one line at each state (schematic; Proposition 1). x_R, x_U and x_D are the targets of the retained, the deleted and all data; deleted weight $\xi = 0$ gives x_R, $\xi = 1$ gives x_D, $\xi > 1$ moves toward x_U, and $\xi < 0$ passes x_R, with no minimum once the deleted share is large enough.

actor at a fixed critic scale, fits an unconstrained action x in the continuous action space with per-outcome loss $\|x - a\|_2^2 - \lambda Q(s, x)$, $\lambda \ge 0$. The *likelihood* actor, the categorical idealization of IQL's advantage-weighted regression, fits a distribution π over a finite action set with loss $-\log \pi(a)$ and fixed positive advantage weights. At the state s, let ρ be the share of its logged observations that are deleted, and x_R, x_U the retained and deleted action means (for the likelihood actor, empirical action distributions; there ρ, x_R and x_U are weighted by the advantage weights); all three depend on s, which we drop while s is fixed. Training on all data targets $x_D = x_R + \rho(x_U - x_R)$, so x_R, x_D and x_U lie on one *deletion line*. Giving every retained observation weight one and every deleted observation relative weight ξ selects the point

$$x_\theta = x_R + \theta(x_U - x_R), \qquad \theta = \frac{\xi\rho}{1 - \rho + \xi\rho}.$$

Proposition 1 (Reweighting selects one effective target per state)**.** *For the quadratic actor at a fixed critic $Q(s, \cdot)$, continuous and bounded, or the likelihood actor with fixed positive weights, fix a state with $x_U \neq x_R$ and $0 < \rho < 1$, and weight the losses of the retained and deleted observations by 1 and ξ. If $1 - \rho + \xi\rho > 0$ and, for the likelihood actor, x_θ is a distribution, this objective has the same minimizing sequences as the learner's objective on data with mean (for the likelihood actor, distribution) x_θ; without a critic term they converge to x_θ. Deletion is $\xi = 0$, with target x_R; the original data are $\xi = 1$, with target x_D; a negative weight with $1 - \rho + \xi\rho > 0$ gives $\theta < 0$, a target beyond x_R on the side away from x_D. Otherwise, that is, if $\xi \le -(1-\rho)/\rho$ or, for the likelihood actor, x_θ has a negative entry, the objective is unbounded below.*

In words, giving the deleted actions a negative weight does not make the fit forget them: it makes the fit run away from them, past the point the retained data alone would choose. Only removal ($\xi = 0$) targets retraining at every state. Because the local deleted share ρ differs from state to state, one global negative weight overshoots by a different amount at each state, and where deleted actions dominate a state the objective has no minimum at all. Both the displacement and the time at which the idealized gradient flow crosses the retained-data target are state dependent, so no single forgetting strength or early-stopping budget implements deletion at every state, as the flow analysis of Appendix B.1 shows.

Sampling makes the weight larger than it looks. TrajDeleter's released procedure draws retained and deleted minibatches in equal numbers (balanced sampling), so a small deleted set is revisited far more often than its size warrants. If a fraction f of the transitions is deleted, one deleted transition carries the weight of $\omega = (1 - f)/f$ retained ones (Appendix B.2 gives unequal draws). Subtracting the deleted actor loss with balance constant α ($\alpha = 1$ in our runs) therefore acts as $\xi = -\alpha\omega$, and by Proposition 1 it has no finite optimum at any state with $\rho \ge 1/(1 + \alpha\omega)$.

The model idealizes joint neural training, which Section 5 measures; Appendix B.3 carries the displacement to occupancy distance, and Proposition B.3 places TOUR's ascent stage on the same line and shows that TrajDeleter's Eq. 4, its paper's actor objective, makes the actor greedy or anti-greedy in the critic.

4.2 Critic suppression

The released TrajDeleter procedure has two stages: a forgetting stage, in which retained updates alternate with updates on deleted transitions whose rewards are negated, and a convergence stage of

retained updates. Consider a state–action pair that occurs in both subsets, let $\rho(s, a)$ be the fraction of its logged transitions that are deleted, and suppose retained and deleted observations there sample the same reward and transition law. Under balanced updates the critic sees the same pair with its ordinary reward in one stream and its negated reward in the other, in a ratio set by the local share $\rho(s, a)$.

