PREDICTING IRREVERSIBLE TRANSITIONS FOR SAFE SWITCHING IN AUTONOMOUS RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Autonomous reinforcement learning (ARL) aims to reduce manual resets by alternating between forward and reset policies. One of the main challenges in ARL is handling irreversible states, from which the agent cannot recover without external intervention. While several recent works have proposed ARL algorithms that can handle irreversible states, these algorithms require task-specific knowledge, such as privileged state information or reversibility labels. Furthermore, they pay little attention to safely switching between forward and reset policies. In this paper, we propose a novel ARL algorithm that identifies irreversible states from image observations without task-specific knowledge and enables safe policy switching. Our algorithm introduces an irreversibility estimator that takes the current image observation and action as input and estimates the probability of transitioning to an irreversible state. We train the estimator on the state-action sequences that lead to irreversible states, so that it predicts an irreversible transition several steps before it occurs and aborts the ongoing policy early. The supervisory signals for these sequences are extracted from rollouts using vision-language models. Our algorithm also introduces a transition policy that guides the agent to safe switching states. These states are reversible, not too difficult for resuming the forward policy, and reachable from the aborted state. Experimental results demonstrate that our algorithm outperforms prior ARL algorithms across diverse navigation and manipulation tasks, even without task-specific knowledge.

1 INTRODUCTION

Autonomous reinforcement learning (ARL) reduces manual resets by alternating between a forward policy, which learns to reach goal states, and a reset policy, which returns the agent to initial states (Eysenbach et al., 2017; Zhu et al., 2020; Sharma et al., 2021; 2022; Xie et al., 2022; Patil et al., 2024; Lee & Seo, 2024; Lee, 2025). Manual resets are easily performed in simulation, but they require a significant amount of time and effort in the real world (Levine et al., 2018; Zhu et al., 2019; Sharma et al., 2020; Zeng et al., 2020). The real world often involves irreversible states from which the agent cannot recover without external intervention, such as a robot falling over or an object being pushed outside the workspace. These states vary across environments and tasks, and they are difficult to identify in general. Handling irreversible states therefore still remains an open problem in ARL.

Several recent works have proposed ARL algorithms that can detect and avoid irreversible states (Eysenbach et al., 2017; Xie et al., 2022). These algorithms have shown that the agent can learn in irreversible environments with fewer manual resets. However, they rely on task-specific knowledge, such as privileged state information or reversibility labels, to identify irreversible states. Privileged state information requires additional sensor instrumentation that is often unavailable in the real world, and reversibility labels involve manual annotation of whether a state can be recovered without external intervention. Such task-specific requirements limit the scalability of these ARL algorithms to real-world environments. Furthermore, these algorithms rarely consider safety when switching between the forward and reset policies. Their switching states can be dangerously close to irreversible states, where the agent may not know how to recover and can enter an irreversible state with only a few actions.

In this work, we propose a new ARL algorithm that identifies irreversible states from image observations and safely switches between the forward and reset policies in irreversible environments. We do

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

not assume privileged information or reversibility labels to identify irreversible states. Our algorithm introduces two new components: an irreversibility estimator and a transition policy. The irreversibility estimator takes the current observation and action as input and predicts whether the agent will enter irreversible states multiple steps in advance, enabling agents to keep a safe distance from them. This contrasts with previous works that typically detect irreversible states only at the preceding state. To detect irreversible states multiple steps ahead, we extract the sequence of reversibility labels from image-based rollouts by leveraging vision-language models. These label sequences serve as the supervisory signals for training the irreversibility estimator. We hypothesize that states are safe to switch if agents will not enter irreversible states with only a few actions and know how to resume the next policy. The transition policy guides the agent to safe switching states from the final state of the previous policy. Since the transition policy is trained along with the forward and reset policies, we also consider whether the switching states are reachable by using the transition policy from that state when identifying the safe switching states.

The main contribution of this work is an ARL algorithm that enables an agent to detect irreversible states multiple steps ahead from image observations and to discover safe states to switch between the forward and reset policies. We evaluate our algorithm against prior ARL algorithms on four navigation and manipulation tasks with irreversible states. We observed that the irreversibility estimator predicts entering irreversible states several steps before the agent reaches them, and that both the estimator and the transition policy reduce the number of manual resets. Experimental results demonstrate that our algorithm achieves comparable or higher success rates than the baselines while requiring fewer manual resets.

