Unlearning Diffusion Policies via Relative Fisher Forgetting

Manuel Kelly[1], **Yingxue Zhang**[2], **Fangzhou Lin**[3,4], **Yanhua Li**[4], **Xin Zhang**[1*]
San Diego State University, USA[1], Binghamton University, USA[2],
Texas A&M University, USA[3], Worcester Polytechnic Institute, USA[4]
mkelly5736@sdsu.edu, yzhang42@binghamton.edu,
fangzhoulin1@tamu.edu, yli15@wpi.edu, xzhang19@sdsu.edu

Abstract

As diffusion-based offline reinforcement learning (RL) moves closer to real deployment, it becomes critical to remove the influence of specific training data for privacy, safety, and regulatory compliance. Since retained data may overlap with or generalize from the forget data, strict retraining equivalence can be ill-posed in offline RL. Existing unlearning methods are ineffective for diffusion policies, as training influence is dispersed across the denoising process and reinforced by critic values. We introduce Relative Fisher Forgetting (RFF), the first principled framework for selective unlearning in diffusion-based offline RL. RFF combines two asymmetric components: critic-side value suppression removes residual value incentives associated with the forget set, eliminating Q-guidance pathways that would otherwise sustain forgotten behaviors; actor-side relative-Fisher updates attenuate forget-set-dominant parameter influence in the denoising policy, reducing behavioral regrowth. To stabilize training, RFF alternates actor-critic updates and employs gradient clipping and retain-set regularization. Experiments on MuJoCo benchmarks show that RFF achieves the lowest identifiable forget-set reliance among baselines while preserving retained performance, and remains over $12\times$ more efficient than retraining. When undesired behaviors are primarily supported by the forget set, RFF additionally suppresses them without collateral degradation.

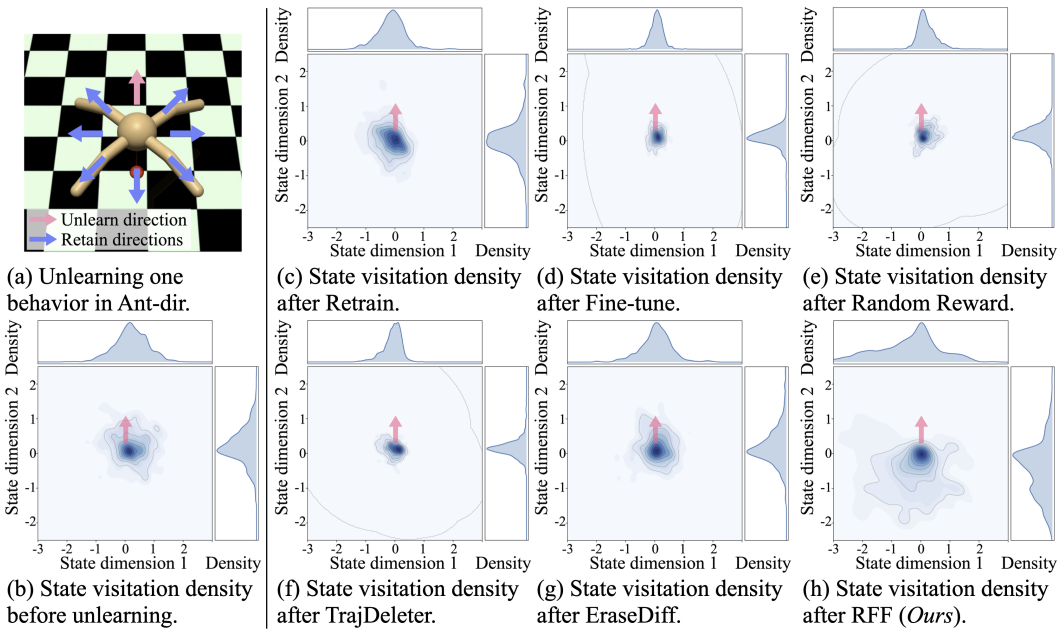
1 Introduction

Offline reinforcement learning (RL) [18, 20] offers a promising path for deploying learning-based decision-making systems in real-world domains where active environment interaction is costly, risky, or infeasible [6, 26, 25]. Recently, diffusion models have emerged as expressive policy classes for offline RL [21, 28, 5, 4, 16], enabling multi-modal action distributions and high-quality action generation through iterative denoising.

As offline RL systems advance to practical deployment, a critical problem emerges: machine unlearning [23, 3]. Unlearning in offline RL aims to remove the influence of designated training data without the prohibitive cost of retraining. It is increasingly important for compliance with governance requirements (*e.g.*, the GDPR "right-to-be-forgotten"), and for maintaining adaptive, trustworthy decision-making systems. In practice, unlearning may require removing specific data (*e.g.*, data from an opt-out user) and, when undesired modes are primarily induced by the forget set, suppressing broader behavioral modes (*e.g.*, forgetting reckless driving), to safeguard privacy, mitigate unsafe or outdated behaviors, and enable post-hoc correction of deployed policies. These requirements are difficult to satisfy with standard retraining because offline RL datasets are temporally correlated: retained data may partially overlap with, or generalize from, the forget data. As a result, "behave as if

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Unlearning one behavior in Ant-dir.

(b) State visitation density before unlearning.

(c) State visitation density after Retrain.

(d) State visitation density after Fine-tune.

(e) State visitation density after Random Reward.

(f) State visitation density after TrajDeleter.

(g) State visitation density after EraseDiff.

(h) State visitation density after RFF (*Ours*).

Figure 1: Visualization of state visitation densities for behavior-level unlearning on Ant-Dir. In (a), the pink arrow indicates the designated unlearn direction, while the purple arrows indicate retained directions. (b) shows the pretrained policy before unlearning, and panels (c)–(h) show the policies after applying Retain, Fine-tune, Random Reward, TrajDeleter, EraseDiff, and RFF, respectively. Darker regions indicate higher visitation frequency. Compared with the baselines, our RFF most effectively suppresses visitation along the unlearn direction while preserving the retained directions.

never trained on the forget data" can be ill-defined, and the core goal becomes removing identifiable reliance on the forget data while preserving utility. We adopt this influence-based view throughout: the goal is not to erase all statistically similar behaviors, but to remove the identifiable parameter-level contribution of the forget set while maintaining retained utility.

Unique challenges. While unlearning has been extensively studied in supervised learning [3] and RL [13, 32], it remains unexplored for diffusion-based offline RL. *First,* unlike supervised settings where data is *i.i.d.* and label-guided, offline RL data is temporally extended and governed by sequential transition dynamics, requiring unlearning to preserve long-horizon consistency and retained-task utility. *Second,* actions are produced via multi-step stochastic denoising, which diffuses sample influences across the full sampling trajectory, making targeted removal difficult. *Third,* diffusion policies are trained through noise-conditioned reconstruction and reward supervision, complicating credit assignment and undermining reward-based unlearning strategies used in RL.

Limitations of state-of-the-art. These challenges render existing approaches inadequate. Offline RL unlearning methods such as TrajDeleter [13] assume explicit policy parameterizations and are not directly compatible with denoising-based generative policies. Conversely, diffusion-model unlearning methods developed for vision or language generation [9, 10, 34, 30, 29] do not account for critic guidance, sequential decision-making, or offline RL constraints. In both cases, the fundamental obstacle is the closed-loop coupling between denoising dynamics and critic guidance: removing forget-set influence requires coordinated intervention in both the generative policy and the value function, a requirement that neither class of existing methods addresses. Thus, unlearning diffusion-based offline RL requires a framework aligned with both denoising dynamics and RL objectives.

Our RFF. We propose Relative Fisher Forgetting (RFF), the first principled framework for selective unlearning in diffusion-based offline RL. RFF targets trajectory-level influence removal as its primary objective, with behavior-level suppression arising when undesired modes are predominantly supported by the forget set. It combines two asymmetric mechanisms. *Critic-side value suppression* removes the residual value incentives associated with the forget set, eliminating the Q-guidance pathways through which forgotten behaviors would otherwise be sustained. *Actor-side relative-Fisher updates* attenuate forget-set-dominant parameter influence in the denoising dynamics, reducing the risk that forgotten

behaviors re-emerge through residual generative pathways. By alternating actor-critic updates with stabilization safeguards, RFF coordinates these mechanisms for robust selective unlearning. As illustrated in Fig. 1, RFF successfully suppresses the unlearn direction while preserving valid behaviors. *Our contributions are as follows*[2]*:*

- We propose RFF, the first principled framework for selective unlearning in diffusion-based offline RL. Under dataset overlap, strict behavior erasure is often ill-posed; we therefore advocate trajectory-level influence removal as a more operational objective and design RFF accordingly.
- We identify an asymmetric division of roles in diffusion offline RL unlearning: critic-side value suppression removes value incentives for forget-supported behaviors, while actor-side relative-Fisher updates attenuate forget-set-dominant parameter influence and reduce behavioral regrowth.
- We conduct extensive experiments on MuJoCo benchmarks, showing that RFF achieves effective forgetting with minimal utility loss, outperforming baselines in both efficacy and efficiency.

2 Preliminaries

2.1 Diffusion-based Offline Reinforcement Learning

We consider a Markov Decision Process (MDP) as $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma \rangle$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition dynamics $\mathcal{T}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$, reward function $r(\boldsymbol{s}, \boldsymbol{a})$, and discount factor $\gamma \in (0, 1)$. In offline RL, the agent is given a static dataset $\mathcal{D} \triangleq \{(\boldsymbol{s}, \boldsymbol{a}, r, \boldsymbol{s}')\}$, collected under an unknown behavior policy π_b, without environment access.

Diffusion-based offline RL [21, 28, 5, 4] aims to learn a policy $\pi_\theta(a|s)$ parameterized as a conditional diffusion model that generates actions through a denoising process, $\pi_\theta(\boldsymbol{a}|\boldsymbol{s}) = p_\theta(\boldsymbol{a}^{0:N}|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{a}^N; \mathbf{0}, \boldsymbol{I}) \prod_{i=1}^{N} p_\theta(\boldsymbol{a}^{i-1}|\boldsymbol{a}^i, \boldsymbol{s})$, where $\boldsymbol{a}^i$ is an intermediate noisy action in the denoising chain. The denoising model $p_\theta(\boldsymbol{a}^{i-1}|\boldsymbol{a}^i, \boldsymbol{s})$ is modeled as a Gaussian distribution whose mean is predicted by a neural network. Following [15], this network is typically implemented in the noise-prediction form as $\boldsymbol{\epsilon}_\theta$. We denote diffusion timesteps by superscripts $i \in \{1, \dots, N\}$ and environment timesteps by subscripts $t \in \{1, \dots, T\}$ to distinguish between the two temporal axes.

