A Related Work

Interpretability in reinforcement learning has become a central research theme because real-world deployment requires agents that are trustworthy and reliable [Arulkumaran et al., 2017, Sutton and Barto, 2018, Milani et al., 2024, Cheng et al., 2025]. Early studies emphasize *feature*-level explanations: they highlight regions of the observation space that most influence an agent's decisions, often through saliency maps or attention heatmaps [Zahavy et al., 2016, Greydanus et al., 2018, Iyer et al., 2018, Mott et al., 2019, Atrey et al., 2020, Puri et al., 2020]. A complementary thread seeks *policy*-level explanations. These works approximate learned policies with human-interpretable rules [Verma et al., 2018, Soares et al., 2020], design transparent architectures [Topin et al., 2021, Demircan et al., 2025], or dissect reward functions to clarify action choices [Juozapaitis et al., 2019, Liu and Zhu, 2025]. More recently, researchers have probed how entire training *trajectories* shape behavior [Deshmukh et al., 2023].

Zooming in further, identifying critical *states* offers a finer-grained view of decision making. Several approaches address offline settings [Guo et al., 2021, Yu et al., 2023, Liu et al., 2023, Rishav et al., 2025]. Closer to our focus are methods that target online RL such as lazy-MDP [Jacq et al., 2022], StateMask [Cheng et al., 2023] and RICE [Cheng et al., 2024]. Lazy-MDP augments the action space with a "lazy" action and penalizes non-lazy choices; states where the agent still acts are interpreted as important. However, this approach requires modifying the training pipeline. StateMask and RICE train an auxiliary mask network alongside the policy, forcing random actions in selected states while keeping returns roughly unchanged; masked states are deemed non-critical. Nevertheless, these methods crucially rely on the policy being sufficiently developed, which limits their applicability when agents are still learning in complex environments.

Moving beyond these constraints, our work introduces data attribution as a principled lens for interpretability in online RL. This approach not only closes a key methodological gap in the literature but also delivers fresh insights for RL researchers and practitioners, and informs more efficient and effective training.

B Detailed Experimental Setups

B.1 Standard RL Environments

We offer a detailed description of the RL environments used in our experiments in Table 1.

Gymnasium and `Highway` are licensed under MIT license; `MiniGrid` is licensed under Apache-2.0 license.

B.2 Experimental setups for standard RL

Training setups. We adopt `Stable-Baselines3`[8] [Raffin et al., 2021] (MIT license) as our training framework for the standard RL experiments. We use PPO [Schulman et al., 2017] as our RL algorithm and adopt the default training hyperparamters and network architectures for most environments unless otherwise specified.

- **Training hyperparameters:** We use n_steps=2048 (i.e., $n = |B^{(k)}| = 2048$), batch_size=64 (i.e., $|\mathcal{B}_j^{(k)}| = 64$), n_epochs=10 (i.e., each rollout buffer will be used for 10 epochs), learning_rate=5e-3 with optimizer=SGD in all environments except `BipedalWalker`, for which we use 3e-4 with Adam. total_timesteps per environment are: 102,400 for `FrozenLake` (50 rounds), 81,920 for `MiniGrid` (40 rounds), 102,400 for `Acrobot` (50 rounds), 204,800 for `Highway` (100 rounds), 307,200 for `LunarLander`

[2] https://minigrid.farama.org/environments/minigrid/EmptyEnv/
[3] https://gymnasium.farama.org/environments/toy_text/frozen_lake/
[4] https://gymnasium.farama.org/environments/classic_control/acrobot/
[5] https://highway-env.farama.org/environments/highway/
[6] https://gymnasium.farama.org/environments/box2d/lunar_lander/
[7] https://gymnasium.farama.org/environments/box2d/bipedal_walker/
[8] https://stable-baselines3.readthedocs.io/en/master/index.html

Table 1: **A summary description of RL environments we use in experiments.** Besides MiniGrid and Highway, other environments are from Gymnasium [Towers et al., 2024].

Env	Env ID & Args	Goal	State Space	Action Space	Reward Structure
`MiniGrid` [Chevalier-Boisvert et al., 2023]	`MiniGrid-Empty-8x8-v0`[2]	Navigate to a target location	$3\times7\times7$ image, representing the egocentric view of the agent's observation	7 **discrete** actions: {turn left, turn right, move forward, pickup, drop, toggle, done}	**Sparse**: 1 - 0.9 (step_count/max_steps) on success, 0 otherwise
`FrozenLake`	`FrozenLake-v1`[3], map=4x4, slippery=False	Navigate from start to goal without falling into holes	1 discrete integer: agent position index on the grid	4 **discrete** actions: {Left, Down, Right, Up}	**Sparse**: +1 on reaching goal, 0 otherwise
`Acrobot`	`Acrobot-v1`[4]	Swing up the link to reach a target height	$\mathbb{R}^6$, providing information about the two rotational joint angles and their angular velocities	3 **discrete** actions: $\{-1, 0, 1\}$ torque (N m)	**Dense**: -1 per step until reaching the target height
`Highway` [Leurent, 2018]	`highway-v0`[5], vehicle_count= 10	Drive at high speed while avoiding collisions	Kinematic Observation: 5×5 array of ego and nearby vehicles, including their location and speed	5 **discrete** actions: {LANE_LEFT, IDLE, LANE_RIGHT, FASTER, SLOWER}	**Dense**: $(v-v_{\min})/(v_{\max}-v_{\min})-b\cdot$ collision at each step
`LunarLander`	`LunarLander-v2`[6]	Land safely on the pad from flight	$\mathbb{R}^8$: the coordinates of the lander, its linear velocities, angle, angular velocity, and whether each leg is in contact with the ground	4 **discrete** actions: {do nothing, fire left, fire main, fire right}	**Dense**: +10 per leg contact; –0.03 per side-engine step; –0.3 per main-engine step; +100 on safe landing; –100 on crash; distance/velocity/angle terms
`BipedalWalker`	`BipedalWalker-v3`[7]	Traverse rough terrain without falling	$\mathbb{R}^{24}$: hull angle speed, angular velocity, horizontal & vertical speed, joints positions & angular speed, legs contact with ground, 10 lidar measurements	4 **continuous** actions: motor speed values in $[-1, 1]$ for 4 joints at hips and knees	**Dense**: +1 per forward step; -100 on fall; small penalty proportional to torque magnitude

(150 rounds), 1,024,000 for `BipedalWalker` (1000 rounds). Other hyperparameters include ent_coef=0.0, clip_range=0.2, gamma=0.99, gae_lambda=0.95, vf_coef=0.5, max_grad_norm=0.5.