2 Related Work

Prior ARL algorithms determine which initial states the reset policy returns the agent to. MEDAL (Sharma et al., 2022) matches the state distribution of expert demonstrations, and VaPRL (Sharma et al., 2021) and SCG (Lee & Seo, 2024) generate a curriculum that selects initial states according to the learning progress of the agent. These algorithms showed that the agent learns diverse tasks with fewer manual resets. However, they assume that the agent has access to the state information required for learning. Such privileged state information requires additional sensor instrumentation, which is often unavailable in the real world. To learn without such instrumentation, R3L (Zhu et al., 2020) learns from image observations alone in the real world and discovers novel initial states with a perturbation controller. All of these algorithms assume that the environment is reversible, whereas real-world environments often involve irreversible states.

Several ARL algorithms detect and avoid irreversible states. Eysenbach et al. (2017) proposed LNT that early aborts the forward policy based on the value function of the reset policy, which is trained with a predefined reset reward function. Xie et al. (2022) proposed PAINT that trains a reversibility classifier with human-provided labels and requests an intervention when the agent reaches an irreversible state. Their experimental results showed that both algorithms reduce manual resets in irreversible environments. However, a reset reward function must be designed for each task, and reversibility labels must be annotated by a human throughout training. Such task-specific knowledge is rarely available in the real world, which limits the scalability of these algorithms. Our algorithm instead queries vision-language models about the image observations collected in each rollout. Their commonsense reasoning is not specific to any environment or task, and our algorithm therefore identifies irreversible states without task-specific knowledge.

Beyond ARL, the problem of switching between consecutive policies has also been addressed in RL. Lee et al. (2019) proposed a transition policy that guides the agent to a state from which the subsequent policy can be successfully executed. Chen et al. (2023) learn a transition feasibility function that evaluates whether the subsequent policy can succeed from the current state, and use it to determine when to switch. Their results showed that the state at which policies are switched affects the performance of the subsequent policy. ARL alternates between the forward and reset policies, and the same problem arises at each switching. In irreversible environments, the aborted state may still be close to an irreversible state, and the forward policy may be difficult to resume from it. Our algorithm therefore introduces a transition policy that guides the agent to a state that is reversible, not too difficult for the forward policy to resume learning, and reachable from the aborted state. Table 1 summarizes the properties that distinguish our algorithm from prior ARL algorithms.

Table 1: Comparison of assumptions and capabilities across ARL algorithms.

	Assumptions		Capabilities		
	Predefined reset reward function	Human-provided reversibility labels	Irreversible environments	Multi-step prediction	Safe policy switching
LNT (Eysenbach et al., 2017)	✓	✗	✓	✗	✗
PAINT (Xie et al., 2022)	✗	✓	✓	✗	✗
R3L (Zhu et al., 2020)	✗	✗	✗	✗	✗
Ours	✗	✗	✓	✓	✓

3 PRELIMINARIES

3.1 GOAL-CONDITIONED REINFORCEMENT LEARNING

We formulate the learning problem as goal-conditioned reinforcement learning (GCRL). The state, action, and goal spaces are denoted by $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{G}$. The transition kernel $\mathcal{T}(s' \mid s, a)$ gives the probability of reaching state s' after taking action a in state s, and $r_g(s, a, s')$ is the reward for goal g. Initial states and goals follow the distributions ρ_0 and ρ_g, and γ is the discount factor. The mapping $m : \mathcal{S} \to \mathcal{G}$ assigns a goal representation to each state and is the identity mapping when the state and goal spaces coincide. A goal-conditioned policy $\pi_{\text{gc}}(a \mid s, g)$ produces an action distribution from the current state s and goal g. Its action-value function $Q(s, a, g)$ represents the expected discounted return after taking action a in state s and following π_{gc} toward g. The GCRL objective is to maximize the expected return over goals sampled from ρ_g under unknown environment dynamics (Kaelbling, 1993; Schaul et al., 2015; Andrychowicz et al., 2017). The forward, reset, and transition policies in Section 3.2 use this formulation.

3.2 POLICY AND REPLAY-BUFFER SETUP

We use a goal-conditioned policy $\pi_{\text{gc}}(a \mid s, g)$ and an exploration policy $\pi_e(a \mid s)$. The goal-conditioned policy is trained to reach a given goal, and the exploration policy is trained with random network distillation (RND) (Burda et al., 2019) to visit diverse states. The forward, reset, and transition policies are the same π_{gc} conditioned on different goals. The forward policy π_f is conditioned on a selected goal g, the reset policy π_r on a selected initial state s_{init}, and the transition policy π_t on a safe switching state s^*_{init}, as described in Section 4.2.