Proposition 2 (Balanced reward negation reverses shared rewards)**.** *Let f be the global deleted fraction of transitions. With equal retained and deleted update counts, uniform sampling within each subset, the same reward and transition law for retained and deleted observations of a pair, and a fixed bootstrap, the critic's expected squared-error objective is minimized pointwise by the reward plus bootstrap target of the retained data with r replaced by the effective reward*

$$r_{\text{eff}}(s, a) = r(s, a)\,\frac{f - \rho(s, a)}{f + \rho(s, a) - 2f\rho(s, a)},$$

which is zero at $\rho = f$ and reverses sign whenever $\rho(s, a) > f$, even when retained observations remain. For $f < 1/2$, in a two-action construction given in Appendix B.4, the greedy policies of the retained-data and the reward-negation fixed points are at occupancy distance 2.

In words, balanced sampling shows the critic a pair with its negated reward more often than with its true reward wherever the pair's deleted share exceeds the global fraction f, so the critic learns the opposite sign there although retained observations of the same pair remain; deletion would instead keep those retained observations and change nothing else.

Not every change of the critic reaches the actor, however. The actor sees the critic only through its within-state action contrasts (Proposition B.8): a statewise constant added to Q leaves it unchanged (at fixed λ), whereas a critic still trained on deleted data moves it wherever those data change the contrasts, even at zero actor weight. The first fact underlies the critic-shift diagnostic of Section 5.3.

The released procedure pairs this critic with an actor that is not suppressed at all. Its actor keeps its ordinary loss on deleted batches, so balanced sampling gives each deleted observation weight $+\omega$ and, by Proposition 1, a target beyond x_D toward the deleted actions (Proposition B.9). The forgetting stage thus asks the actor to imitate the deleted data more closely, and only the critic can push the other way. The convergence stage then runs retained updates, which contract every state's target toward x_R at a rate set by that state's retained data (Proposition B.10). Whatever the forgetting stage did is therefore damped geometrically, and later deletion requests damp the excursions of earlier ones.

5 Empirical study

This section applies the audit of Section 3 to the benchmark released with TrajDeleter (Gong et al., 2025), whose release reproduces the full pipeline, and asks four questions, one per subsection: is the deletion visible to the audit at all, since a pass proves nothing where the original policy would pass too; where it is, does suppression reproduce retraining; does the benchmark's own forgetting score track retraining fidelity; and do the mechanisms of Section 4 explain what the methods do?

The benchmark has twelve settings: two learners (IQL and TD3+BC, the two actors of Section 4), three D4RL medium-expert datasets (HalfCheetah, Hopper and Walker2d; Fu et al., 2020) and two deletion sizes (1% and 5% of the episodes), each with three original agents trained with independent seeds. The released procedure alternates retained updates with updates on deleted data whose rewards are negated for 8k updates, then runs 2k retained updates anchored to the original critic. We evaluate it alongside its stage controls, the two forms of its paper's objective (the literal Eq. 4, and the subtracted actor loss that its code carries commented out), ports of TOUR's GA+Refit and of Safe-RULE's value penalty (the reward-hinge port), and retained-only continuation; Appendix C.3 defines every candidate and Appendix D reports every result.

Because the benchmark's deletions leave the original policy indistinguishable from retraining, we also declared 30% deletions under the same deletion rule before training their retrains, with fifty reference and forty held-out retrains against eleven and ten per benchmark setting (HalfCheetah TD3+BC reached these sizes by a preregistered extension after its first ten held-out retrains failed the gate). Occupancies are estimated from 50 rollouts per policy on a 257-cell partition fitted on separate pilot retrains, and a setting is calibrated when nine of ten (thirty-six of forty) held-out retrains pass; Appendix C.2 gives the full protocol.

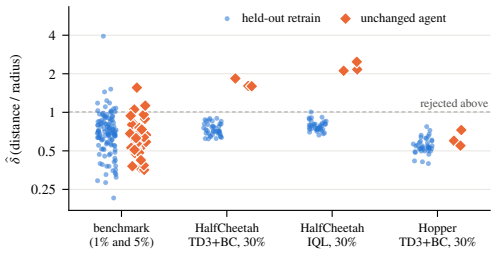

Figure 3: The visibility check, applied before comparing methods. Each point is one policy, placed by its distance to the nearest reference retrain in units of its setting's family radius ($\widehat{\delta}$, log scale); above the dashed line the audit rejects it. Orange diamonds: unchanged agents, three per setting. Blue dots: held-out retrains, which enter neither the radius nor the partition. The benchmark column pools twelve settings with ten held-out retrains each, 9/12 of which pass the held-out gate, so a few retrains sit above the line; each 30% deletion has forty held-out retrains.