- **Network architectures:** For `FrozenLake`, `Acrobot`, `Highway`, `LunarLander`, and `BipedalWalker`, we use the default `MlpPolicy` in Stable-Baselines3. This policy uses two-layer MLP networks (64 hidden units per layer), taking the flattened observation as input. For `MiniGrid` with image input, we use an adapted `CnnPolicy` with a custom feature extractor. The extractor comprises two convolutional layers (with 16 and 32 filters respectively, and 3x3 kernels) followed by a linear layer of 64 hidden units.

Evaluation setups. We evaluate the *stochastic* performance of each policy $\pi_{\theta^{(k)}}$ at every training round k by averaging returns over multiple evaluation episodes. Specifically, we run 1000 episodes for `LunarLander`, `Acrobot`, `MiniGrid`, and `FrozenLake`; and 100 episodes for `Highway` and `BipedalWalker`.

B.3 Experimental setups for RLHF

We follow Hugging Face [2023] to set up this experiment. The base model is a 2.7B parameter GPT-Neo model [Black et al., 2021] (MIT license).

Training setups. We adopt `TRL`[9] [von Werra et al., 2020] (Apache-2.0 license) as our training framework to fine-tune the based model via PPO. The dataset for PPO training is

[9] https://huggingface.co/docs/trl/index

`real-toxicity-prompts`[10] [Gehman et al., 2020] (Apache-2.0 license). For each example, we extract the first 10-15 tokens as a prompt, generate a 30-token continuation, and score it with the reward model, a toxicity detector `LFTW R4 Target`[11][Vidgen et al., 2021]. The reward signal is the raw logits of the label "neutral" of the detector.

The naming of the hyperparameters in `TRL` slightly differs from the ones in `Stable-Baselines3`. Here we stick to the naming in `TRL` to report the hyperparameters and clarfy their meanings using our notations. We follow Hugging Face [2023] to use batch_size=256 (i.e., $n = |B^{(k)}| = 256$), mini_batch_size=1 (i.e., $|\mathcal{B}_j^{(k)}| = 1$), ppo_epochs=4 (i.e., each rollout buffer will be used for 4 epochs), learning_rate=1e-5 with Adam optimizer, and all other default hyperparameters in `TRL`. We train for one epoch over the training dataset, which amounts to 109 rounds in total.

Evaluation setups. We evaluate the performance of each policy $\pi_{\theta^{(k)}}$ at every training round k. Evaluation is performed on `Wiki-Toxic`[12], which is of a different distribution than the training dataset. For each toxic sample, we use the full sample as the prompt (significanlty longer than used in training and thus more likely to elicit toxic continuations), and generate a 30-token continuation (same as the training setup). We then evaluate the toxicity of the generated continuation using another toxicity detector `da-electra-hatespeech-detection`[13]. Evaluation is conducted over 400 samples, and we report the mean toxicity probability.

C Additional Experimental Results

C.1 More demonstrations of harmful records

Harmful records for learning across training rounds. We examine the bottom records w.r.t f^{return} in different training rounds k and present the results in Fig. 8. (Results in the main paper, Fig. 3(a), corresponds to $k = 5$ here.)

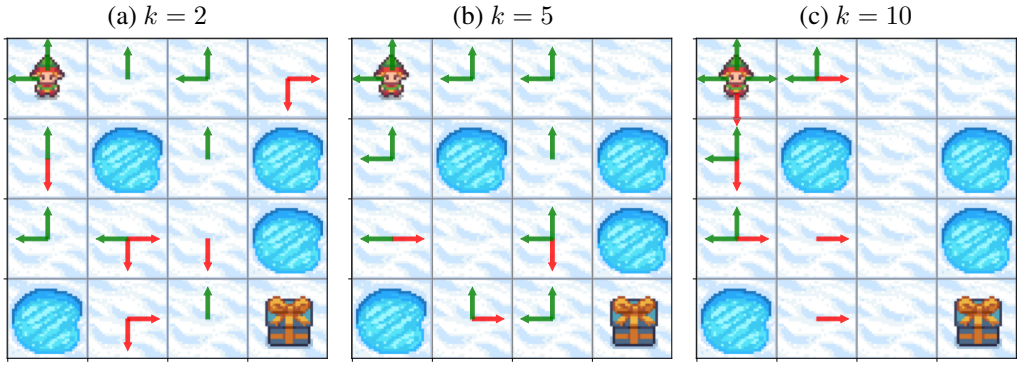

Figure 8: **Bottom records in different training rounds in `FrozenLake`.** Arrow indicates action, green/red indicates positive/negative $\hat{A}$.

Across all three snapshots ($k = 2, 5, 10$), the bottom records share a clear and consistent pattern: inaccurate advantage estimate, rewarding the agent for a poor action (moving away from the goal) and penalizing the agent for a good one (moving towards the goal).

C.2 Quantifying phase change via weighted graph roughness analysis

Measurement protocol. We provide full details of our quantitative investigation.

For each round k, we build the similarity graph $\mathcal{G}_k$ using records with positive influence scores in $B^{(k)}$ and their influence scores [Von Luxburg, 2007]. We embed each record z_i as a node in the graph,

[10] https://huggingface.co/datasets/allenai/real-toxicity-prompts
[11] https://huggingface.co/facebook/roberta-hate-speech-dynabench-r4-target
[12] https://huggingface.co/datasets/OxAISH-AL-LLM/wiki_toxic
[13] https://huggingface.co/alexandrainst/da-hatespeech-detection-base

with the node value being the L_∞-normalized influence score $\tilde{I}_i = {}^{I_i}/_{\|I\|_\infty}$, the node embedding being the record embedding e_i extracted by a well-trained network (obtained at the end of the PPO training). We set edge weights by a Gaussian kernel $w_{ij} = \exp(-\|e_i - e_j\|^2/\sigma^2)$ with σ chosen via the median-distance heuristic. We retain each node's u nearest neighbors when building the similarity graph. This reduces computational cost. In practice, we find that varying u from 20 to 100 has little effect on the roughness measure.

With the graph $\mathcal{G}_k$ built, we compute the graph roughness as follows:

$$\text{Roughness}(\mathcal{G}_k) = \frac{\sum_{i<j} w_{ij}(\tilde{I}_i - \tilde{I}_j)^2}{\sum_{i<j} w_{ij}}$$

We repeat this process for all rounds k and plot the change of roughness over rounds.