Transitions collected by π_{gc} are stored in the replay buffer B_{gc} and are also added to a second buffer B_{e}. Transitions collected by π_e are stored only in B_{e}. As a result, B_{e} contains states visited through both goal-directed behavior and RND-based exploration, providing a broad distribution of states. We use this distribution to train π_e and to sample candidate states for selecting informative initial states and safe switching states.

3.3 STATE INFORMATION ESTIMATOR

We use a state information estimator $I(s, g)$ (Lee, 2025; Lee & Seo, 2024). It takes a pair of states as input and estimates the probability of reaching g from s when the agent follows π_{gc}. Here we describe how our algorithm utilizes the state information estimator to generate a curriculum that selects the goal and the initial state at each iteration.

Let $\mathcal{G}_k^{\text{cand}}$ denote the candidate goals available at iteration k. Our algorithm determines the goal for the kth iteration as $g_k \sim \text{Unif}(\mathcal{G}_k^*)$, where

$$\mathcal{G}_k^* \triangleq \left\{ g \in \mathcal{G}_k^{\text{cand}} \;\middle|\; \lambda_{\text{low}} \leq \mathbb{E}_{s \sim B_{\text{e}}} \left[I(s, g) \right] \leq \lambda_{\text{high}} \right\}. \tag{1}$$

$\mathcal{G}_k^*$ is the set of informative goals for the kth iteration, and λ_{low} and λ_{high} are the lower and upper thresholds of the estimated probability. This prevents the agent from being assigned goals that are either too difficult or too easy.

After the goal g_k is determined, our algorithm samples a candidate-state set $\mathcal{C}_k$ from B_e and determines the initial state as $s_{\text{init}} \sim \text{Unif}(\mathcal{S}^*_{\text{init},k})$, where

$$\mathcal{S}^*_{\text{init},k} \triangleq \{s \in \mathcal{C}_k \mid \lambda_{\text{low}} \leq I(s, g_k) \leq \lambda_{\text{high}}\}. \tag{2}$$

This provides the agent with initial states that match its current capability of reaching g_k.

We extract outcome labels e_t from the rollouts collected by π_{gc} and use them to train the state information estimator in a self-supervised manner. State-goal pairs from the rollouts in which the agent reaches the goal are treated as positive examples, and those from failed rollouts are treated as negative examples. The training objective is

$$\min_I -\mathbb{E}_{(s_t,g_t,e_t)\sim B_{\text{gc}}} \left[e_t \log I(s_t, g_t) + (1 - e_t) \log(1 - I(s_t, g_t))\right]. \tag{3}$$

The estimator is updated as π_{gc} improves, and the selected goals and initial states therefore follow the learning progress of the agent.

4 ARL via Safe Switching in Irreversible Environments

Our algorithm is designed to enable an agent to learn in irreversible environments with fewer manual resets. Similar to previous ARL algorithms, our algorithm alternates between the forward policy π_f and the reset policy π_r. Unlike previous ARL algorithms, our algorithm does not switch policies at the state where the previous policy ends. At the beginning of each iteration, our algorithm selects an informative goal g_k and an informative initial state s_{init} with the state information estimator I described in Section 3.3. The agent then follows π_r toward s_{init}, explores with π_e, and follows π_f toward g_k. Throughout an iteration, the irreversibility estimator evaluates the risk of entering an irreversible state at each step. If the risk exceeds a threshold, the ongoing policy is early aborted and the transition policy π_t guides the agent to a safe switching state before training resumes. In the remainder of this section, we describe the irreversibility estimator in Section 4.1 and the transition policy in Section 4.2.