Table 1: Candidates on the 30% deletions (HC: HalfCheetah; Hop: Hopper), each declared before it ran; definitions in Table C.2, all candidates in Table D.6. $\widehat{\delta}$: mean over three agents of the distance to the nearest reference retrain in family radii (above one is rejected; rejected agents in parentheses); return: mean D4RL normalized return (0: random, 100: expert), against 26–74, 34–43 and 112–113 for the retrained policies. Families have fifty references and forty held-out retrains; HalfCheetah is informative, Hopper is not. The suppression candidates use 5k to 10.5k updates.

Method	HC TD3+BC 30%		HC IQL 30%		Hop TD3+BC 30%	
	$\widehat{\delta}$	return	$\widehat{\delta}$	return	$\widehat{\delta}$	return
Unchanged policy	1.7 (3)	93	2.3 (3)	67	0.6 (0)	112
TrajDeleter (released)	1.6 (3)	98	2.6 (3)	74	0.6 (0)	112
Subtracted actor loss	3.3 (3)	37	8.3 (3)	−3	8.1 (3)	3
Literal Eq. 4	4.1 (3)	8	1.3 (3)	35	8.6 (3)	2
GA+Refit port	1.5 (3)	30	1.7 (3)	34	0.9 (0)	112
Reward-hinge port	1.2 (3)	96	1.4 (3)	62	0.8 (1)	112
Retained-only, 10k updates	1.3 (3)	85	1.9 (2)	59	0.7 (0)	112
Retained-only, 100k updates	0.8 (0)	73	1.1 (2)	50	0.9 (1)	112

5.1 Is the deletion behaviorally visible?

At the benchmark's deletion sizes the unchanged policy is indistinguishable from the retrained policies, so the deletion does not change behavior at the audit's resolution; at 30% it does on HalfCheetah and not on Hopper (Figure 3, which places the unchanged agents and held-out retrains by $\widehat{\delta}$). We call a deletion detected, and its setting informative, when at least two of its three unchanged agents are rejected, a rule declared in advance whose false-detection probability is bounded in Appendix A.1. No calibrated benchmark setting is detected (Tables D.1 and D.7), whereas the 30% HalfCheetah deletions separate all unchanged agents from retrains of calibrated families (40/40 and 39/40 held-out retrains pass). Hopper remains uninformative at that size (neither at 30% of its episodes, a tenth of its transitions, nor in its targeted 30% cells; Table D.7) and at a declared 45%, reported in Appendix D.2, where the unchanged agents pass and 40/40 held-out retrains are accepted. Visibility is thus a property of the deletion, not of its size: it depends on how much the deleted data change what the learner does, so it has to be measured for each request rather than assumed.

In the benchmark settings whose families pass the gate (9/12) the unchanged policy is rejected in only 1/27 agent evaluations, so a method's acceptance there cannot establish a deletion-specific effect. Yet the same settings reject the subtracted actor loss in 9/12 TD3+BC and 15/15 IQL evaluations, so the audit does detect departures there; on HalfCheetah at the 5% deletion one deleted transition carries the weight of 17 retained ones (Table C.1), so with $\alpha = 1$ that objective has no finite optimum wherever $\rho \geq f$, and the audit catches the damage even where the deletion itself is invisible. A pass at these sizes is therefore uninformative, while a rejection is not, and credit for deletion can be assessed only on the two HalfCheetah 30% settings.

5.2 When deletion is visible, does suppression reproduce retraining?

No evaluated suppression candidate reproduces retraining where the deletion is visible: in both informative HalfCheetah settings every one is rejected in every agent (Table 1). The released

procedure stays about as far from the family as the unchanged policy, with similar return: it leaves in place the behavior that the deleted data taught the original policy. The subtracted actor loss and the literal Eq. 4 variant fall far below the unchanged policy's return, as the unbounded or reversed objectives of Section 4 predict; the GA+Refit port's low return may also reflect its head reset and short refit, which our model does not cover. On seven further agents per learner, declared to test these verdicts, the released procedure is rejected in 7/7 (TD3+BC) and 7/7 (IQL), and the reward-hinge port, the suppression candidate nearest the family on TD3+BC, in 7/7 and 6/7, so the verdict does not rest on three agents.