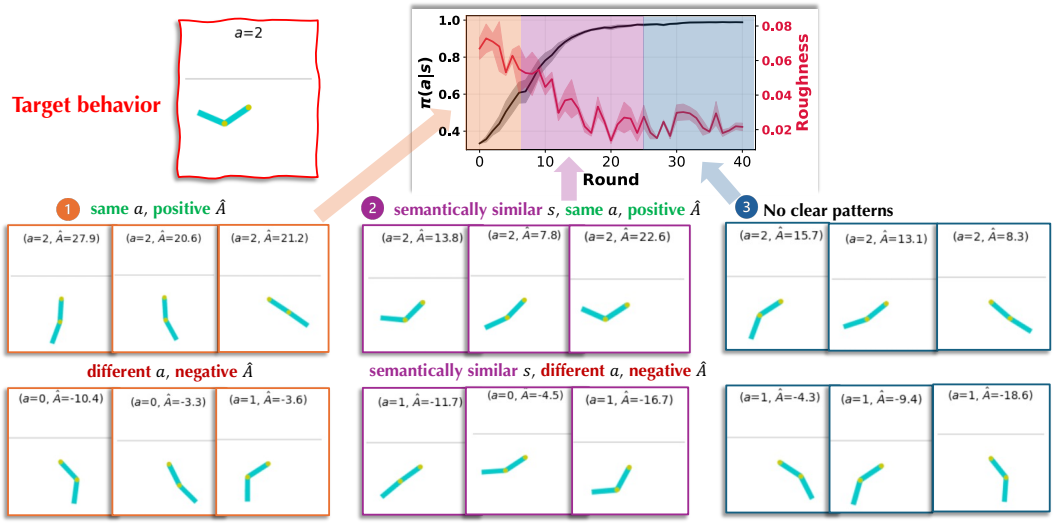

Figure 9: **Phase change of top records in `Acrobot`.**

Results in more environments. We study another environment `Acrobot`, investigating the phase change and measuring the roughness metric across rounds. The results are presented in Fig. 9. We observe a consistent trend of the three phases, aligned with the findings discussed in Sec. 4.2.

In Phase 1, top records include those with the same action and positive $\hat{A}$, and those with alternative actions and negative $\hat{A}$. Roughness is high in this phase. In Phase 2, semantically similar records (that consistently show the action-advantage association) emerge as top records; roughness decreases significantly in this phase. In Phase 3, learning approaches convergence and the semantic clustering stabilizes; influence scores become dominated by noise, causing roughness to show minor fluctuations.

C.3 Additional results for single-round intervention

Fig. 10 (as an extension of Fig. 5) presents the results of single-round interventions in four environments, additionally comparing with the random baseline that discards a similar amount of records.

We discuss several key takeaways: (1) Influence-guided intervention mostly leads to performance gains, while random drop mostly leads to performance degradation. (2) When standard PPO fails to improve (e.g. a dip at round $k = 9$ in `Highway`; see Fig. 6), the attribution signal can become unreliable, producing negative Δ return (see Fig. 10 at $k = 9$ in `Highway`), leading occasionally to interventions that fail to bring any improvement. However, as long as PPO's overall trend is upward, our intervention can effectively *purify* the learning and and drive net improvement over the full run.

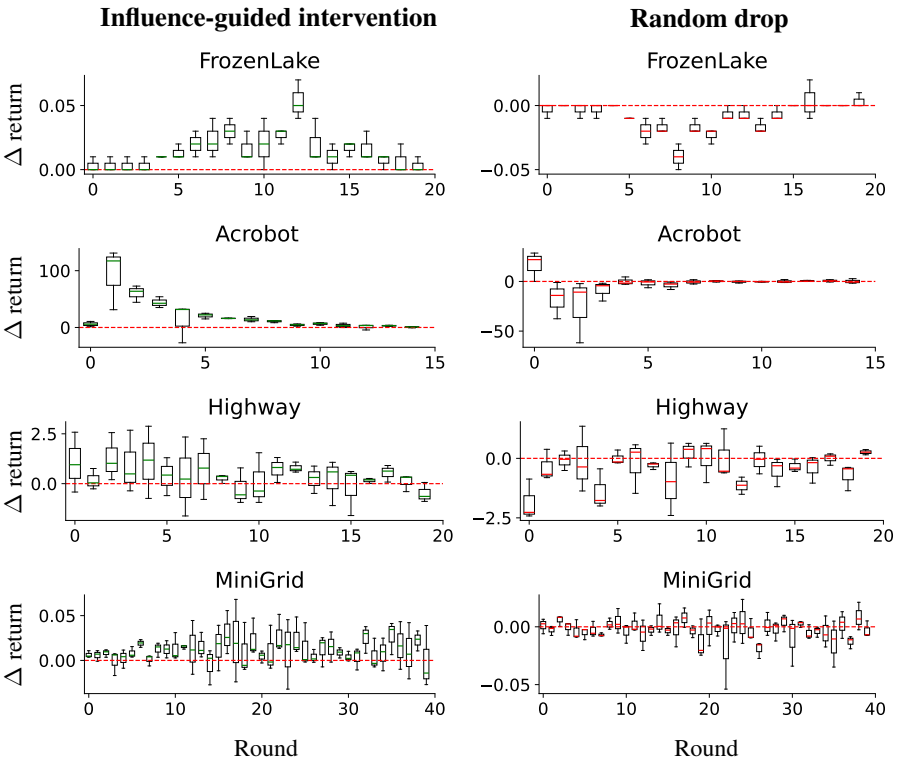

Figure 10: **Boxplots of Δ return for single rollout interventions in four environments, comparing influence-guided intervention (left) with random drop (right)**. We perform intervention for each iteration *independently* by removing bottom records and then retrain the model. The Δ return is calculated as the difference between the return from the model trained on the *filtered* dataset and the *original* dataset. Results are shown for three random seeds.

C.4 Advantage-based heuristic

Method. Sec. 4.1 characterizes the properties of the bottom harmful records—*sign mismatch* and *large magnitude errors*. Inspired by these findings, we design the following two heuristics for experience filtering:

- Heuristic 1: We discard records with opposite signs for $\bar{A}$ and $\hat{A}$. Among these records, we sort them by $|\bar{A} - \hat{A}|$ and discard the top $p\%$ records with the largest error.
- Heuristic 2: We discard records with opposite signs for $\bar{A}$ and $\hat{A}$. Among these records, we sort them by $\bar{A} \cdot \hat{A}$ and discard the bottom $p\%$ records with the smallest product (i.e., the most negative).