4.1 Irreversibility Estimator

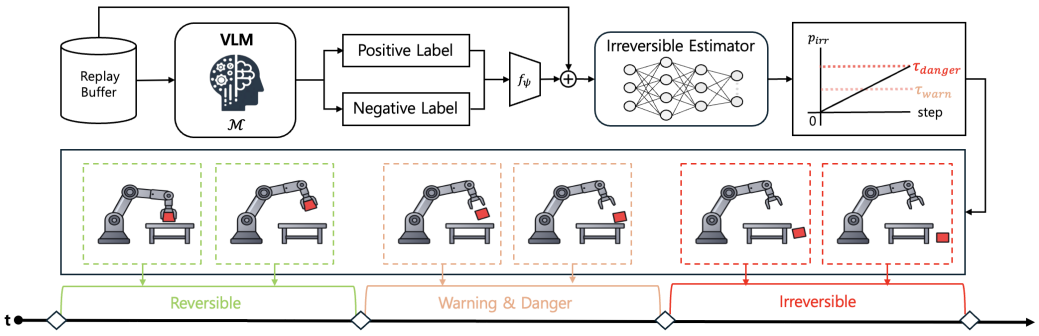


Figure 1: Overview of irreversibility labeling and training of the irreversibility estimator. The VLM $\mathcal{M}$ labels the image observations in each rollout as reversible or irreversible, using binary search to identify the first irreversible state. The irreversibility estimator E_ϕ is trained on these labels, taking the image observation encoded by f_ψ and the action as input.

The irreversibility estimator E_ϕ takes the image observation o_t and the action a_t as input. It estimates the probability that the agent enters an irreversible state after taking a_t at o_t:

$$p_{\text{irr}}(o_t, a_t) = E_\phi(f_\psi(o_t), a_t), \tag{4}$$

where f_ψ is an image encoder. Conditioning the estimator on the action allows it to distinguish the actions that lead the agent toward an irreversible state from the other actions at the same observation.

An agent reaches an irreversible state through a sequence of transitions rather than a single transition. A label placed at the transition into an irreversible state provides no signal until the agent is already adjacent to that state. At that point the agent has little opportunity to move away from it. We therefore assign the training target $y_t = 1$ to the first irreversible transition and to the N transitions preceding it. The estimator then predicts which action at the current observation leads the agent into an irreversible state within the next N steps. An early abort is thus triggered while the agent can still move away from an irreversible state.

The labels are obtained without a predefined reset reward function or human-provided reversibility labels. Irreversible states vary across environments and tasks, whereas the commonsense reasoning of vision-language models (VLMs) is not specific to any environment or task. After the agent takes a_t at o_t, our algorithm provides the next image observation o_{t+1} to a VLM $\mathcal{M}$ and obtains a binary reversibility label

$$\mathcal{M}(o_{t+1}) \in \{0, 1\}, \tag{5}$$

where a label of 1 indicates that o_{t+1} is an irreversible state and a label of 0 indicates a reversible state. Appendices B.2–D provide the details, comparisons, and ablations for the VLM.

Labeling every image observation in a rollout of length T requires $O(T)$ queries. All states following an irreversible state are irreversible, and all states preceding it are reversible (Xie et al., 2022). We exploit this property to locate the first irreversible transition with binary search, which reduces the number of queries to $O(\log T)$.

The labeled transitions are stored in B_e. We periodically sample them from B_e and update both f_ψ and E_ϕ with the binary cross-entropy loss

$$\mathcal{L}_\mathrm{irr} = -y_t \log p_\mathrm{irr}(o_t, a_t) - (1 - y_t) \log \left(1 - p_\mathrm{irr}(o_t, a_t)\right). \tag{6}$$

Since the labels are expanded over the N preceding transitions, p_irr increases as the agent approaches an irreversible state. Our algorithm evaluates p_irr at every step of the reset, exploration, and forward policies, and distinguishes two levels of risk with the thresholds τ_warning and τ_danger, where $\tau_\mathrm{danger} > \tau_\mathrm{warning}$. A warning prediction, $\tau_\mathrm{warning} \leq p_\mathrm{irr} < \tau_\mathrm{danger}$, indicates that the agent is approaching an irreversible state but can still avoid entering it. A danger prediction, $p_\mathrm{irr} \geq \tau_\mathrm{danger}$, indicates that the agent is about to enter one. Both trigger an early abort of the ongoing policy, and a danger prediction incurs a larger penalty.