The actor is trained with a hybrid loss combining behavior cloning with critic guidance [28, 17], *i.e.*,

$$\mathcal{L}_{\text{actor}}(\theta; \mathcal{D}) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\mathcal{D}, \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I}), i\sim\texttt{Unif}[1,N]} \left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{a}^i, \boldsymbol{s}, i) \right\|^2 \right] - \alpha \cdot \mathbb{E}_{\boldsymbol{s}\sim\mathcal{D}, \boldsymbol{a}^0\sim\pi_\theta}[Q_\phi(\boldsymbol{s}, \boldsymbol{a}^0)],$$

where $\boldsymbol{a}^0$ is the denoised action, and α balances imitation and value guidance [5]. Gradients from the critic are backpropagated through the entire denoising chain, steering the policy toward high-return actions. The critic $Q_\phi(\boldsymbol{s}, \boldsymbol{a})$ is trained via temporal-difference (TD) regression:

$$\mathcal{L}_{\text{TD}}(\phi; \mathcal{D}) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a},r,\boldsymbol{s}')\sim\mathcal{D}} \Big[\big(r + \gamma \mathbb{E}_{\boldsymbol{a}'\sim\pi_\theta(\cdot|\boldsymbol{s}')}[Q_{\hat{\phi}}(\boldsymbol{s}', \boldsymbol{a}')] - Q_\phi(\boldsymbol{s}, \boldsymbol{a}) \big)^2 \Big].$$

where $Q_{\hat{\phi}}(\boldsymbol{s}, \boldsymbol{a})$ is an ensemble-based or EMA-smoothed target network [17].

Together, the actor and critic form a diffusion-based offline RL agent in which the policy is both imitation-grounded and value-guided. This coupling makes diffusion policies expressive but also complicates unlearning, as both denoising dynamics and critic values must be modified consistently.

2.2 Machine Unlearning in Offline RL

Machine unlearning aims to remove the influence of specific training data from a learned model without retraining from scratch [3, 23, 13]. Let $\mathcal{D}$ denote the training dataset, with a forget set $\mathcal{D}_\text{f} \subset \mathcal{D}$ and retained data $\mathcal{D}_\text{r} = \mathcal{D} \setminus \mathcal{D}_\text{f}$. Given a training algorithm $\mathcal{R}$ producing $\pi = \mathcal{R}(\mathcal{D})$, an unlearning algorithm $\mathcal{U}$ produces an updated policy $\pi' = \mathcal{U}(\pi, \mathcal{D}, \mathcal{D}_\text{f})$.

In correlated offline RL, the classical "as-if-unseen" objective is ill-posed: because $\mathcal{D}_\text{r}$ may overlap with or generalize from $\mathcal{D}_\text{f}$ (Fig. 2), a retrained model $\mathcal{R}(\mathcal{D}_\text{r})$ may still reproduce behaviors statistically similar to those induced by $\mathcal{D}_\text{f}$. We thus adopt an *influence-based* view formalized below:

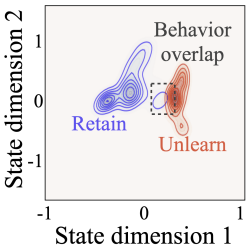


Figure 2: $\mathcal{D}_\text{f}$ and $\mathcal{D}_\text{r}$ overlap in state visitation in Ant-dir.

[2]Our code is available at `https://github.com/STAI-SDSU/RFF`.

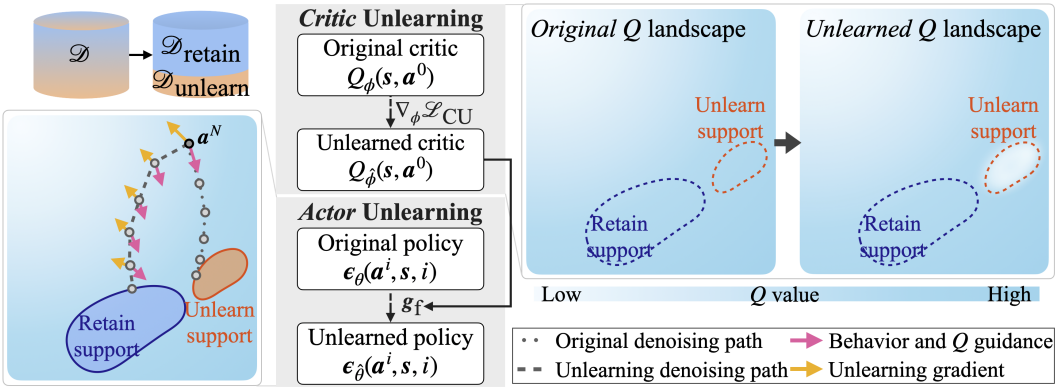


Figure 3: RFF framework. RFF combines *critic-side value suppression*, which removes residual value incentives associated with $\mathcal{D}_{\text{f}}$, and *actor-side influence removal*, which attenuates $\mathcal{D}_{\text{f}}$-dominant parameter influence in the denoising policy to reduce behavioral regrowth. The denoising paths (left) show the policy shifting away from forget-supported behaviors, while the Q-landscape (right) shows suppression of $\mathcal{D}_{\text{f}}$-associated value regions while maintaining retained behaviors.

Definition 1 (Selective Unlearning in Offline RL)**.** *We say* $\pi' = \mathcal{U}(\pi, \mathcal{D}, \mathcal{D}_{\text{f}})$ *satisfies selective unlearning if:*

i) Influence removal. *The learned parameters of* π' *exhibit reduced identifiable reliance on* $\mathcal{D}_{\text{f}}$.

ii) Utility preservation. *Performance on* $\mathcal{D}_{\text{r}}$ *is maintained, i.e.,* $\mathcal{J}(\pi'; \mathcal{D}_{\text{r}}) \approx \mathcal{J}(\pi; \mathcal{D}_{\text{r}})$.

iii) Efficiency. *The computational cost of* $\mathcal{U}$ *is substantially lower than retraining, i.e.,* $\text{cost}(\mathcal{U}) \ll \text{cost}(\mathcal{R}(\mathcal{D}_{\text{r}}))$. *We do* not *require* $\pi' \approx \mathcal{R}(\mathcal{D}_{\text{r}})$.

We distinguish two manifestations: *Trajectory-level unlearning* is the primary objective, removing the identifiable contribution of $\mathcal{D}_{\text{f}}$ to learned parameters, even when similar patterns remain in $\mathcal{D}_{\text{r}}$. *Behavior-level suppression* is a conditional outcome, arising when the corresponding behavior is predominantly supported by $\mathcal{D}_{\text{f}}$.

3 Offline RL Unlearning via Relative Fisher Forgetting

We propose Relative Fisher Forgetting (RFF) (Fig. 3), the first principled framework for selective unlearning in diffusion-based offline RL. RFF targets the influence removal objective of Def. 1, using the relative Fisher ratio between forget and retain sets to adjust parameters predominantly associated with $\mathcal{D}_{\text{f}}$, while preserving utility on $\mathcal{D}_{\text{r}}$ without retraining. RFF achieves this through two asymmetric mechanisms: *i) critic-side value suppression* serves as the primary driver for unlearning by removing residual value incentives associated with $\mathcal{D}_{\text{f}}$; *ii) actor-side relative-Fisher updates* use noise-aware forgetting gradients with relative Fisher preconditioning to attenuate $\mathcal{D}_{\text{f}}$-dominant parameter influence in the diffusion policy. Stabilization coordinates the two via alternating updates, gradient clipping, and retain-set regularization.

3.1 Critic Unlearning: Critic-side Value Suppression

Unlearning in diffusion-based offline RL must act at the critic level: if samples from $\mathcal{D}_{\text{f}}$ retain high relative Q-values, the actor can be pulled back toward forgotten behaviors even after parameter updates. We therefore apply retain-calibrated value suppression, which removes the relative value advantage of $\mathcal{D}_{\text{f}}$ without collapsing the overall critic scale.

Why retain-calibrated? The goal is not to minimize absolute Q-values on $\mathcal{D}_{\text{f}}$, but to remove their relative advantage over $\mathcal{D}_{\text{r}}$-supported values. Under dataset overlap, a retrained critic need not assign low absolute values to $\mathcal{D}_{\text{f}}$; it only needs to remove the value advantage that would cause Q-guidance to reintroduce forgotten actions (Fig. 8). We therefore set a suppression floor τ_{floor} as the q-quantile of stopped-gradient TD targets on $\mathcal{D}_{\text{r}}$, anchoring suppression to the retained value scale.

The critic is thus trained with two complementary terms: *i)* TD regression on $\mathcal{D}_\mathrm{r}$, *i.e.*, $\mathcal{L}_\mathrm{TD}(\phi; \mathcal{D}_\mathrm{r})$, with bootstrapping performed only on retained samples to prevent leakage from $\mathcal{D}_\mathrm{f}$ into the value targets; and *ii)* A hinge-style suppression penalty on $\mathcal{D}_\mathrm{f}$,

$$\mathcal{L}_\mathrm{sup}(\phi; \mathcal{D}_\mathrm{f}) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\mathcal{D}_\mathrm{f}}[\max(0, Q_\phi(\boldsymbol{s},\boldsymbol{a}) - \tau_\mathrm{floor})].$$

The overall critic loss is,

$$\mathcal{L}_\mathrm{CU}(\phi; \mathcal{D}_\mathrm{f}, \mathcal{D}_\mathrm{r}) \triangleq \lambda\mathcal{L}_\mathrm{TD}(\phi; \mathcal{D}_\mathrm{r}) + \mathcal{L}_\mathrm{sup}(\phi; \mathcal{D}_\mathrm{f}), \tag{1}$$

where $\lambda > 0$ balances $\mathcal{D}_\mathrm{r}$ fitting against $\mathcal{D}_\mathrm{f}$ suppression. In practice, we warm up λ and clip the suppression gradient to avoid unstable updates.

3.2 Actor Unlearning: Actor-side Relative Fisher Updates

Critic-side value suppression removes residual value incentives associated with $\mathcal{D}_\mathrm{f}$, but it is insufficient on its own: diffusion policies encode training data through multi-step denoising dynamics, allowing $\mathcal{D}_\mathrm{f}$ influence to persist and re-emerge even after critic suppression. Actor-side updates therefore serve to attenuate $\mathcal{D}_\mathrm{f}$-dominant parameter influence while protecting retain-critical parameters.

Unlearning gradient. Given $(\boldsymbol{s},\boldsymbol{a}) \in \mathcal{D}_\mathrm{f}$, we align actor-side influence removal with the diffusion sampling mechanism by drawing a diffusion step $i \sim \texttt{Unif}[1, N]$ and forming a noisy action $\boldsymbol{a}^i$ via the forward process. From $\boldsymbol{a}^i$, the actor predicts noise $\boldsymbol{\epsilon}_\theta(\boldsymbol{a}^i, \boldsymbol{s}, i)$ and recovers a multi-step denoised estimate $\hat{\boldsymbol{a}}^0$ by reversing the diffusion from timestep i. Using $\hat{\boldsymbol{a}}^0$ instead of the fully denoised action $\boldsymbol{a}^0$ applies the update consistently along the denoising path and reduces the risk that residual influences at earlier diffusion steps regrow forgotten behaviors. The per-sample loss is given by, $\ell_\theta(\boldsymbol{s},\boldsymbol{a}, i, \boldsymbol{\epsilon}) = \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{a}^i, \boldsymbol{s}, i)\right\|^2 - \alpha \cdot Q_\phi(\boldsymbol{s}, \hat{\boldsymbol{a}}^0)$, where the first term enforces denoising consistency and the second provides critic supervision on the predicted clean action. The forgetting gradient is then,

$$\boldsymbol{g}_\mathrm{f}(\boldsymbol{s},\boldsymbol{a}) := \mathbb{E}_{i\in\texttt{Unif}[1,N],\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\nabla_\theta \ell_\theta(\boldsymbol{s},\boldsymbol{a}, i, \boldsymbol{\epsilon})\right], \tag{2}$$

which captures both the supervised denoising signal and the path-wise influence of the value term through the diffusion trajectory. Ascending $\boldsymbol{g}_\mathrm{f}$ does not simply aim to maximize ℓ_θ, but rather to move parameters away from regions where $\mathcal{D}_\mathrm{f}$ induces low-loss attractors, thereby reducing identifiable reliance on those trajectories. In practice, we approximate this expectation using Monte Carlo samples over diffusion steps and noise, with truncated sampling for computational efficiency.