Implementation. These heuristics fundamentally rely on obtaining a reliable estimate of the true advantage function, $\bar{A}^{\pi}(s, a)$, for each training record. We obtain $\bar{A}$ using Monte Carlo (MC) estimates, i.e.,

$$\bar{A}^{\pi}(s,a) = \bar{Q}^{\pi}(s,a) - \bar{V}^{\pi}(s) = \mathbb{E}\left[\sum_k \gamma^k r_{t+k} | s_t = s, a_t = a\right] - \mathbb{E}\left[\sum_k \gamma^k r_{t+k} | s_t = s\right],$$

In environments with small, discrete state and action spaces, we can leverage the collected rollout buffer $B^{(k)}$ to obtain the estimate $\bar{A}^{\pi_{\theta^{(k)}}}$, as $B^{(k)}$ itself would include multiple occurrences of (s, a) pairs or visits to state s, allowing for empirical averaging.

However, in environments with large discrete or contiunous state/action spaces, specific state-action pairs (s, a) are rarely encountered multiple times in $B^{(k)}$. Accurately estimating $\bar{A}^{\pi_{\theta^{(k)}}}(s, a)$ for each record in these more complex settings would require resetting the environment to the specific s

and then performing numerous independent rollouts under policy $\pi_{\theta^{(k)}}$. This procedure is generally computationally infeasible.

For consideration of computational efficiency, in our study below, we limit to environments with *discrete* state and action spaces, where we compute $\bar{A}$ using the collected rollout buffer $B^{(k)}$, instead of performing additional sampling in the environment.

Results. Fig. 11 compares the two advantage-based heuristics against IIF and standard training in `FrozenLake` and `MiniGrid`.

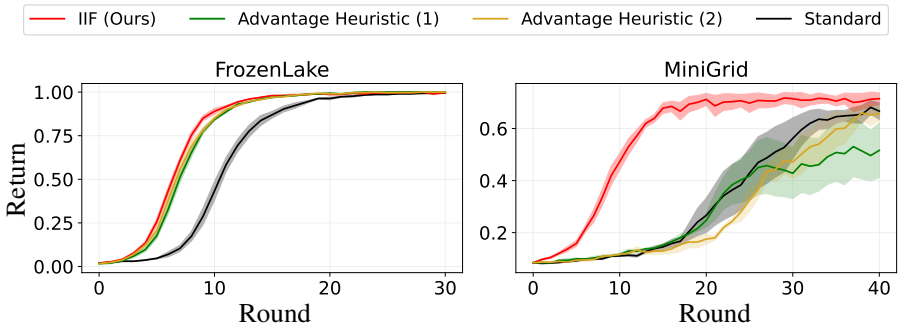

Figure 11: **Test returns over training rounds for the two advantage-based heuristics**, compared with IIF and standard PPO. Results are averaged over three random seeds.

In `FrozenLake`, a small discrete environment, both heuristics closely match IIF's learning curve and final return, and substantially outperforms standard PPO. This result serves as a validation of our initial findings in Section 4.1, confirming that transitions exhibiting sign mismatch or large advantage estimation errors are indeed key properties of harmful experiences, and that filtering based on these properties can significantly improve training efficiency.

However, in `MiniGrid`, which features a significantly larger state space, the advantage-based heuristics fail to improve upon the standard PPO baseline and in fact even degrade performance. There are two possible reasons. (1) The advantage estimates $\bar{A}$ are noisy due to the limited number of repeated visits per (s, a) and s in $B^{(k)}$, leading to inaccurate filtering. (2) These heuristics rely solely on the relationship between estimated and true advantages; in comparison, IIF's influence score, derived from gradients, captures a broader, more nuanced set of characteristics of harmful records. This richer representation allows IIF to perform effective filtering when simple advantage heuristics fail.

In summary, these results validate our core insights: properties like sign mismatch and large estimation errors are indeed indicative of harmful training records. At the same time, their failure in more complex environments highlights the limitations of these simple heuristics. Our IIF framework, by contrast, is more generally applicable; its influence scores capture a broader and more nuanced understanding of records' values beyond simple advantage discrepancies, enabling effective filtering even in complex domains.

C.5 TD error based heuristic

Motivation. Prioritized Experience Replay (PER) [Schaul et al., 2016] demonstrate that reweighting transitions in proportion to their temporal-difference (TD) error accelerates learning and improves performance in **off-policy** methods. TD error serves as a useful heuristic, indicating how "surprising" or "important" a transition is for updating the *value function*. While PPO is an on-policy method that typically uses a smaller, on-policy rollout buffer rather than a large replay buffer like those in off-policy algorithms, the core idea of focusing learning on more impactful experiences remains relevant. Inspired by PER, we investigate integrating a TD error based reweighting mechanism into the PPO training process to prioritize samples within its rollout buffer.

Implementation. For each transition (s_i, a_i, r_i, s_i') collected and stored in the rollout buffer $B^{(k)}$, we first compute its TD error. The TD error for record i is defined as:

$$\delta_i = r_i + \gamma V^{\pi_{\theta^{(k)}}}(s_i') - V^{\pi_{\theta^{(k)}}}(s_i),$$

where $V^{\pi_{\theta^{(k)}}}$ denotes the current value function estimate (under the current policy $\pi_{\theta^{(k)}}$).

We then assign a priority to each record using a rank-based approach following Schaul et al. [2016]. We sort all transitions in the buffer $B^{(k)}$ in descending order based on the absolute value of their TD error, $|\delta_i|$. The base priority for transition i is set as $P_i = 1/\text{rank}(i)$, where $\text{rank}(i)$ denotes the rank of transition i. Then, the probability of sampling record i is

$$w_i = \frac{P_i^\alpha}{\sum_{j \in B^{(k)}} P_j^\alpha}, \quad \text{where } \alpha = 0.6 \text{ (following Schaul et al. [2016])}$$

This weighting scheme ensures that transitions with larger absolute TD errors receive higher emphasis during the PPO optimization steps.

Results. We evaluate the performance of the TD error based reweighting heuristic by comparing it against our IIF and standard PPO on `FrozenLake` and `LunarLander`. Fig. 12 presents the test returns over training rounds for these approaches.