Two questions decide what these rejections mean. First, is the retrained region reachable from the original policy at all? Only sometimes. Retained-only continuation, which trains the original policy on the retained data alone, is still rejected after 10k updates in 3/3 TD3+BC and 2/3 IQL agents, so at the methods' budget the table alone does not separate suppression from too short a run. After 100k updates all three TD3+BC agents are inside the family, but only 3/7 further agents are, and a million updates bring in no more (3/7), against a declared prediction of at least five. On IQL, 1/3 agents remain rejected after a million. Second, does the audit reject further training as such? No. Applied to ten held-out retrains per learner, policies that never saw the deleted data, retained-only continuation stays accepted in 9/10 and 10/10 at 100k updates (Table D.5). The suppression objectives should leave these policies alone, since they never saw the deleted data; not all do. The subtracted actor loss moves them out of the family in 10/10 and 10/10, and the reward-hinge port in 3/10 and 4/10. The released procedure stays inside in 7/10 and 10/10 but raises the TD3+BC return from 56 to 69, learning from data the policy never saw. What the audit rejects, then, is what the original policy learned from the deleted data: retained training does not reliably undo it, and deletion must.

5.3 Does the published forgetting score track retraining fidelity?

TrajAuditor's forgetting score does not track agreement with retraining. It reports the percentage of positive predictions (PPR), the fraction of deleted trajectories that its value-based test still classifies as present; lower PPR means stronger reported forgetting. Figure 4(a) shows that retrains need not read as forgotten and that a low PPR does not come with a small distance to the retrain family. Held-out retrains of the benchmark settings span 0.00–0.86 (Table D.2), so a policy that is exactly what deletion should produce can read as unforgotten. Agents outside the family can read as forgotten: on the informative HalfCheetah TD3+BC setting the released procedure reads 0.00 while all three of its agents are rejected, as do the subtracted actor loss and literal Eq. 4 at several radii. A low PPR is thus neither necessary nor sufficient for agreement with retraining.

Since PPR uses the candidate's own critic, as Appendix B.5 details, we take each unchanged agent, freeze its actor and subtract a constant from its critic, which cannot change a single action or rollout, as Appendix C.4 verifies. PPR nonetheless falls to zero (Figure 4(b)): at a shift of ten percent of the mean deleted-row value the 39/39 shifted agents of the benchmark settings and the HalfCheetah TD3+BC 30% setting read as forgotten (Table D.8), and raising the critic instead gives the same reading. A score that a constant critic shift can set to its extreme cannot by itself establish agreement with retraining or removal of information.

5.4 Mechanism checks: why do the methods behave this way?

Five predictions of Section 4 were tested with statistics and thresholds declared in advance; Table D.10 scores them and Appendix D.4 gives every statistic. Each is found in the predicted direction on HalfCheetah, with one missed IQL threshold and a weak or absent reward-negation effect on Hopper.

What happens inside the two-stage procedure. Along the saved HalfCheetah TD3+BC checkpoints at the 1% and 5% deletions (Figure D.1), every released run moves away from the nearest retrain during the forgetting stage, and the convergence stage reverses most of the excursion, leaving the agent near the unchanged policy. Both halves match the model (Table D.10). Because the convergence stage undoes most of what the forgetting stage did, four sequential requests that together delete the HalfCheetah TD3+BC 30% end as far from the family as one (Table D.14; three agents).

Changing the deleted-data weight. The signed sweep (Table D.15) varies only the actor's deleted-data weight ξ, rewards fixed, and gives the order in Table D.10. Negative weights damage the policy,