Retain-aware safe direction. While Eq. (2) gradients identify directions for reducing the influence of $\mathcal{D}_\mathrm{f}$, directly applying them can destabilize parameters that are also important for $\mathcal{D}_\mathrm{r}$. Effective forgetting must therefore concentrate on directions that are informative for $\mathcal{D}_\mathrm{f}$ but not critical for preserving $\mathcal{D}_\mathrm{r}$. To characterize parameter importance with respect to each dataset, we estimate diagonal Fisher matrices [1, 7] on both datasets: $[F_\theta]_j^\mathrm{data} = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\in\mathcal{D}_\mathrm{data},\, i\in\texttt{Unif}[1,N],\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\left(\frac{\partial \ell_\theta(\boldsymbol{s},\boldsymbol{a},i,\boldsymbol{\epsilon})}{\partial \theta_j}\right)^2\right]$, where data $\in \{\mathrm{f}, \mathrm{r}\}$, and formalize this as a bi-objective constrained update:

Proposition 2 (Retain-Aware Safe Direction)**.** *Let $\boldsymbol{g}_\mathrm{f} \triangleq \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\in\mathcal{D}_\mathrm{f}}[\boldsymbol{g}_\mathrm{f}(\boldsymbol{s},\boldsymbol{a})]$ denote the aggregated unlearning gradients over $\mathcal{D}_\mathrm{f}$ and $\boldsymbol{g}_\mathrm{r} \triangleq \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\in\mathcal{D}_\mathrm{r}}[\boldsymbol{g}_\mathrm{r}(\boldsymbol{s},\boldsymbol{a})]$ denote those over $\mathcal{D}_\mathrm{r}$. Consider:*

$$\max_{\Delta_\theta} \boldsymbol{g}_\mathrm{f}^\top \Delta_\theta - \boldsymbol{g}_\mathrm{r}^\top \Delta_\theta \quad \textit{s.t.} \quad \Delta_\theta^\top [F_\theta]^\mathrm{r} \Delta_\theta \le \rho,$$

where $[F_\theta]^\mathrm{r}$ is the empirical Fisher on $\mathcal{D}_\mathrm{r}$. Under the diagonal Fisher approximation, the optimal update satisfies:

$$\Delta^*_{\theta,j} \propto \frac{[\boldsymbol{g}_\mathrm{f}]_j - [\boldsymbol{g}_\mathrm{r}]_j}{[F_\theta]^\mathrm{r}_j}. \tag{3}$$

Prop. 2 characterizes an ideal retain-aware, forget-prioritized update direction. Directly incorporating $\boldsymbol{g}_\mathrm{r}$ into the update would require jointly estimating gradients on both $\mathcal{D}_\mathrm{f}$ and $\mathcal{D}_\mathrm{r}$ at each step, which introduces high-variance interactions between forgetting and retention signals and can destabilize training in practice. We instead adopt a surrogate based on the relative Fisher ratio, which provides a low-variance, geometry-aware proxy for approximating the relative contribution of $[\boldsymbol{g}_\mathrm{f}]_j$ over $[\boldsymbol{g}_\mathrm{f}]_j$ in Eq. (3), yielding,

$$\theta_j \leftarrow \theta_j + \eta \cdot \frac{[F_\theta]^\mathrm{f}_j}{[F_\theta]^\mathrm{r}_j + \epsilon'} \cdot [\boldsymbol{g}_\mathrm{f}]_j, \tag{4}$$

where η controls the forgetting rate and $\epsilon' > 0$ ensures numerical stability.

Remark (Justification of the Surrogate)**.** The update in Eq. (4) is a low-variance surrogate for the retain-aware direction in Prop. 2, rather than its exact solution. Prop. 2 requires estimating $(\boldsymbol{g}_\mathrm{f} - \boldsymbol{g}_\mathrm{r})$, which can introduce high variance and unstable cancellation when $\mathcal{D}_\mathrm{f}$ and $\mathcal{D}_\mathrm{r}$ overlap. Instead, we use a Fisher-weighted forgetting gradient $[F]^\mathrm{f} \oslash ([F]^\mathrm{r} + \epsilon') \odot \boldsymbol{g}_\mathrm{f}$, which leverages more stable second-order statistics to emphasize forget-dominant parameters while attenuating retain-critical ones. When the pretrained policy is near convergence, per-parameter gradient magnitudes tend to scale with the square root of the corresponding Fisher information [22], so the ratio $[F]^\mathrm{f}_j / [F]^\mathrm{r}_j$ approximates the relative dominance of $[\boldsymbol{g}_\mathrm{f}]_j$ over $[\boldsymbol{g}_\mathrm{r}]_j$ without requiring explicit gradient cancellation. Under heavy overlap ($\boldsymbol{g}_\mathrm{f} \approx \boldsymbol{g}_\mathrm{r}$, $[F]^\mathrm{f} \approx [F]^\mathrm{r}$), the surrogate reduces to a scaled forgetting gradient rather than the near-zero optimal update, potentially introducing residual updates on shared parameters. We mitigate this effect via retain-set regularization and gradient clipping, and empirically observe stable performance (Tab. 7), and direction alignment (Appx. A.4).

3.3 Stabilizing the Unlearning Process

Because actor and critic updates operate on coupled actor-critic dynamics, stabilization is essential to maintain consistency between behavioral suppression and parameter-level forgetting. For this, we add two standard safeguards. *Gradient control* clips the L_2-norm of aggregated unlearning gradients to prevent unstable parameter jumps. *Retain-set regularization* penalizes deviations from the pre-unlearning policy on $\mathcal{D}_\mathrm{r}$, anchoring the policy to retained behaviors and preventing erosion of useful knowledge. Together, these elements enable RFF to selectively remove the influence of unlearning samples while preserving performance on retained data.

4 Experiment

We design our experiments to answer three key questions: *(Q1)* Can RFF reduce identifiable reliance on the forget set while preserving retained-task utility? *(Q2)* When an undesired behavior is primarily supported by the forget set, can RFF suppress that behavior without substantial collateral degradation? *(Q3)* What roles do critic-side suppression, actor-side influence removal, and stabilization play in faithful selective unlearning?

4.1 Experiment Setup

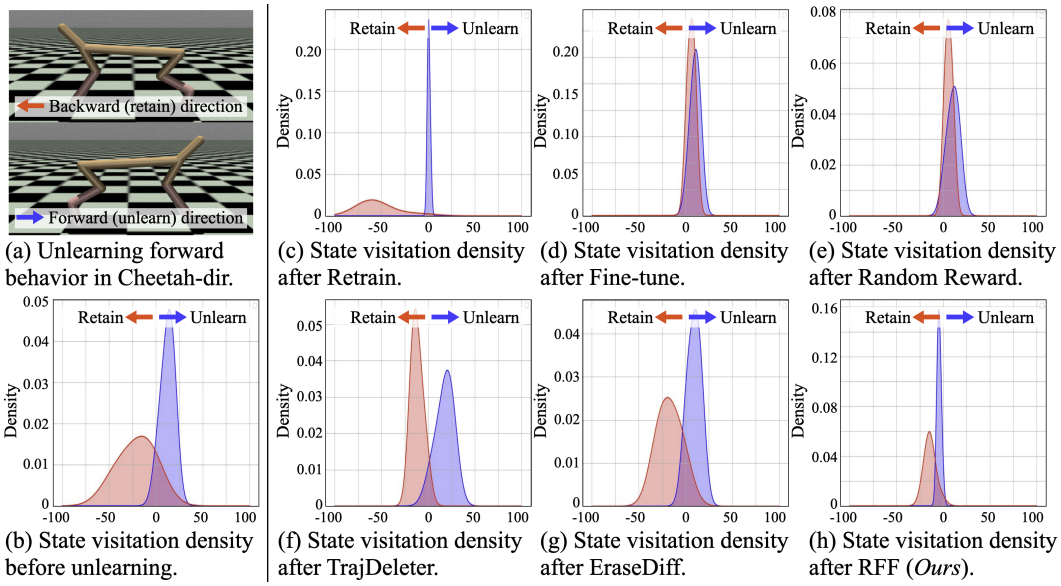
Tasks and datasets. We evaluate on both single-task and multi-task offline RL benchmarks to test trajectory-level and behavior-level unlearning. For single-task settings, we use MuJoCo control environments [27] (Hopper, HalfCheetah, Walker2D) with D4RL medium-expert datasets [8], following [13]. For each task, we randomly sample 1% and 5% of the training data to unlearn. For multi-task settings, we adopt Cheetah-Dir, Cheetah-Vel, and Ant-Dir, where forgetting one task naturally tests behavior-level suppression. Following [31], datasets are collected using SAC-trained policies. In each environment, we randomly sample one task as the behavior to unlearn, corresponding to 50%, 10% and 4% of the data in Cheetah-Dir, Cheetah-Vel, and Ant-Dir, respectively. Details are in Appx. A.3.

Baselines. We focus on unlearning diffusion policies [28], a widely used policy class in offline RL. Extending our framework to other diffusion-based approaches [4, 21, 16] is left for future work. We compare RFF against four baselines following [13]: *Retrain*, which trains from scratch on $\mathcal{D}_\mathrm{r}$ and serves as a reference for the classical "as-if-unseen" objective, though not an oracle for influence removal under overlap; *Fine-tuning*, which continues training on $\mathcal{D}_\mathrm{r}$ for a limited number of iterations; *Random-reward*, which replaces rewards in $\mathcal{D}_\mathrm{f}$ with random values before fine-tuning; and *TrajDeleter [13]*, which suppresses Q-values on $\mathcal{D}_\mathrm{f}$ while maximizing those on $\mathcal{D}_\mathrm{r}$. We additionally include *EraseDiff* [30], a state-of-the-art diffusion unlearning method that perturbs score estimates on $\mathcal{D}_\mathrm{f}$ while regularizing on $\mathcal{D}_\mathrm{r}$. All methods start from the same pre-trained diffusion policy and critic, and we match total update budgets for fairness. Details in Appx. A.3.