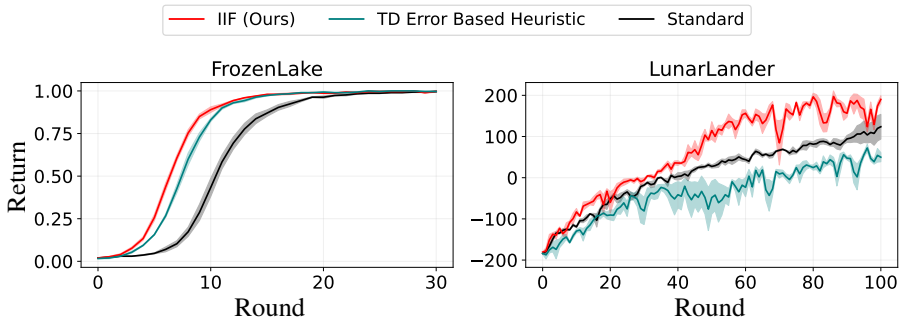

Figure 12: **Test returns over training rounds for the TD error based heuristic**, compared with IIF and standard PPO. Results are averaged over three random seeds.

In `FrozenLake`, a simple environment, both TD error and IIF accelerate convergence, reaching optimal return sooner. The TD error heuristic nearly matches IIF's speed, confirming that large TD errors align well with truly *useful* transitions when the state-action space is small and reward structure simple.

In contrast, in the more complex `LunarLander`, the TD error heuristic degrades performance: it learns more slowly than even standard PPO and exhibits greater variance. Although this heuristic succeeds in PER, we comment that there are intrinsic differences in the off-policy scenario where PER was proposed and evaluated, vs. the on-policy scenario (e.g., PPO) we study in this paper (Fig. 1). PER applies the TD error heuristic on a vast, diverse buffer. However, in PPO, raw TD errors mix estimator noise with true signal; PPO's small, fresh, on-policy batches exacerbate that noise; Our influence scores, in comparison, appears more robust in such scenarios.

C.6 IIF performance under various filtering percentages

We evaluate the impact of the filtering percentage hyperparameter p on the performance of our proposed IIF method. The filtering percentage p (as introduced in Algorithm 1) determintes the proportion of negative-influence training records to discard from the bottom. We explore a wide range of values for $p \in \{100.0\%, 50.0\%, 25.0\%, 12.5\%, 6.25\%\}$, reducing the percentage by half at each level. Note that $p = 100.0\%$ means discarding all negative-influence records.

Fig. 13 shows the test returns over training rounds for IIF with varying p's compared to baselines. We additionally quantify their efficiency using two metrics: SE_{ave} and SE_{peak} (introduced in Sec. 5.2). We summarize these efficiency statistics in Table 2.

We highlight several key findings:

- **Discarding all negative records ($p = 100\%$) is suboptimal.** As shown in Figure 13, setting $p = 100\%$ leads to suboptimal final performance, slower learning progress (also reflected in Table 2), and instability in training. This observation aligns with the concept of non-additivity of sample influence [Hu et al., 2024].

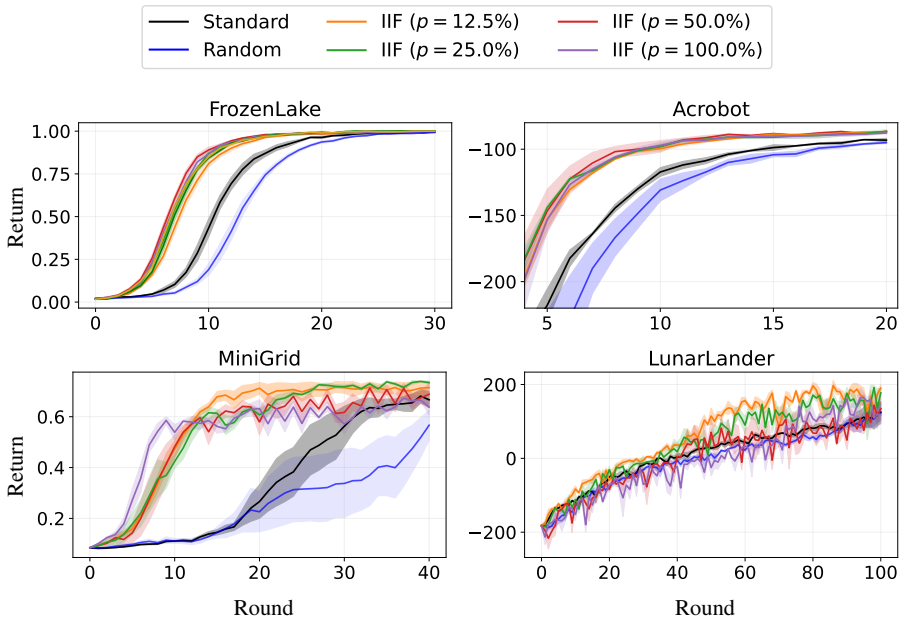

Figure 13: **Test returns over training rounds for IIF with a range of filtering percentages p**, compared to the baselines. Larger p means more aggressive filtering. Results are averaged over three random seeds.

Table 2: **Sample efficiency comparison across varying filtering percentages.** Results show the improvement in sample efficiency metrics (SE_{ave} and SE_{peak}) for different filtering percentages, across simpler and more complex environments. **Bold** values indicate the best performing value of p; *italicized* values show the second best. Results are averaged over three runs.

(a) SE_{ave} (↑)

	`FrozenLake`	`Acrobot`	`MiniGrid`	`LunarLander`
$p = 12.5\%$	23.5% ± 3.1%	29.2% ± 0.8%	*67.5%* ± 5.1%	**28.2%** ± 1.3%
$p = 25.0\%$	30.5% ± 3.3%	*35.1%* ± 0.6%	60.3% ± 10.6%	*22.7%* ± 5.6%
$p = 50.0\%$	**33.7%** ± 3.4%	**36.7%** ± 6.5%	67.0% ± 5.3%	10.2% ± 6.5%
$p = 100.0\%$	*32.7%* ± 1.7%	35.0% ± 0.5%	**75.4%** ± 3.6%	8.9% ± 2.0%

(b) SE_{peak} (↑)

	`FrozenLake`	`Acrobot`	`MiniGrid`	`LunarLander`
$p = 12.5\%$	15.6% ± 5.1%	31.5% ± 2.2%	**67.4%** ± 4.4%	**41.6%** ± 5.7%
$p = 25.0\%$	**22.1%** ± 7.4%	**48.5%** ± 0.8%	*58.8%* ± 13.1%	*32.9%* ± 13.1%
$p = 50.0\%$	*19.6%* ± 8.4%	**48.5%** ± 0.8%	50.6% ± 20.7%	15.5% ± 17.1%
$p = 100.0\%$	15.9% ± 5.5%	43.1% ± 5.7%	54.9% ± 22.5%	15.8% ± 7.3%

- **Any level of filtering improves performance over standard training.** Applying IIF with almost any filtering percentage demonstrates improvement compared to standard training. This underscores the general effectiveness of IIF in mitigating negative influence by removing a portion of identified negative samples.
- **The optimal filtering percentage varies with environment complexity.** In simpler environments (e.g. `FrozenLake`, `Acrobot`), removing half of the negative samples ($p = 50\%$) yields the best performance overall—simple environments could involve plenty of redundancy; aggressive pruning focuses learning on the most informative transitions. In contrast, in more complex environments (`MiniGrid`, `LunarLander`), the interplay among records is subtler: overly large filtering discard borderline-useful transitions, while a gentler filtering ($p = 12.5\%$) can achieve better performance.