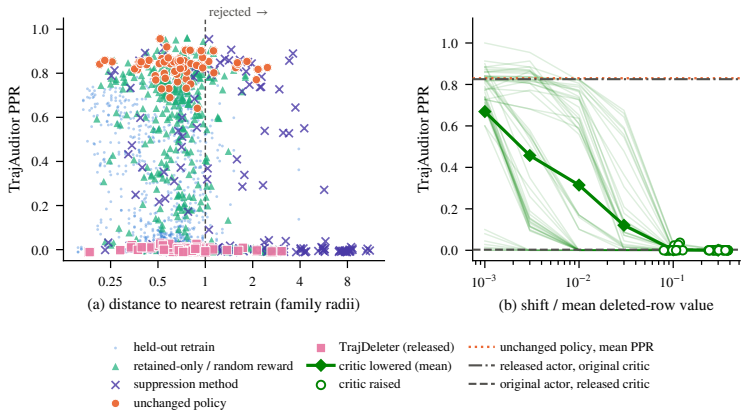

Figure 4: TrajAuditor's forgetting score does not track behavioral agreement with retraining (lower PPR reports more forgetting). (a) Each audited agent in 21 settings, placed by its distance to the nearest reference retrain in units of the family radius (log scale; rejected right of the dashed line) and its PPR: held-out retrains span the whole PPR range, and many rejected agents read as forgotten. "Suppression method": the subtracted actor loss, literal Eq. 4 and the two ports; "retained-only / random reward": the retained-only controls and the random-reward baseline. (b) PPR after subtracting from the critic a constant (a fraction of the mean deleted-row critic value, log scale) with the actor frozen, so that no action changes; thin lines are single agents, the thick line their mean, and hollow circles add the constant instead. Horizontal lines: the unchanged policies' mean PPR and the two actor–critic hybrids (original actor with the released critic, and vice versa).

on TD3+BC to 2.9 radii at $\xi = -1$ and on IQL to collapse (return -3), as the unbounded objective predicts (Proposition B.2). Two budget predictions fail, this one and the retained-only prediction of Section 5.2. At 100k updates $\xi = 0$ is accepted in one of three TD3+BC agents, against a declared two (Table D.16). A matched control whose critic avoids the deleted transitions was accepted in all three but rejected in 6/7 further agents, as was $\xi = 0$ (Table D.17); the critic's deleted-data updates do not explain the gap, and, as in Section 5.2, what must be removed is what the original policy already learned.

6 Discussion and Conclusion

A behavioral audit calibrated on independent retrains and gated by the unchanged policy yields three findings: at the deletion sizes of the TrajDeleter benchmark the original policy already behaves like retraining, so a pass there proves nothing; on the larger HalfCheetah deletions that do change behavior, no suppression method we evaluated or ported reliably reaches the retrain family, and the audit is not what stands in the way, because retrained policies stay inside the family under further retained training while the suppression objectives move some out or destroy them; and the benchmark's own forgetting score follows the level of the critic, which a constant shift drives to complete forgetting with no action changed. A per-state model explains why, and declared tests find each of its effects in the predicted direction on HalfCheetah. We conclude that unlearning in offline RL should be judged against the behavior of retraining, since pushing against the deleted data is no evidence of deletion. TrajDeleter's release made the audit possible.

The visible-deletion evidence comes from one environment, because the other declared deletions changed behavior too little or made retraining too variable to serve as a reference (Table D.7); finer measurement could extend the audit there. And behavioral fidelity, which we evaluate, is neither value-level change nor information erasure, so the audit should be paired with information-level tests. A method that implements deletion must remove what the original policy learned from the deleted data; the released audit kit scores candidates against the calibrated families without retraining.

REPRODUCIBILITY STATEMENT

All reported experimental results are generated by committed code from recorded runs. Appendix C specifies the experimental protocol, controls and deviations from the original TrajDeleter implementation, and Appendices C and E provide software, seeds and provenance. The code, the frozen protocols and an audit kit (every calibrated cell's partition, retrain-family vectors, radius and rule, with a standalone scorer) accompany the submission; the run records, which identify our computing environment, and the retrain checkpoints will be released after review.

AI USE STATEMENT

We designed and conducted the research, and we specified, reviewed and tested all AI-assisted work; we take full responsibility for the content of the paper, including its theory, claims, text and code. We used generative AI tools to write the code for the experiments and analyses; to give feedback on the interpretation of results; to assist with literature searches; and to improve the presentation of the text. We did not use them to develop theoretical models or conceptual frameworks, to generate synthetic data, for translation, or to clean or reformat datasets.

REFERENCES

Anastasios N. Angelopoulos and Stephen Bates. Conformal prediction: A gentle introduction. *Foundations and Trends in Machine Learning*, 16(4):494–591, 2023.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *IEEE Symposium on Security and Privacy (S&P)*, pp. 141–159, 2021.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *IEEE Symposium on Security and Privacy (S&P)*, pp. 463–480, 2015.