Evaluation metrics and protocol. Consistent with our influence-based view, we evaluate selective unlearning along three complementary axes. *Trajectory-level reliance on $\mathcal{D}_\mathrm{f}$:* For single-task settings, we use TrajAuditor [13] to assess whether rollouts from the post-unlearning policy remain attributable to $\mathcal{D}_\mathrm{f}$. Lower positive prediction rates indicate weaker identifiable reliance on $\mathcal{D}_\mathrm{f}$. We emphasize that this metric is not intended to certify retraining equivalence; rather, it operationalizes whether the

Table 1: Mean ± standard deviation of TrajAuditor's positive-prediction rates over five random seeds. $\mathcal{D}_{\mathrm{r}}$ indicates that the retaining dataset is not subjected to unlearning. The $\mathcal{D}_{\mathrm{f},1\%}$ and $\mathcal{D}_{\mathrm{f},5\%}$ denote the unlearning dataset with the size of 1% and 5% of the original dataset. On $\mathcal{D}_{\mathrm{r}}$ the higher positive predictions the better, and on $\mathcal{D}_{\mathrm{f}}$, the lower the better.

	Hopper			HalfCheetah			Walker2D		
Method	$\mathcal{D}_{\mathrm{r}}$	$\mathcal{D}_{\mathrm{f},1\%}$	$\mathcal{D}_{\mathrm{f},5\%}$	$\mathcal{D}_{\mathrm{r}}$	$\mathcal{D}_{\mathrm{f},1\%}$	$\mathcal{D}_{\mathrm{f},5\%}$	$\mathcal{D}_{\mathrm{r}}$	$\mathcal{D}_{\mathrm{f},1\%}$	$\mathcal{D}_{\mathrm{f},5\%}$
Retrain	85.8 ± 0.8	85.2 ± 0.7	84.3 ± 0.7	93.7 ± 0.7	93.2 ± 0.5	92.9 ± 0.9	95.0 ± 0.3	95.2 ± 0.7	95.3 ± 0.6
Fine-Tune	84.9 ± 1.1	28.7 ± 0.9	25.9 ± 1.4	92.7 ± 0.6	58.0 ± 1.3	31.0 ± 1.5	94.7 ± 0.8	58.3 ± 1.0	46.3 ± 1.3
Random Reward	86.0 ± 0.9	30.3 ± 1.9	28.8 ± 0.6	93.4 ± 1.1	67.8 ± 0.9	57.5 ± 1.1	95.2 ± 0.4	60.8 ± 1.7	57.5 ± 1.1
TrajDeleter	86.1 ± 1.3	13.8 ± 0.5	12.1 ± 1.0	93.3 ± 1.0	5.7 ± 0.4	4.7 ± 0.4	95.1 ± 0.6	11.0 ± 0.8	9.9 ± 0.8
EraseDiff	86.4 ± 0.8	12.1 ± 0.4	14.0 ± 0.9	93.1 ± 0.4	5.6 ± 0.5	5.2 ± 0.5	95.0 ± 0.7	12.1 ± 0.8	11.7 ± 0.9
RFF (Ours)	85.6 ± 0.7	$\mathbf{10.4 \pm 1.0}$	$\mathbf{11.6 \pm 0.7}$	93.4 ± 0.4	$\mathbf{4.5 \pm 0.5}$	$\mathbf{4.2 \pm 0.8}$	95.1 ± 0.8	$\mathbf{9.6 \pm 1.0}$	$\mathbf{8.5 \pm 0.6}$



(a) Unlearning forward behavior in Cheetah-dir.
(b) State visitation density before unlearning.
(c) State visitation density after Retrain.
(d) State visitation density after Fine-tune.
(e) State visitation density after Random Reward.
(f) State visitation density after TrajDeleter.
(g) State visitation density after EraseDiff.
(h) State visitation density after RFF (*Ours*).

Figure 4: Visualization of state visitation densities for behavior-level unlearning on Cheetah-Dir. In (a), the forward direction is designated as the unlearn behavior, while the backward direction is retained. Panel (b) shows the pretrained policy before unlearning, and panels (c)–(h) show the policies after applying Retain, Fine-tune, Random Reward, TrajDeleter, EraseDiff, and our RFF, respectively. Our RFF most effectively suppresses visitation along the forward unlearn direction while preserving the backward retain direction.

post-unlearning policy still carries identifiable forget-set influence. *Behavior-level suppression:* For multi-task settings, we evaluate whether an undesired behavior or task is selectively suppressed by measuring return drops on the designated unlearn task. We interpret this as evidence of behavior-level unlearning only when the undesired mode is primarily supported by $\mathcal{D}_{\mathrm{f}}$; when the same mode also appears in $\mathcal{D}_{\mathrm{r}}$, complete erasure is not expected. *Retained-task utility and efficiency:* We report normalized return on retained single-task environments and average normalized return across retained tasks in the multi-task setting, together with wall-clock runtime and gradient steps. This allows us to assess whether forgetting is achieved without unnecessary collateral degradation, and whether it is substantially cheaper than retraining.

4.2 Unlearning Performance Analysis

Trajectory-level unlearning. Tab. 1 reports trajectory-level unlearning results on Hopper, HalfCheetah, and Walker2D. High positive prediction rates on $\mathcal{D}_{\mathrm{r}}$ are expected and desirable, as they indicate that the learned policy continues to generalize over the retained data; the key signal of effective unlearning is the reduction on $\mathcal{D}_{\mathrm{f}}$. Notably, Retrain does not fully eliminate TrajAuditor positives on

$\mathcal{D}_\mathrm{f}$. This is not anomalous: in correlated offline RL, retained trajectories can still support behaviors statistically similar to those in $\mathcal{D}_\mathrm{f}$, so retraining on $\mathcal{D}_\mathrm{r}$ need not remove all forget-like attribution signals. This reinforces our position that retraining equivalence is an incomplete proxy for faithful unlearning under overlap. Fine-tuning and Random Reward reduce auditor predictions to some extent, but often leave high scores on $\mathcal{D}_\mathrm{f}$, indicating that shallow updates do not reliably remove forget-set reliance. TrajDeleter reduces predictions more substantially but exhibits inconsistent forgetting across tasks, while EraseDiff improves forgetting in HalfCheetah yet remains unstable. In contrast, RFF achieves the lowest positive prediction rates on forget data while preserving strong retained-data performance, demonstrating effective and robust trajectory-level influence removal. The reported standard deviations are small relative to the differences between RFF and the strongest competing baselines, particularly on the forget sets. This indicates that the observed forgetting performance is consistent across random seeds rather than being driven by a single training run.

Behavior-level unlearning. Fig. 1 illustrates the state visitation in Ant-Dir before and after unlearning. Before unlearning, the policy explores nearly uniformly over all directions. After applying RFF, the unlearn direction is effectively suppressed while retained directions remain intact. We interpret this as behavior-level unlearning only when the undesired mode is primarily supported by $\mathcal{D}_\mathrm{f}$; under correlated offline data, complete behavioral erasure is not expected when the same mode remains strongly supported by $\mathcal{D}_\mathrm{r}$. Fine-tuning and Random Reward often leave visible traces of the unlearn behavior, while Retrain still exhibits partial drift toward the undesired direction due to task correlations. Return-drop results in Fig. 6(c) further confirm that RFF most strongly suppresses the undesired behavior while maintaining satisfactory retained-task performance. Consistent observations are shown for Cheetah-Dir in Fig. 4 and Cheetah-Vel in Fig. 7 in Appx. A.4.

The failure modes of baselines reveal a consistent pattern. Fine-tuning and Random Reward apply only weak supervision signals without explicit forget-set targeting, leaving D_f's influence embedded in both actor and critic. TrajDeleter addresses the critic but leaves actor parameters imprinted with D_f trajectories; Q-suppression alone cannot prevent the actor from imitating forgotten behaviors through the denoising process. EraseDiff modifies the actor's score function but leaves critic Q-values intact; residual value incentives from D_f continue to attract the actor back toward forgotten behaviors through Q-guidance. This structural analysis validates the necessity of joint actor-critic unlearning: in diffusion-based offline RL, the closed-loop coupling between denoising dynamics and value guidance means that single-component intervention is systematically insufficient.

To assess generalization beyond MuJoCo locomotion, we additionally evaluate RFF on the MetaWorld ML10 robotic-manipulation benchmark [33]. We designate `window-open-v3` as the forget task and use the remaining tasks to evaluate retained-task utility; the results are reported in Appx. A.4.

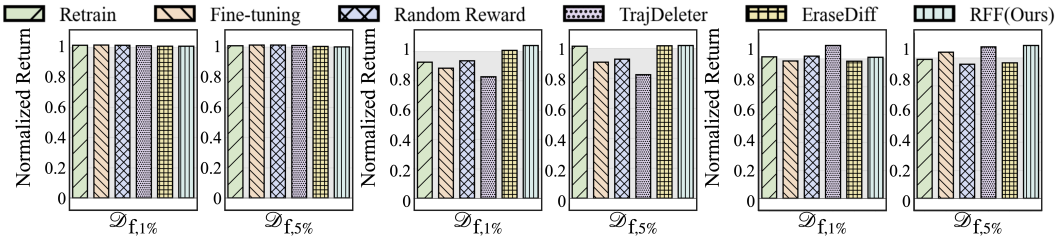
4.3 Utility & Efficiency Comparison

Utility evaluation. Fig. 5 reports normalized returns on retained single-task environments after unlearning. Across Hopper, HalfCheetah, and Walker2D, RFF consistently preserves higher retained-task performance than competing baselines, indicating strong utility retention under trajectory-level unlearning. For behavior-level settings, Fig. 6 shows return changes on retained and unlearn tasks. Among all baselines, RFF best preserves retained behaviors while most strongly suppressing the unlearned ones, highlighting its ability to remove targeted behavior modes without unnecessary collateral degradation. This contrast is especially clear in Cheetah-Vel, where forgetting one speed conflicts strongly with retaining others: simpler baselines often collapse across tasks, whereas RFF better maintains the forgetting-utility trade-off. TrajDeleter and EraseDiff also struggle because they intervene primarily in only one component of the actor-critic system, leaving residual influence in either the policy or the value function.

Table 2: Runtime and gradient steps (GS) in Ant-Dir.

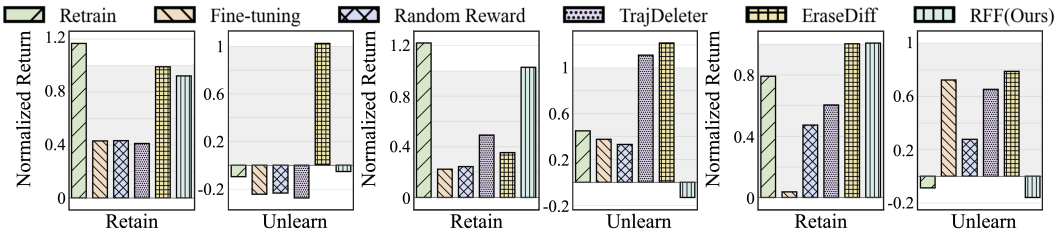
Method	Time	GS
Retrain	364 min	100k
Finetuning	26 min	200
Random Reward	26 min	200
TrajDeleter	30 min	75
EraseDiff	26 min	75
RFF (Ours)	29 min	75

Efficiency evaluation. Tab. 2 reports runtime and gradient steps on Ant-Dir. Retraining is prohibitively expensive, requiring over 100k gradient steps. All approximate unlearning methods are substantially more efficient, with RFF requiring only 75 steps while achieving superior forgetting and utility preservation. Although alternating actor-critic updates introduce slight overhead, RFF



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

Figure 5: Normalized returns on Hopper, Half-Cheetah, Walker2D after unlearning. Results are shown for forgetting ratios $\mathcal{D}_{\mathrm{f},1\%}$ and $\mathcal{D}_{\mathrm{f},5\%}$, highlighting that RFF maintains high performance on retain data compared to baselines.