Based on these findings, for our main experiments (see Sec. 5.2) we choose the specific filtering percentages to reflect the optimal configuration per environment. We use $p = 50\%$ for `FrozenLake`, `Acrobot`, `Highway`; $p = 12.5\%$ for `MiniGrid`, `LunarLander`; and $p = 6.25\%$ for `BipedalWalker`.

C.7 Runtime for experiments on traditional RL environments

We report the runtime for experiments on traditional RL environments in Table 3.

For **per-round runtime**, we report the time for the influence calculation step and the optimization step. The overhead of IIF in the influence calculation step is negligible. As IIF discards $p\%$ of the negative records, it enjoys a reduction in optimization time.

For **total runtime**, we first report the runtime for all training rounds (labeled as "All rounds"), and then report the runtime corresponding to the (reduced) rounds needed for IIF to match the peak performance of standard PPO (labeled as "Matching peak"). IIF's improvement in sample efficiency leads to a further speedup.

Finally, we report $\boldsymbol{RT}_{\text{peak}}$ (also presented in Fig. 6(b)), calculated as the reduced percentage of wall clock time for IIF to match standard PPO. In summary, IIF presents a 29%-67% reduction in runtime, effectively speeding up learning.

Table 3: **Per-round runtime and total runtime (in seconds), as well as the percentage of overall reduced runtime for experiments on traditional RL environments.** Results are averaged over 3 training runs each for IIF and standard training. A dash (—) indicates that a measure is not applicable.

		`FrozenLake`		`Acrobot`		`MiniGrid`	
		IIF	standard	IIF	standard	IIF	standard
Per-round runtime	Influence calc	0.11 ± 0.01	—	0.25 ± 0.01	—	0.25 ± 0.02	—
	Optimization	1.51 ± 0.04	2.01 ± 0.05	1.42 ± 0.02	2.02 ± 0.02	4.52 ± 0.06	5.02 ± 0.07
Total runtime	All rounds	82.15 ± 2.93	93.85 ± 2.68	70.01 ± 0.72	79.87 ± 1.00	365.23 ± 3.11	378.41 ± 2.98
	Matching peak	64.64 ± 3.98	—	35.80 ± 0.79	—	107.43 ± 3.32	—
$\boldsymbol{RT}_{\text{peak}}$ **(reduced runtime %) (↑)**		31.27% ± 3.28%		55.16% ± 1.04%		71.59% ± 1.05%	

		`Highway`		`LunarLander`		`BipedalWalker`	
		IIF	standard	IIF	standard	IIF	standard
Per-round runtime	Influence calc	0.13 ± 0.02	—	0.13 ± 0.01	—	0.12 ± 0.01	—
	Optimization	2.39 ± 0.48	3.29 ± 0.59	1.85 ± 0.04	2.05 ± 0.01	3.09 ± 0.20	3.30 ± 0.23
Total runtime	All rounds	214.41 ± 0.22	233.66 ± 0.24	318.68 ± 1.27	328.79 ± 3.65	676.78 ± 4.71	691.28 ± 13.33
	Matching peak	93.73 ± 1.69	—	183.64 ± 6.69	—	489.55 ± 4.71	—
$\boldsymbol{RT}_{\text{peak}}$ **(reduced runtime %) (↑)**		59.89% ± 0.72%		44.11% ± 2.29%		29.16% ± 0.66%	

C.8 Comparing two target functions for RLHF

In the main text (Sec. 5.3), we introduced two target functions for RLHF: the standard one f^{return}, and an adapted sequence-level objective f^{seq}. Here we show the comparison of the two in Fig. 14.

Overall, from both the training and testing curves, IIF with f^{seq} clearly outperforms the others. Although IIF with f^{return} initiallly improves faster than standard PPO, it soon plateaus, eventually converging to the same levels as the standard PPO baseline. This highlights that, the adapted sequence-level objective is more effective in RLHF's trajectory-centric setting with dual reward signals.

C.9 A breakdown of runtime for the RLHF experiments

Table 4 breaks down the wall-clock time (in seconds) for each component of one RLHF training round, under standard PPO and our IIF. The overhead of influence calculation in IIF is significantly offset by reduced optimization time, leading to a $2\times$ speedup *per round*.

Beyond this per-round saving, IIF requires fewer rounds to achieve comparable performance with standard PPO (requiring 32.75% ± 1.52% of training rounds, taking up 16.82% ± 1.32% of runtime combined with per-round speedup). Furthermore, IIF reaches convergence to a higher reward faster

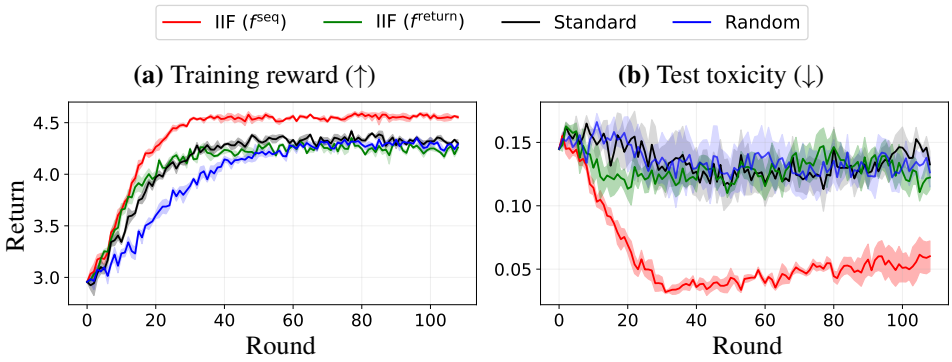

Figure 14: **Comparing two target functions f^{seq} with f^{return} for RLHF**. Results are averaged over 3 random seeds.

as well (requiring 48.51% $\pm$ 2.44% of training rounds, taking up 24.90% $\pm$ 0.80% of wall-clock time). This marks a $4\times$ overall speedup plus performance improvement compared to standard PPO.