Linkang Du, Min Chen, Mingyang Sun, Shouling Ji, Peng Cheng, Jiming Chen, and Zhikun Zhang. ORL-AUDITOR: Dataset auditing in offline deep reinforcement learning. In *Network and Distributed System Security Symposium (NDSS)*, 2024. URL `https://arxiv.org/abs/2309.03081`.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 20132–20145, 2021.

Chen Gong, Kecen Li, Jin Yao, and Tianhao Wang. TrajDeleter: Enabling trajectory forgetting in offline reinforcement learning agents. In *Network and Distributed System Security Symposium (NDSS)*, 2025.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens van der Maaten. Certified data removal from machine learning models. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3832–3842, 2020.

Shengyuan Hu, Yiwei Fu, Zhiwei Steven Wu, and Virginia Smith. Unlearning or obfuscating? Jogging the memory of unlearned LLMs via benign relearning. In *International Conference on Learning Representations*, 2025.

Shixiong Jiang, Taozheng Zhu, and Fanxin Kong. Safe-RULE: Safe Reinforcement UnLEarning. *arXiv preprint arXiv:2606.09559*, 2026.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1885–1894, 2017.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022.

Lucien Le Cam. *Asymptotic Methods in Statistical Decision Theory*. Springer Series in Statistics. Springer, New York, 1986.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Jiazhuo Li, Yu Zhang, Yiming Fei, Kangkang Dong, Xiaojun Zhu, Houde Liu, and Jinze Tao. Rethinking demonstration unlearning in imitation learning for robotics. *arXiv preprint arXiv:2608.20784*, 2026.

Ioannis Mavrothalassitis, Pol Puigdemont, Noam Itzhak Levi, and Volkan Cevher. Ascent fails to forget. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*, volume 132 of *Proceedings of Machine Learning Research*, pp. 931–962, 2021.

Thanh Nguyen-Tang and Raman Arora. Exact unlearning in reinforcement learning. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. arXiv:2606.04182.

Chaofan Pan, Lingfei Ren, Xiangyu Jiang, Yanhua Li, Xuemei Cao, Xiangkun Wang, Hao Yu, Wei Wei, and Xin Yang. TOUR: A trajectory-level unlearning benchmark for offline reinforcement learning. *arXiv preprint arXiv:2607.21111*, 2026.

Martin Pawelczyk, Jimmy Z. Di, Yiwei Lu, Gautam Kamath, Ayush Sekhari, and Seth Neel. Machine unlearning fails to remove data poisoning attacks. In *International Conference on Learning Representations*, 2025.

Yijun Quan, Wentai Wu, and Giovanni Montana. Exact federated continual unlearning for ridge heads on frozen foundation models. In *Machine Learning and Knowledge Discovery in Databases. Research Track (ECML PKDD 2026)*, volume 16946 of *Lecture Notes in Computer Science*, pp. 114–131. Springer, 2027. doi: 10.1007/978-3-032-37673-2_7. URL `https://link.springer.com/chapter/10.1007/978-3-032-37673-2_7`. First published online 5 September 2026.

Yan Scholten, Stephan Günnemann, and Leo Schwinn. A probabilistic perspective on unlearning and alignment for large language models. In *International Conference on Learning Representations*, 2025.

Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 18075–18086, 2021.

Tamás Takács and László Gulyás. Targeted behavioral unlearning for discrete on-policy reinforcement learning. In *Proceedings of the Intelligent Robotics Fair (IntRob)*, 2026. URL `https://www.tamastheactual.com/papers/tbu/`. To appear.

Anvith Thudi, Hengrui Jia, Ilia Shumailov, and Nicolas Papernot. On the necessity of auditable algorithmic definitions for machine unlearning. In *31st USENIX Security Symposium (USENIX Security 22)*, pp. 4007–4022, 2022.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, 2005.

Dayong Ye, Tianqing Zhu, Congcong Zhu, Derui Wang, Kun Gao, Zewei Shi, Sheng Shen, Wanlei Zhou, and Minhui Xue. Reinforcement unlearning. In *Network and Distributed System Security Symposium (NDSS)*, 2025.

Zichun Ye, Runqi Wang, Xuchuang Wang, Xutong Liu, Shuai Li, and Mohammad Hajiesmaili. Unlearning offline stochastic multi-armed bandits. *arXiv preprint arXiv:2605.00638*, 2026.

Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. Negative preference optimization: From catastrophic collapse to effective unlearning. In *First Conference on Language Modeling*, 2024.