(a) Cheetah-dir. (b) Cheetah-vel. (c) Ant-dir.

Figure 6: Normalized return on Cheetah-Dir, Cheetah-Vel, and Ant-Dir. Higher returns on retain tasks indicate better utility, while lower returns on unlearn tasks reflect stronger forgetting. Grey background bars show the average normalized returns of retain/unlearn tasks.

remains almost $12\times$ faster than retraining and more effective than competing baselines. More results are provided in Appx. A.4.

4.4 Ablation Study

To validate the asymmetric roles of critic-side suppression, actor-side influence removal, and stabilization, we ablate RFF into three variants: RFF(A✓/C×), which disables critic unlearning; RFF(A×/C✓), which disables actor unlearning; and RFF−, which removes the stabilization mechanism. For each variant, we measure two metrics on the Ant-dir task: the retain performance change ΔR_{r}, defined as the return difference before and after unlearning (lower is better, indicating less degradation on retain data), and the unlearning success ΔR_{f}, defined as the return difference before and after unlearning on the forget set (higher is better, indicating more effective forgetting).

Table 3: Ablation results in Ant-Dir.

Variants	$\Delta R_{\mathrm{r}}\downarrow$	$\Delta R_{\mathrm{f}}\uparrow$
RFF(A✓/C×)	**0.5**	52.5
RFF(A×/C✓)	69.3	86.1
RFF-	58.7	<u>508.3</u>
RFF(A✓/C✓)	<u>44.9</u>	**514.2**

Tab. 3 reveals an asymmetric division of roles among the three components. When critic-side suppression is removed (RFF(A✓/C×)), retain performance remains highly stable, but forgetting efficacy drops sharply, indicating that actor-side updates alone are insufficient to induce behavioral forgetting. This shows that critic-side suppression is the primary driver of unlearning efficacy, since forgotten behaviors can persist whenever the critic continues to assign them favorable values. Conversely, when actor-side influence removal is removed (RFF(A×/C✓)), critic suppression still produces substantial forgetting, but retained-task performance degrades severely. This is because the actor is still updated through critic guidance, so critic-side suppression continues to reshape the realized policy; however, without actor-side relative-Fisher protection, these updates are not steered away from retain-critical parameters. As a result, forgetting occurs, but with substantially larger collateral damage to retained behaviors. Removing stabilization yields strong suppression but causes substantial retained-task degradation, showing that coordinated actor-critic stabilization is necessary to prevent utility collapse during unlearning. The full method achieves the best overall balance, combining strong forgetting with substantially better retain preser-

vation than the critic-only variant. Overall, these results empirically validate the asymmetric design of RFF: critic-side suppression serves as the primary mechanism for unlearning, actor-side updates improve faithfulness and retain-data preservation, and stabilization coordinates the two to enable robust selective unlearning.

5 Conclusion, Limitations and Impact

We presented Relative Fisher Forgetting (RFF), a principled framework for *selective unlearning* in diffusion-based offline RL. Instead of enforcing strict retraining equivalence, RFF removes the identifiable influence of the forget set while preserving retained-task utility through critic-side value suppression and actor-side relative-Fisher updates. Experiments across trajectory-level and behavior-level settings show that RFF achieves effective forgetting with strong utility retention and substantially better efficiency than retraining and prior baselines.

RFF has several limitations. It assumes explicit access to the forget set and is evaluated only on simulated, state-based locomotion and robotic-manipulation benchmarks. Moreover, under correlated offline data, complete behavioral erasure is not always expected when the same behavior is also supported by the retain set; in such cases, the more appropriate objective is to remove forget-set influence rather than enforce absolute behavioral absence. Stronger theoretical guarantees under overlapping support, distribution shift, and partial deletion requests also remain open.

This work can improve trustworthiness and governance in decision-making systems by supporting data-deletion requests, correction of unsafe or outdated behaviors, and post-hoc auditing without full retraining. Potential risks include misuse to evade accountability or unintended utility degradation when the forget set is misspecified or heavily overlaps with retained behaviors. We therefore emphasize utility monitoring, conservative stabilization, and transparent reporting of forgetting outcomes.

6 Acknowledgment

We thank the anonymous reviewers for their constructive feedback, which helped improve the presentation of this paper. Manuel Kelly and Xin Zhang were supported in part by NSF awards IIS-2449864 and IIS-2617313. Yingxue Zhang was supported in part by the NSF under Grants IIS-2539629 (CAREER) and IIS-2617314. Yanhua Li was supported in part by NSF grant IIS-1942680 (CAREER).

References

[1] Abhishek Aich. Elastic weight consolidation (ewc): Nuts and bolts. *arXiv preprint arXiv:2105.04093*, 2021.

[2] Bijal Bharadva, Manuel Kelly, Yingxue Zhang, and Xin Zhang. Cure: Conservative unlearning with soft-gating regularization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 2026.

[3] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE symposium on security and privacy (SP)*, pages 141–159. IEEE, 2021.

[4] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[5] Linjiajie Fang, Ruoxue Liu, Jing Zhang, Wenjia Wang, and Bing-Yi Jing. Diffusion actor-critic: Formulating constrained policy iteration as diffusion noise regression for offline reinforcement learning. *arXiv preprint arXiv:2405.20555*, 2024.

[6] Xing Fang, Qichao Zhang, Yinfeng Gao, and Dongbin Zhao. Offline reinforcement learning for autonomous driving with real world driving data. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 3417–3422. IEEE, 2022.

[7] Jack Foster, Stefan Schoepf, and Alexandra Brintrup. Fast machine unlearning without retraining through selective synaptic dampening. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 12043–12051, 2024.

[8] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: datasets for deep data-driven reinforcement learning. *CoRR*, abs/2004.07219, 2020.

[9] Rohit Gandikota, Joanna Materzynska, Jaden Fiotto-Kaufman, and David Bau. Erasing concepts from diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2426–2436, 2023.

[10] Rohit Gandikota, Hadas Orgad, Yonatan Belinkov, Joanna Materzyńska, and David Bau. Unified concept editing in diffusion models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5111–5120, 2024.

[11] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[12] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9304–9312, 2020.

[13] Chen Gong, Kecen Li, Jin Yao, and Tianhao Wang. Trajdeleter: Enabling trajectory forgetting in offline reinforcement learning agents. *arXiv preprint arXiv:2404.12530*, 2024.

[14] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019.

[15] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[16] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[17] Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:67195–67212, 2023.

[18] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *CoRR*, abs/2110.06169, 2021.

[19] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

[20] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020.

[21] Liyuan Mao, Haoran Xu, Xianyuan Zhan, Weinan Zhang, and Amy Zhang. Diffusion-dice: In-sample diffusion guidance for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 37:98806–98834, 2024.

[22] James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.

[23] Thanh Tam Nguyen, Thanh Trung Huynh, Zhao Ren, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. A survey of machine unlearning. *arXiv preprint arXiv:2209.02299*, 2022.

[24] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 7(1):2, 2019.

[25] Samarth Sinha, Ajay Mandlekar, and Animesh Garg. S4rl: Surprisingly simple self-supervision for offline reinforcement learning in robotics. In *Conference on Robot Learning*, pages 907–917. PMLR, 2022.

[26] Shengpu Tang, Maggie Makar, Michael Sjoding, Finale Doshi-Velez, and Jenna Wiens. Leveraging factored action spaces for efficient offline reinforcement learning in healthcare. *Advances in Neural Information Processing Systems*, 35:34272–34286, 2022.

[27] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[28] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

[29] Chen Wu, Sencun Zhu, and Prasenjit Mitra. Federated unlearning with knowledge distillation. *arXiv preprint arXiv:2201.09441*, 2022.

[30] Jing Wu, Trung Le, Munawar Hayat, and Mehrtash Harandi. Erasediff: Erasing data influence in diffusion models. *arXiv preprint arXiv:2401.05779*, 2024.

[31] Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua Tenenbaum, and Chuang Gan. Prompting decision transformer for few-shot policy generalization. In *international conference on machine learning*, pages 24631–24645. PMLR, 2022.

[32] Dayong Ye, Tianqing Zhu, Congcong Zhu, Derui Wang, Kun Gao, Zewei Shi, Sheng Shen, Wanlei Zhou, and Minhui Xue. Reinforcement unlearning. *arXiv preprint arXiv:2312.15910*, 2023.

[33] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Avnish Narayan, Hayden Shively, Adithya Bellathur, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, 2021.

[34] Yu Zhou, Dian Zheng, Qijie Mo, Renjie Lu, Kun-Yu Lin, and Wei-Shi Zheng. Decoupled distillation to erase: A general unlearning method for any class-centric tasks. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 20350–20359, 2025.

A Appendix

A.1 Overview of the RFF Algorithm

We summarize Relative Fisher Forgetting (RFF) as an alternating actor-critic procedure that removes the influence of the forget set while preserving retained behavior. RFF operates through coordinated components: *i)* Critic update, which suppresses value estimates on $\mathcal{D}_\mathrm{f}$ while fitting TD targets on $\mathcal{D}_\mathrm{r}$; *ii)* Actor update, which applies relative Fisher-weighted forgetting gradients to attenuate parameters associated with $\mathcal{D}_\mathrm{f}$ while protecting those important for $\mathcal{D}_\mathrm{r}$; *iii)* Stabilization, which ensures consistent interaction between actor and critic through gradient clipping and retain-set regularization. After each actor update, we apply stabilization to prevent excessive deviation from the pre-unlearning policy on $\mathcal{D}_\mathrm{r}$. The full procedure is summarized in Algorithm 1.

Algorithm 1 Relative Fisher Forgetting.