Table 4: **Per-round runtime (in seconds) for RLHF with IIF vs. standard PPO.** IIF halves optimization time by pruning $\sim 50\%$ of the data each round, while the overhead of influence calculation is negligible. Reported results are averaged over all 109 training rounds in 3 training runs (using 3 random seeds). A dash (—) indicates that a measure is not applicable.

	IIF	Standard PPO	%
Response generation & scoring	1.71 $\pm$ 0.06	1.59 $\pm$ 0.05	
Forward	1.03 $\pm$ 0.04	0.99 $\pm$ 0.00	
Influence calculation	2.15 $\pm$ 0.02	—	
Optimization	40.39 $\pm$ 0.35	85.56 $\pm$ 0.17	
Total per-round runtime	45.28 $\pm$ 0.47	88.15 $\pm$ 0.22	51.37%

D Compute resources

All experiments were conducted on two Linux servers:

- **Machine 1**: Dual Intel Xeon Silver 4314 CPUs (16 cores/socket, 64 threads total), 251 GiB RAM, 4 NVIDIA RTX A6000 GPUs (48 GiB VRAM each).
- **Machine 2**: Dual AMD EPYC 7J13 CPUs (64 cores/socket, 256 threads total), 2 TiB RAM, 4 NVIDIA A100-SXM4-80GB GPUs (80 GiB VRAM each).

For experiments on standard RL benchmarks, we use both Machine 1 and 2; for experiments on RLHF, we use Machine 2 only.

All runtime results reported in Appendix C.7 were measured on Machine 1; all runtime results in Appendix C.9 were measured on Machine 2.

References

O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

K. Asadi, R. Fakoor, and S. Sabach. Resetting the optimizer in deep rl: An empirical study. *Advances in Neural Information Processing Systems*, 36:72284–72324, 2023.

A. Atrey, K. Clary, and D. Jensen. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkl3m1BFDB.

C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

S. Black, G. Leo, P. Wang, C. Leahy, and S. Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, Mar. 2021. URL https://doi.org/10.5281/zenodo.5297715. If you use this software, please cite it using these metadata.

T. A. Chang, D. Rajagopal, T. Bolukbasi, L. Dixon, and I. Tenney. Scalable influence and fact tracing for large language model pretraining. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=gLa96FlWwn.

Z. Cheng, X. Wu, J. Yu, W. Sun, W. Guo, and X. Xing. Statemask: Explaining deep reinforcement learning through state mask. *Advances in Neural Information Processing Systems*, 36:62457–62487, 2023.

Z. Cheng, X. Wu, J. Yu, S. Yang, G. Wang, and X. Xing. RICE: Breaking through the training bottlenecks of reinforcement learning with explanation. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=PKJqsZD5nQ.

Z. Cheng, J. Yu, and X. Xing. A survey on explainable deep reinforcement learning. *arXiv preprint arXiv:2502.06869*, 2025.

M. Chevalier-Boisvert, B. Dai, M. Towers, R. de Lazcano, L. Willems, S. Lahlou, S. Pal, P. S. Castro, and J. Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

F. R. Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

N. Das, S. Chakraborty, A. Pacchiano, and S. R. Chowdhury. Active preference optimization for sample efficient RLHF. In *ICML 2024 Workshop on Theoretical Foundations of Foundation Models*, 2024. URL https://openreview.net/forum?id=uSCvfYNn0s.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

C. Demircan, T. Saanum, A. K. Jagadish, M. Binz, and E. Schulz. Sparse autoencoders reveal temporal difference learning in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=2tIyA5cri8.

S. V. Deshmukh, A. Dasgupta, B. Krishnamurthy, N. Jiang, C. Agarwal, G. Theocharous, and J. Subramanian. Explaining RL decisions with trajectories. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=5Egggz1q575.

G. Dulac-Arnold, D. Mankowitz, and T. Hester. Challenges of real-world reinforcement learning, 2019. URL https://openreview.net/forum?id=S1xtR52NjN.

S. Gehman, S. Gururangan, M. Sap, Y. Choi, and N. A. Smith. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. *arXiv preprint arXiv:2009.11462*, 2020.

A. Ghorbani and J. Zou. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pages 2242–2251. PMLR, 2019.

S. Greydanus, A. Koul, J. Dodge, and A. Fern. Visualizing and understanding atari agents. In *International conference on machine learning*, pages 1792–1801. PMLR, 2018.

R. Grosse, J. Bae, C. Anil, N. Elhage, A. Tamkin, A. Tajdini, B. Steiner, D. Li, E. Durmus, E. Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

W. Guo, X. Wu, U. Khan, and X. Xing. Edge: Explaining deep reinforcement learning policies. *Advances in Neural Information Processing Systems*, 34:12222–12236, 2021.

Z. Hammoudeh and D. Lowd. Training data influence analysis and estimation: A survey. *Machine Learning*, 113(5):2351–2403, 2024.

S. Hara, A. Nitanda, and T. Maehara. Data cleansing for models trained with sgd. *Advances in Neural Information Processing Systems*, 32, 2019.

P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Y. Hu, P. Hu, H. Zhao, and J. Ma. Most influential subset selection: Challenges, promises, and beyond. *Advances in Neural Information Processing Systems*, 37:119778–119810, 2024.

Hugging Face. Detoxifying a language model using ppo. https://huggingface.co/docs/trl/en/detoxifying_a_lm, 2023. TRL documentation (v0.17.0), accessed May 8, 2025.

A. Ilyas, K. Georgiev, L. Engstrom, S. M. Park, and A. Madry. Data Attribution at Scale. Tutorial presented at the International Conference on Machine Learning (ICML), July 2024. Vienna, Austria. Official ICML page: https://icml.cc/virtual/2024/tutorial/35228. Materials available at https://ml-data-tutorial.org/.

R. Iyer, Y. Li, H. Li, M. Lewis, R. Sundar, and K. Sycara. Transparency and explanation in deep reinforcement learning neural networks. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 144–150, 2018.

A. Jacq, J. Ferret, O. Pietquin, and M. Geist. Lazy-mdps: Towards interpretable rl by learning when to act. In *Proceedings of the International Foundation for Autonomous Agents and Multiagent Systems*, pages 669–677, 2022.

Z. Juozapaitis, A. Koul, A. Fern, M. Erwig, and F. Doshi-Velez. Explainable reinforcement learning via reward decomposition. In *IJCAI/ECAI Workshop on explainable artificial intelligence*, 2019.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *The Third International Conference on Learning Representations*, 2015.

P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

E. Leurent. An environment for autonomous driving decision-making. https://github.com/eleurent/highway-env, 2018.

X. Li, H. Zou, and P. Liu. Limr: Less is more for rl scaling. *arXiv preprint arXiv:2502.11886*, 2025.

H. Lin, J. Long, Z. Xu, and W. Zhao. Token-wise influential training data retrieval for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 841–860, 2024.

H. Liu, M. Zhuge, B. Li, Y. Wang, F. Faccio, B. Ghanem, and J. Schmidhuber. Learning to identify critical states for reinforcement learning from videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1955–1965, 2023.

S. Liu and M. Zhu. UTILITY: Utilizing explainable reinforcement learning to improve reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Tk1VQDadfL.

S. Milani, N. Topin, M. Veloso, and F. Fang. Explainable reinforcement learning: A survey and comparative review. *ACM Computing Surveys*, 56(7):1–36, 2024.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PmLR, 2016.

A. Mott, D. Zoran, M. Chrzanowski, D. Wierstra, and D. Jimenez Rezende. Towards interpretable reinforcement learning using attention augmented agents. *Advances in neural information processing systems*, 32, 2019.

W. Muldrew, P. Hayes, M. Zhang, and D. Barber. Active preference learning for large language models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=CTgEV6qgUy.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

S. M. Park, K. Georgiev, A. Ilyas, G. Leclerc, and A. Madry. Trak: Attributing model behavior at scale. In *International Conference on Machine Learning*, pages 27074–27113. PMLR, 2023.

G. Pruthi, F. Liu, S. Kale, and M. Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930, 2020.

N. Puri, S. Verma, P. Gupta, D. Kayastha, S. Deshmukh, B. Krishnamurthy, and S. Singh. Explain your move: Understanding agent actions using specific and relevant feature attribution. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgzLkBKPB.

A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22 (268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

R. Rishav, S. Nath, V. Michalski, and S. E. Kahou. Behaviour discovery and attribution for explainable reinforcement learning. *arXiv preprint arXiv:2503.14973*, 2025.

A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani. Deep reinforcement learning framework for autonomous driving. *arXiv preprint arXiv:1704.02532*, 2017.

T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. In *International Conference on Learning Representations (ICLR)*, 2016. URL http://arxiv.org/abs/1511.05952.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Y. Shen, H. Sun, and J.-F. Ton. Reviving the classics: Active reward modeling in large language model alignment. *arXiv preprint arXiv:2502.04354*, 2025.

T. Shi, Y. Wu, L. Song, T. Zhou, and J. Zhao. Efficient reinforcement finetuning via adaptive curriculum learning. *arXiv preprint arXiv:2504.05520*, 2025.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

E. Soares, P. P. Angelov, B. Costa, M. P. G. Castro, S. Nageshrao, and D. Filev. Explaining deep learning models through rule-based approximation and visualization. *IEEE Transactions on Fuzzy Systems*, 29(8):2399–2407, 2020.

C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

N. Topin, S. Milani, F. Fang, and M. Veloso. Iterative bounding mdps: Learning interpretable policies via non-interpretable methods. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9923–9931, 2021.

M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. D. Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, H. Tan, and O. G. Younis. Gymnasium: A standard interface for reinforcement learning environments, 2024. URL https://arxiv.org/abs/2407.17032.

A. Verma, V. Murali, R. Singh, P. Kohli, and S. Chaudhuri. Programmatically interpretable reinforcement learning. In *International conference on machine learning*, pages 5045–5054. PMLR, 2018.

B. Vidgen, T. Thrush, Z. Waseem, and D. Kiela. Learning from the worst: Dynamically generated datasets to improve online hate detection. In *ACL*, 2021.

U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17:395–416, 2007.

L. von Werra, Y. Belkada, L. Tunstall, E. Beeching, T. Thrush, N. Lambert, S. Huang, K. Rasul, and Q. Gallouédec. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

H. Wang, Z. Wu, and J. He. Fairif: Boosting fairness in deep learning via influence functions with validation set sensitive attributes. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 721–730, 2024.

J. T. Wang, P. Mittal, D. Song, and R. Jia. Data shapley in one training run. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=HD6bWcj87Y.

J. T. Wang, D. Song, J. Zou, P. Mittal, and R. Jia. Capturing the temporal dependence of training data influence. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=uHLgDEgiS5.

S.-Y. Wang, A. A. Efros, J.-Y. Zhu, and R. Zhang. Evaluating data attribution for text-to-image models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7192–7203, 2023.

Y. Wang, Q. Yang, Z. Zeng, L. Ren, L. Liu, B. Peng, H. Cheng, X. He, K. Wang, J. Gao, W. Chen, S. Wang, S. S. Du, and Y. Shen. Reinforcement learning for reasoning in large language models with one training example. *arXiv preprint arxiv:2504.20571*, 2025c.

M. Xia, S. Malladi, S. Gururangan, S. Arora, and D. Chen. LESS: Selecting influential data for targeted instruction tuning. In *ICLR 2024 Workshop on Navigating and Addressing Data Problems for Foundation Models*, 2024. URL https://openreview.net/forum?id=Kw3ckB2Kfc.

T. Xie, H. Li, A. Bai, and C.-J. Hsieh. Data attribution for diffusion models: Timestep-induced bias in influence estimation. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=P3Lyun7CZs.

Y. E. Xu, Y. Savani, F. Fang, and Z. Kolter. Not all rollouts are useful: Down-sampling rollouts in llm reinforcement learning. *arXiv preprint arXiv:2504.13818*, 2025.

J. Yu, W. Guo, Q. Qin, G. Wang, T. Wang, and X. Xing. {AIRS}: Explanation for deep reinforcement learning based security applications. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 7375–7392, 2023.

Q. Yu, Z. Zhang, R. Zhu, Y. Yuan, X. Zuo, Y. Yue, T. Fan, G. Liu, L. Liu, X. Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Y. Yu. Towards sample efficient reinforcement learning. In *IJCAI*, pages 5739–5743, 2018.

T. Zahavy, N. Ben-Zrihem, and S. Mannor. Graying the black box: Understanding dqns. In *International conference on machine learning*, pages 1899–1908. PMLR, 2016.

R. Zhao, D. Morwani, D. Brandfonbrener, N. Vyas, and S. M. Kakade. Deconstructing what makes a good optimizer for autoregressive language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=zfeso8ceqr.

X. Zheng, T. Pang, C. Du, J. Jiang, and M. Lin. Intriguing properties of data attribution on diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vKViCoKGcB.