Input: Retain data $\mathcal{D}_\mathrm{r}$, Unlearn data $\mathcal{D}_\mathrm{f}$; actor π_θ, critic Q_ϕ, target critic $Q_{\hat{\phi}}$; η, λ.
Initialize EMA states $M_\mathrm{f}{=}\mathbf{0}, M_\mathrm{r}{=}\mathbf{0}, t_F{=}0$
Save pre-unlearning actor snapshot $\pi_{\theta^{pre}} \leftarrow \pi_\theta$.
while not converged **do**
  $Q_\phi, Q_{\hat{\phi}} \leftarrow$CRITICUNLEARNING$(Q_\phi, Q_{\hat{\phi}}, \pi_\theta, \mathcal{D}_\mathrm{r}, \mathcal{D}_\mathrm{f}, \lambda)$
  $\pi_\theta, M_\mathrm{f}, M_\mathrm{r}, t_F \leftarrow$ACTORRFF$(\pi_\theta, \mathcal{D}_\mathrm{f}, \mathcal{D}_\mathrm{r}, \eta, M_\mathrm{f}, M_\mathrm{r}, t_F)$
end while
Return π_θ, Q_ϕ

Critic Update Details. The critic update corresponds to Eq. (1) in the main paper, combining TD regression on $\mathcal{D}_\mathrm{r}$ with value suppression on $\mathcal{D}_\mathrm{f}$. Algorithm 2 computes TD targets using only retained samples to prevent leakage from $\mathcal{D}_\mathrm{f}$, and applies a hinge-style suppression penalty to ensure that $Q_\phi(s,a)$ on $\mathcal{D}_\mathrm{f}$ does not exceed a retain-calibrated threshold. This removes the relative value advantage of $\mathcal{D}_\mathrm{f}$ while preserving the overall value scale.

Algorithm 2 CRITICUNLEARNING

Input: $Q_\phi, Q_{\hat{\phi}}, \pi_\theta$; datasets $\mathcal{D}_\mathrm{r}, \mathcal{D}_\mathrm{f}$; λ, δ_Q;
for $t = 1$ to $\frac{|\mathcal{D}_\mathrm{f}|}{|\mathcal{B}_\mathrm{f}|}$ **do**
  Sample $\mathcal{B}_\mathrm{r} \subset \mathcal{D}_\mathrm{r}$, $\mathcal{B}_\mathrm{f} \subset \mathcal{D}_\mathrm{f}$
  $y(\boldsymbol{s}, \boldsymbol{a}, r, \boldsymbol{s}') \leftarrow r + \gamma\, \mathbb{E}_{\boldsymbol{a}' \sim \pi_\theta(\cdot|\boldsymbol{s}')}\big[Q_{\hat{\phi}}(\boldsymbol{s}', \boldsymbol{a}')\big]$
  $\mathcal{L}_\mathrm{TD} \leftarrow \frac{1}{|\mathcal{B}_\mathrm{r}|} \sum_{(\boldsymbol{s},\boldsymbol{a},r,\boldsymbol{s}') \in \mathcal{B}_\mathrm{r}} (Q_\phi(\boldsymbol{s}, \boldsymbol{a}) - y)^2$
  $\tau_\text{floor} \leftarrow \texttt{qQuantile}\Big(\{\texttt{stopgrad}\big(Q_\phi(\boldsymbol{s}, \boldsymbol{a})\big) : (\boldsymbol{s}, \boldsymbol{a}) \in \mathcal{B}_\mathrm{r}\}\Big)$
  $\mathcal{L}_\text{sup} \leftarrow \frac{1}{|\mathcal{B}_\mathrm{f}|} \sum_{(\boldsymbol{s},\boldsymbol{a}) \in \mathcal{B}_\mathrm{f}} \max\big(0,\, Q_\phi(\boldsymbol{s}, a) - \tau_\text{floor}\big)$
  $\phi \leftarrow \phi - \nabla_\phi\big(\lambda \cdot \mathcal{L}_\mathrm{TD} + \mathcal{L}_\text{sup}\big)$
  $\hat{\phi} \leftarrow \tau_\text{target}\, \phi + (1{-}\tau_\text{target})\, \hat{\phi}$
end for
Return $Q_\phi, Q_{\hat{\phi}}$

Actor Update Details. The actor update implements Eq. (4) using a relative Fisher preconditioner. Fisher matrices are estimated via exponential moving averages of squared gradients on both datasets. Since the update depends on the relative contrast between $\mathcal{D}_\mathrm{f}$ and $\mathcal{D}_\mathrm{r}$, approximate Fisher estimation is sufficient.

A.2 Proof Sketch for Prop. 2

Proof. Let $\boldsymbol{g} = \boldsymbol{g}_\mathrm{f} - \boldsymbol{g}_r$. Under the diagonal Fisher approximation, the constraint becomes $\sum_j [F_\theta]^\mathrm{r}_j \Delta^2_{\theta,j} \le \rho$. When $\boldsymbol{g} \neq 0$, the linear objective is maximized on the boundary of this Fisher ball, so the constraint is active. The Lagrangian is

$$\mathcal{L}(\Delta_\theta, \mu) = \boldsymbol{g}^\top \Delta_\theta - \mu(\Delta_\theta^\top [F]^\mathrm{r} \Delta_\theta - \rho).$$

Algorithm 3 UPDATEFISHERS (bias-corrected diagonal EMAs)

Input: Batches $\mathcal{B}_\mathrm{f} \subset \mathcal{D}_\mathrm{f}$, $\mathcal{B}_\mathrm{r} \subset \mathcal{D}_\mathrm{r}$, actor π_θ, EMA states $M_\mathrm{f}, M_\mathrm{r}, t_F$, factor β_F.
$\boldsymbol{h}_\mathrm{f} \leftarrow \frac{1}{|\mathcal{B}_\mathrm{f}|}\sum_{(\boldsymbol{s},\boldsymbol{a})\in\mathcal{B}_\mathrm{f}}\big(\nabla_\theta \ell_\theta(\boldsymbol{s},\boldsymbol{a},i,\boldsymbol{\epsilon})\big)^{\odot 2}$
$\boldsymbol{h}_\mathrm{r} \leftarrow \frac{1}{|\mathcal{B}_\mathrm{r}|}\sum_{(\boldsymbol{s},\boldsymbol{a})\in\mathcal{B}_\mathrm{r}}\big(\nabla_\theta \ell_\theta(\boldsymbol{s},\boldsymbol{a},i,\boldsymbol{\epsilon})\big)^{\odot 2}$
$M_\mathrm{f} \leftarrow \beta_F M_\mathrm{f} + (1-\beta_F)\,\boldsymbol{h}_\mathrm{f}; \quad M_\mathrm{r} \leftarrow \beta_F M_\mathrm{r} + (1-\beta_F)\,\boldsymbol{h}_\mathrm{r}; \quad t_F \mathrel{+}= 1$
$F^\mathrm{f} \leftarrow M_\mathrm{f}/(1-\beta_F^{t_F}); \quad F_\mathrm{r} \leftarrow M_\mathrm{r}/(1-\beta_F^{t_F})$
Return $F^\mathrm{f}, F^\mathrm{r}, M_\mathrm{f}, M_\mathrm{r}, t_F$

Algorithm 4 ACTORRFF

Input: $\pi_\theta, \mathcal{D}_\mathrm{f}, \mathcal{D}_\mathrm{r}; \eta$; EMA states $M_\mathrm{f}, M_\mathrm{r}, t_F$
for $t = 1$ to $\frac{|\mathcal{D}_\mathrm{f}|}{|\mathcal{B}_\mathrm{f}|}$ **do**
  Sample $\mathcal{B}_\mathrm{f} \subset \mathcal{D}_\mathrm{f}$, $\mathcal{B}_\mathrm{r} \subset \mathcal{D}_\mathrm{r}$
  $(F^\mathrm{f}, F^\mathrm{r}, M_\mathrm{f}, M_\mathrm{r}, t_F) \leftarrow$ UPDATEFISHERS$(\mathcal{B}_\mathrm{f}, \mathcal{B}_\mathrm{r}, \pi_\theta, M_\mathrm{f}, M_\mathrm{r}, t_F)$
  $\boldsymbol{g}_\mathrm{f} \leftarrow \frac{1}{|\mathcal{B}_\mathrm{f}|}\sum_{(\boldsymbol{s},\boldsymbol{a})\in\mathcal{B}_\mathrm{f}} \nabla_\theta \ell_\theta(\boldsymbol{s},\boldsymbol{a},i,\boldsymbol{\epsilon})$;
  $W \leftarrow F^\mathrm{f} \oslash (F^\mathrm{r}+\epsilon'); \quad \Delta\theta \leftarrow \eta\,(W \odot \boldsymbol{g}_\mathrm{f})$
  $\theta \leftarrow \theta + \Delta\theta$
end for
Return Updated $\pi_\theta, M_\mathrm{f}, M_\mathrm{r}, t_F$

Setting $\frac{\partial \mathcal{L}}{\partial \Delta_{\theta,j}} = 0$ gives

$$\boldsymbol{g}_j - 2\mu[F]^\mathrm{r}_j \Delta_{\theta,j} = 0.$$

Thus,

$$\Delta^*_{\theta,j} = \frac{\boldsymbol{g}_j}{2\mu[F]^\mathrm{r}_j} \propto \frac{[\boldsymbol{g}_\mathrm{f}]_j - [\boldsymbol{g}_\mathrm{r}]_j}{[F]^\mathrm{r}_j}.$$

The multiplier μ rescales the update to satisfy the radius ρ, yielding Eq. (3). □

A.3 Implementation Details

Efficient Fisher Estimation. While the diagonal Fisher information matrix offers a tractable approximation to local parameter curvature, computing it explicitly across all parameters in diffusion models remains computationally intensive. Therefore, we use online estimation during training and subsampled Fisher estimation to reduce the computational burden of estimating the Fisher matrix [7] in the actor unlearning, without sacrificing its regularization benefits.

Instead of computing the Fisher matrix post hoc, we adopt an online estimation strategy by tracking an exponential moving average (EMA) of squared gradients. We initialize the EMA states $M_\mathrm{f} = M_\mathrm{r} = 0$ at the start of the unlearning phase and update them online during unlearning using mini-batches drawn from $\mathcal{D}_\mathrm{f}$ and $\mathcal{D}_\mathrm{r}$ respectively. Specifically, for each model parameter θ_j, we maintain: $[F_\theta]_j \leftarrow \beta \cdot [F_\theta]_j + (1-\beta) \cdot \left(\frac{\partial \ell_\theta}{\partial \theta_j}\right)^2$, where $\beta \in [0,1)$ is a smoothing coefficient. This approach amortizes the cost of Fisher estimation across training and requires no additional backpropagation passes.

To further reduce overhead, we estimate the Fisher matrix using a small, randomly selected subset of the unlearning dataset $\mathcal{D}_\mathrm{f}$. For each selected sample, we compute gradients at only a few diffusion steps i and noise samples $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, rather than integrating over the full diffusion horizon. This Monte Carlo approximation maintains a good fidelity of the diagonal Fisher estimate while drastically reducing computation.

Since actor unlearning uses Fisher only to capture the relative importance of parameters across unlearning and retaining sets, approximate estimation is sufficient: what matters is the contrast between the two sets, not the exact value of each Fisher term.

Implementation Setup. All results are averaged across the same random seeds and reported with normalized scores. Experiments were conducted on a single NVIDIA RTX 6000 Ada GPU (48GB)

Table 5: Training and unlearning dataset splits for multi-task and single-task MuJoCo benchmarks.

Environment	Training Set	Unlearning Set
Cheetah-dir	1 task [0]	1 task [1]
Cheetah-vel	10 tasks [0, 4, 8, 12, 16, 20, 24, 28, 32, 36]	1 task [28]
Ant-dir	25 tasks [0:2:48]	1 task [40]
Hopper	999,494 samples (∼4998 traj.)	1% (∼50 traj.) / 5% (∼250 traj.)
HalfCheetah	1M samples (5000 traj.)	1% (50 traj.) / 5% (250 traj.)
Walker2d	999,214 samples (∼4997 traj.)	1% (∼50 traj.) / 5% (∼250 traj.)

using PyTorch and Gymnasium libraries. We evaluate our unlearning process using the MuJoCo benchmark [27], chosen for its complexity and wide adoption in prior unlearning studies. This benchmark is particularly well-suited for testing generalized forgetting in diffusion-based policies.

MuJoCo Multi-Task Agents. Following the multi-task protocol of Prompt-DT [31], we evaluate on three locomotion environments: **Cheetah-dir**, **Cheetah-vel**, and **Ant-dir**. These tasks penalize deviations from target goals and are designed to assess task-specific unlearning in few-shot settings.

- **Cheetah-dir:** Two tasks (forward, backward). Reward is proportional to velocity along the target direction.
- **Cheetah-vel:** 10 tasks, each with a target velocity. Reward penalizes the ℓ_2 distance to the target.
- **Ant-dir:** 25 tasks, each with a target direction. Reward is proportional to velocity along the goal direction.

MuJoCo Single-Task Agents. Following TrajDeleter [13], we evaluate on three locomotion environments: **Hopper**, **HalfCheetah**, and **Walker2d**. These tasks use built-in Gymnasium return functions, where higher directional velocity yields higher returns.

Pretraining. For multi-task agents, fine-tuning was necessary to avoid task imbalance:

•**Cheetah-dir:** Equal split of forward/backward led to forward bias. Fine-tuning on backward-only tasks for at least 1k batches (batch size 2048) corrected performance.

•**Cheetah-vel:** Equal split across tasks led to poor performance at high velocities. Fine-tuning on the fastest 5 velocities for 1k batches improved stability.

Table 4: Hyperparameter configurations.

Task	η	β_F	ϵ'	λ	Iters
Cheetah-dir (Backward)	1	0.95	2e-3	2	25
Cheetah-dir (Forward)	1	0.95	2e-3	2	75
Cheetah-vel	1	0.95	2e-3	2	75
Ant-dir	1	0.95	2e-3	2	75

•**Ant-dir:** Pretraining on tasks provided sufficient performance without extra fine-tuning. For single-task agents, full dataset pretraining was sufficient for stable unlearning without fine-tuning.

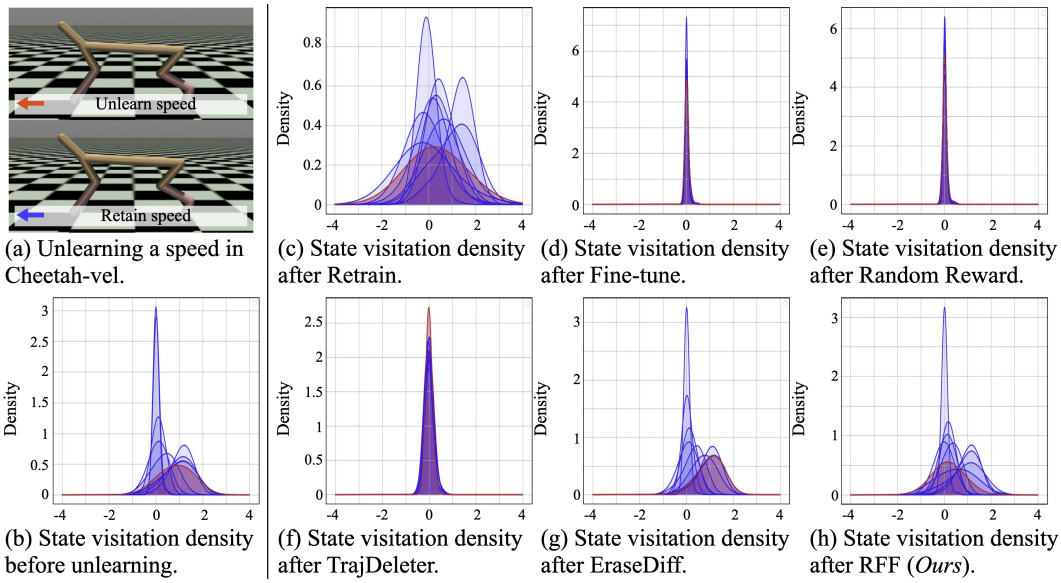
Dataset Splits & Hyperparameters. Dataset split details and hyperparameters are provided in Tab. 5 and Tab. 4 respectively.

A.4 More Experiment Results

Behavior-unlearning Performance. Fig. 7 shows the unlearning result of Cheetah-vel. It shows that RFF is able to unlearn the specific unlearning speed while maintaining the retain speed well. Since this unlearning task involves correlated behaviors, baselines are not able to clearly identify the behavior to unlearn.

Unlearning Efficiency. Tab. 6 shows the runtime for baselines and our RFF. Though our RFF is not the most efficient, we are able to attain good unlearning quality and model utility.

$\mathcal{D}_\mathrm{f}$'s Q-value Distribution Post RFF. We further provide the post-unlearning Q-value distribution on hopper in Fig. 8, showing that RFF does not collapse Q-values on $\mathcal{D}_\mathrm{f}$ to minimal levels. Instead, forgotten samples settle into a lower but non-degenerated range relative to $\mathcal{D}_\mathrm{r}$, confirming that RFF performs retain-calibrated suppression rather than indiscriminative value shrinking.



(a) Unlearning a speed in Cheetah-vel.
(b) State visitation density before unlearning.
(c) State visitation density after Retrain.
(d) State visitation density after Fine-tune.
(e) State visitation density after Random Reward.
(f) State visitation density after TrajDeleter.
(g) State visitation density after EraseDiff.
(h) State visitation density after RFF (*Ours*).

Figure 7: State visitation densities in Cheetah-vel. Our RFF effectively forgets the designated unlearn speed while preserving the retain speeds.

Table 6: Training duration (minutes) across tasks for different unlearning baselines and our method. Retraining is reported at 0%, 1%, and 5% duration.

Methods	Tasks					
	Hopper		HalfCheetah		Walker2D	
	$\mathcal{D}_{\mathrm{f},1\%}$	$\mathcal{D}_{\mathrm{f},5\%}$	$\mathcal{D}_{\mathrm{f},1\%}$	$\mathcal{D}_{\mathrm{f},5\%}$	$\mathcal{D}_{\mathrm{f},1\%}$	$\mathcal{D}_{\mathrm{f},5\%}$
Retraining	267 min	274 min	269 min	281 min	272 min	277 min
Fine-tune	0.52 min	0.52 min	0.52 min	0.52 min	0.50 min	0.50 min
Random-reward	0.52 min	0.52 min	0.52 min	0.52 min	0.52 min	0.52 min
TrajDeleter	9 min	9 min	8 min	8 min	9 min	9 min
EraseDiff	6 min	6 min	6 min	6 min	6 min	6 min
RFF	7 min	7 min	10 min	10 min	11 min	11 min

Why Relative Fisher Weighting. An ablation on Walker2D (5% unlearning) further confirms the necessity of relative Fisher weighting. We tried unlearning using the hard threshold approach in SSD [7] (denoted as SSD-CU) and the $\mathcal{D}_{\mathrm{f}}$-Fisher weighted unlearning (denoted as $\mathcal{D}_{\mathrm{f}}$-Fisher). SSD-CU fails to remove the designated trajectories (93.1% positive predictions), while the $\mathcal{D}_{\mathrm{f}}$-Fisher variant improves forgetting (10.7%) but suffers notable utility loss (0.90). RFF achieves both strong forgetting (8.7%) and high utility (0.94), demonstrating that balancing forget-set and retain-set Fisher signals is essential for stable unlearning. Tab. 7 confirms the necessity of relative Fisher weighting on Walker2D (5% unlearning): SSD-CU fails to remove forget trajectories, $\mathcal{D}_{\mathrm{f}}$-Fisher improves forgetting but degrades utility, while RFF achieves both.

Scalability. We evaluate RFF's performance under substantially larger unlearning ratios of 25% and 50% on Hopper (Tab. 8). Across both settings, RFF maintains strong forgetting and is able to achieve the lowest TrajAuditor scores while preserving competitive normalized returns. This demonstrates that its unlearning dynamics remain stable even when a significant fraction of the dataset is removed. In contrast, retraining exhibits high positive prediction rates (48.4% at 25% unlearning), reflecting the strong correlation structure of the dataset and the diffusion policy's inherent generalization across overlapping state–action regions. This behavior indicates that the classical "as-if-never-seen" criterion is inherently ill-posed for diffusion policies, given their tendency to generalize forgotten behaviors from correlated retain data. RFF directly addresses this challenge through critic-driven behavioral suppression complemented by actor-side influence removal, enabling more faithful and robust unlearning than baselines under large unlearning ratios.

Table 7: Fisher weighting ablation on Walker2D (5% unlearning).

Method	TrajAuditor (↓)	Utility (↑)
SSD-CU [7]	93.1%	0.94
$\mathcal{D}_f$-Fisher	10.7%	0.90
RFF (Ours)	**8.7%**	**0.94**

Table 8: Utility ($\hat{R}$ ↑) *vs.* unlearning efficacy at 25% and 50% unlearning.

	25% Unlearning		**50% Unlearning**	
Method	$\hat{R}$ (↑)	$\mathcal{D}_f$ (↓)	$\hat{R}$ (↑)	$\mathcal{D}_f$ (↓)
Retrain	0.84	48.4%	0.80	5.2%
TrajDeleter	0.84	7.1%	0.84	3.9%
EraseDiff	**0.87**	11.4%	**0.86**	12.5%
RFF (Ours)	0.86	**6.7%**	**0.86**	**3.8%**

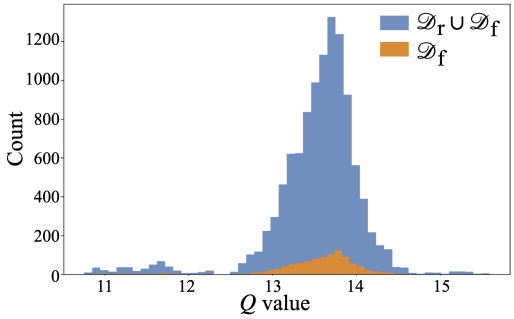


Figure 8: Post-RFF Q-value distribution on Hopper.

Generalization to Robotic Manipulation. To evaluate whether RFF generalizes beyond MuJoCo locomotion, we conduct an additional experiment on MetaWorld ML10 [33], a benchmark containing diverse robotic-manipulation tasks. We designate `window-open-v3` as the forget task and use the remaining nine tasks as the retain tasks. We report the success rate on the designated forget task and the average success rate across the retained tasks. Lower forget-task success indicates stronger behavior suppression, whereas higher retain-task success indicates better utility preservation.

As shown in Tab. 9, RFF completely suppresses the designated `window-open-v3` behavior while preserving the retained-task performance. In contrast, retraining still achieves an 80% success rate on the forget task, despite excluding the designated forget trajectories. This result illustrates that retained data can support behaviorally overlapping skills, making strict retraining equivalence an inadequate proxy for behavior-level forgetting in correlated offline RL. Fine-tuning, Random Reward, and TrajDeleter fail to suppress the forget behavior and also incur substantial degradation on retained tasks. EraseDiff preserves retained performance relatively well but does not remove the designated behavior. Overall, this experiment provides evidence that the joint critic-side suppression and actor-side relative-Fisher update of RFF can extend beyond MuJoCo locomotion to state-based robotic manipulation.

Table 9: MetaWorld ML10 performance. We designate `window-open-v3` to forget and evaluate retained utility on the remaining tasks. Lower forget-task success and higher retain success are better.

Method	Forget-task success ↓	Retain-task average success ↑
Pretrained	100%	90%
Retrain	80%	80%
Fine-tuning	100%	45%
Random-reward	100%	70%
TrajDeleter	100%	55%
EraseDiff	100%	95%
RFF (Ours)	**0%**	**100%**

Sensitivity to the Actor Update Strength. We study the sensitivity of RFF to the actor-side update strength η on Cheetah-Dir, with the forward behavior designated as the forget task and the backward behavior used for retention. Lower forget-task return indicates stronger forgetting, whereas higher retain-task return indicates better utility preservation. The default setting is $\eta = 1$.

When $\eta = 0.1$, the actor update is too weak to fully suppress the forget behavior. The range $\eta \in [1, 10]$ provides a stable operating region, achieving near-zero forget-task return while preserving high retain-task performance. In contrast, excessively large update strengths ($\eta = 100$ and $\eta = 1000$) cause severe degradation on the retain task, indicating policy collapse due to overly aggressive actor updates. These results show that RFF is effective across a nontrivial range of update strengths rather than relying on a single isolated value.

Table 10: Sensitivity to the actor-side update strength η on Cheetah-Dir.

η	Forget-task return ↓	Retain-task return ↑
0.1	370.91	694.42
1	**0.19**	**740.11**
10	0.19	739.56
100	0.67	0.39
1000	-1.40	4.90

Empirical Alignment of the Relative-Fisher Surrogate. Prop. 2 characterizes an ideal retain-aware update direction, $d_{\text{ideal}} = \frac{\boldsymbol{g}_f - \boldsymbol{g}_r}{[\mathbf{F}_\theta]^r}$, where $\boldsymbol{g}_f$ and $\boldsymbol{g}_r$ denote the aggregated actor gradients on the forget and retain sets, respectively. In practice, directly estimating $\boldsymbol{g}_f - \boldsymbol{g}_r$ can be noisy, particularly when the two datasets have overlapping state-action support. RFF therefore uses the relative-Fisher surrogate $d_{\text{RFF}} = \frac{[\mathbf{F}_\theta]^f}{[\mathbf{F}_\theta]^r + \epsilon'} \odot \boldsymbol{g}_f$, where the division is element-wise.

To empirically assess the relationship between the two directions, we evaluate the Pearson correlation, cosine similarity, and coordinate-wise sign agreement between d_{ideal} and d_{RFF}. The comparison is performed on a frozen actor at three stages of the unlearning process: before unlearning, at an intermediate checkpoint, and after unlearning. The results are reported in Tab. 11.

The relative-Fisher surrogate remains positively aligned with the ideal retain-aware direction throughout unlearning. Pearson correlation and cosine similarity decrease moderately at the intermediate checkpoint but recover after unlearning, while coordinate-wise sign agreement increases from 78.86% to 92.90%. These results provide empirical support for the use of the relative-Fisher update as a low-variance surrogate.

Table 11: Empirical alignment between the ideal retain-aware direction from Proposition 2 and the relative-Fisher surrogate.

Checkpoint	Pearson correlation ↑	Cosine similarity ↑	Sign agreement ↑
Pretrained	0.500	0.500	78.86%
Intermediate	0.387	0.385	91.05%
Fully unlearned	0.549	0.544	92.90%

Fisher Sensitivity Analysis. To complement the behavioral evaluation, we measure the change in the model's local Fisher sensitivity on the forget and retain sets. We conduct this analysis on HalfCheetah with a 5% forget set. For each method, the reported Forget Ratio and Retain Ratio are normalized by the corresponding Fisher sensitivity of the pretrained policy. A retain ratio close to one indicates that the local geometry associated with the retained data is largely preserved, whereas a large deviation from one on the forget set indicates a substantial change in the model's sensitivity to the forget data.

RFF produces the largest change in Fisher sensitivity on the forget set, with a Forget Ratio of 66.98, while maintaining a Retain Ratio of 1.07, which is close to the pretrained reference. In comparison, fine-tuning and Random Reward substantially alter the retain-set sensitivity, while EraseDiff causes a strong deviation from the pretrained retain-set geometry. These results provide complementary model-internal evidence that RFF selectively modifies the forget-set-associated parameters while preserving the local geometry relevant to retained data. This analysis is not intended to provide a formal unlearning certificate, but rather to complement the behavioral, state-visitation, critic-value, and trajectory-attribution evaluations.

Table 12: Fisher sensitivity on HalfCheetah with a 5% forget set. The Forget Ratio and Retain Ratio are normalized by the corresponding Fisher sensitivity of the pretrained policy.

Method	Forget Ratio ↑	Retain Ratio ≈ 1
Retrain	37.31	0.81
Fine-tuning	1.06	1.71
Random Reward	1.22	1.60
TrajDeleter	0.73	0.95
EraseDiff	0.97	0.22
RFF (Ours)	**66.98**	**1.07**

A.5 Extended Related Work

Behavior Unlearning and Concept Ablation. Behavior-level unlearning in offline RL is conceptually related to concept ablation in diffusion models, where specific semantic concepts are removed by editing model representations [9, 30]. However, the analogy is limited. Concept ablation typically operates in the latent or score space and targets high-level semantic features ("erase the concept of cars"). In contrast, behavior unlearning in RL is fundamentally data-grounded: the behavioral "concepts" correspond to structured state-action patterns supported by trajectories in the forget set. Rather than editing internal representations, RFF removes the parameter-level influence of forget set through relative Fisher weighting and retain-calibrated value suppression. When a behavior is uniquely induced by the forget set, this influence removal naturally yields behavior-level forgetting; when correlated behaviors also appear in the retain data, strict behavioral erasure is ill-posed. Thus, RFF can be viewed as extending concept-level unlearning into the behavioral domain while remaining firmly rooted in trajectory-level data provenance.

Machine unlearning and generative models. Machine unlearning aims to remove the influence of training data without full retraining, often motivated by privacy regulations and the "right to be forgotten." Early strategies include partition-based retraining (*e.g.*, SISA) [3], influence-function–based updates [14], and distillation to a student model trained only on retained data [12]. More recently, unlearning has been extended to generative models, particularly diffusion and transformer architectures, where methods suppress specific concepts or examples through gradient ascent, distribution re-weighting, or score-function editing [9, 30]. While effective in supervised and generative contexts, these approaches do not address the behavioral dynamics of reinforcement learning, where forgetting must alter both policy actions and value estimates.

Policy editing and unlearning in RL. In reinforcement learning, undesired behaviors are often mitigated by reward penalties, action masking, or constraint learning [11, 24], but these methods impose new incentives rather than removing the influence of unwanted data. Recent work proposes actor-only unlearning techniques, such as randomizing rewards or directly penalizing the forget set [13]. However, such methods cannot fully suppress the critic's value estimates, allowing forgotten behaviors to resurface. A concurrent work, CURE [2], addresses offline RL unlearning in the overlap regime via TD-gradient interference scoring and soft-gated actor-critic suppression, with theoretical guarantees on forgetting descent. In contrast, RFF targets diffusion-based policies, where the additional challenge of multi-step denoising dynamics requires noise-aware forgetting gradients and relative Fisher preconditioning to remove forget-set influence from the generative policy. Our approach differs by jointly unlearning both the actor and the critic, while stabilizing value estimates on retain data to preserve task utility.

Safe reinforcement learning. Safe reinforcement learning (safe RL) focuses on preventing agents from executing unsafe or undesirable actions during training or deployment. Common approaches include reward shaping, constrained policy optimization, shielding, and action masking [11, 24]. These methods proactively discourage unsafe behavior but operate prospectively, requiring constraints or safety signals to be specified in advance. In contrast, unlearning addresses a retroactive setting: the policy has already been trained on data that may later be deemed undesirable, and the goal is to remove its influence without retraining from scratch.

Offline RL and diffusion policies. Offline RL learns policies from static datasets, typically through conservative critics, policy constraints, or behavior regularization to handle distribution shift [20, 19]. Diffusion policies extend this line by parameterizing actions as samples from a conditional denoising process [28, 4, 21, 5], improving multimodality and robustness. However, the challenge of unlearning trajectories in diffusion-parameterized policies has not been addressed. RFF provides the first framework for unlearning in this setting, combining actor–critic suppression with retain-set stabilization to achieve both effective forgetting and performance preservation.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction clearly state the three main claims that RFF is the first framework for unlearning in diffusion-based offline RL; RFF jointly unlearns actor and critic; and RFF outperforms baselines in efficacy and efficiency. These are supported by experiments in Sec. 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes].

Justification: Limitations are discussed in Sec. 5, including: reliance on explicit access to the forget set, evaluation restricted to simulated continuous-control environments and robotics control, and lack of provable guarantees under distribution shift or partial deletion requests.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes].

   Justification: Proof was provided in Appx. A.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes].

Justification: The paper describes the experimental setup, datasets, baselines, and evaluation metrics in detail, including MuJoCo environments, dataset splits, and evaluation protocols (Sec. 4 and Appx. A.3).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: The paper provides a public code repository for reproducibility, along with sufficient experimental details to replicate the reported results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.

- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: The paper specifies training details such as datasets (D4RL), environments (MuJoCo), hyperparameters, and training procedures, with additional implementation details in the Appx A.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

   Justification: Tab. 1 reports mean $\pm$ standard deviation over five random seeds for the trajectory-level unlearning results. The standard deviation captures variability across independent random seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes].

Justification: The paper specifies compute resources, including GPU type (NVIDIA RTX 6000 Ada), and reports runtime and gradient steps for efficiency comparison Tab. 6.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes].

Justification: The research complies with NeurIPS Code of Ethics and does not involve harmful or unethical data collection or usage; it focuses on improving privacy and safety through unlearning mechanisms.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes].

Justification: The paper discusses both positive impacts (privacy compliance, safer policies, regulatory alignment) and potential risks (misuse for accountability evasion, utility degradation) (Sec. 5).

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The paper does not release high-risk models or datasets that require special safeguards; it focuses on algorithmic development and evaluation on standard benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes].

    Justification: The paper uses standard benchmark datasets (*e.g.*, D4RL, MuJoCo) and properly cites prior work and datasets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A].

    Justification: The paper does not introduce new datasets; it provides code but does not define a new dataset requiring documentation.

    Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The paper does not involve human subjects or crowdsourcing.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: No human subject research is involved, so IRB approval is not required.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [No].

    Justification: Large language models are not part of the proposed methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.

- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.