4.2 Transition Policy

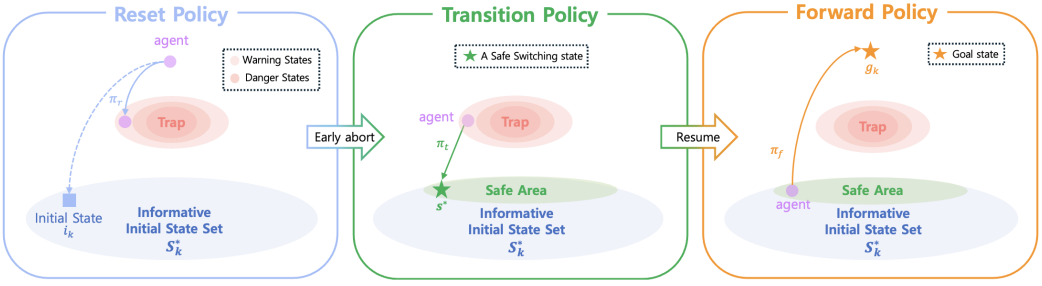


Figure 2: Overview of safe switching with the transition policy. After an early abort, the transition policy π_t guides the agent from the aborted state s_t to a safe switching state s^*_init, and training resumes at that state.

The transition policy guides the agent to a safe switching state, which is reversible, not too difficult for the forward policy to resume learning toward the current goal g_k, and reachable from the aborted state s_t. Here we describe how our algorithm identifies such a state. Our algorithm samples the candidate set $\mathcal{C}_k$ from states in B_e that the VLM labels as reversible and the irreversibility estimator identifies as low-risk. This excludes the irreversible states together with the states within N steps of them. Among these candidates, our algorithm identifies the states from which the agent can reach the

current goal as

$$\mathcal{S}_k^* \triangleq \{s \in \mathcal{C}_k \mid \lambda_{\text{low}} \leq I(s, g_k) \leq \lambda_{\text{high}}\}. \tag{7}$$

$I(s, g_k)$ is the probability that the current π_{gc} reaches g_k from s, and λ_{low} and λ_{high} are the lower and upper thresholds of this probability. Since I is trained on the outcomes of the rollouts collected by π_{gc}, this probability reflects what the agent can currently achieve and changes as π_{gc} improves. This prevents training from resuming at a state where the current goal is either too difficult or too easy. The states in $\mathcal{S}_k^*$ are suitable for resuming training, but the agent is still at s_t. Our algorithm therefore selects the states it can reach from s_t, and the transition policy is the same π_{gc}, so the same estimator applies to this pair. The safe switching state is determined as $s_{\text{init}}^* \sim \text{Unif}(\mathcal{R}_k^*(s_t))$, where

$$\mathcal{R}_k^*(s_t) \triangleq \{s \in \mathcal{S}_k^* \mid \lambda_{\text{low}} \leq I(s_t, s) \leq \lambda_{\text{high}}\}. \tag{8}$$

$I(s_t, s)$ is the probability that the agent reaches a candidate s from s_t. This prevents our algorithm from selecting a candidate that the agent rarely reaches or one that lies within a few steps of s_t, which would leave the agent near an irreversible state again. The transition policy is executed as π_{gc} conditioned on s_{init}^*, and the transitions collected during its execution are stored in B_{gc} and B_{e} and used to update π_{gc} and I. Once the agent reaches s_{init}^*, training resumes at that state. The detailed transition-policy procedure is provided in Appendix A.

4.3 Algorithm Summary

Algorithm 1

Initialize policies $\pi_{\text{gc}}(a \mid s, g)$ and $\pi_e(a \mid s)$
Initialize irreversibility estimator $E_\phi(f_\psi(o_t), a_t)$ and VLM labeler $\mathcal{M}(o_{t+1})$
Initialize replay buffers B_{gc}, B_{e} and state information estimator $I(s, g)$
for $k \leftarrow 1$ to K **do**
  $g_k \leftarrow \text{get_informative_goal}\left(B_{\text{e}}, \mathcal{G}_k^{\text{cand}}, I\right)$
  $s_{\text{init}} \leftarrow \text{get_informative_initial_state}\left(B_{\text{e}}, g_k, I\right)$
  for $t \leftarrow 1$ to T_{reset} **do** ▷ Reset policy
    Observe o_t; sample $a_t \sim \pi_r(\cdot \mid s_t, s_{\text{init}})$; estimate $p_{\text{irr}}(o_t, a_t)$ with E_ϕ
    if $p_{\text{irr}}(o_t, a_t) \geq \tau_{\text{warning}}$ **then** execute π_t as described in Section 4.2; **break**
    Execute a_t and observe (s_{t+1}, o_{t+1}, r_t)
    Store transition in B_{gc} and B_{e}; update π_{gc}
  end for
  for $t \leftarrow 1$ to $T_{\text{exploration}}$ **do** ▷ Exploration policy
    Observe o_t; sample $a_t \sim \pi_e(\cdot \mid s_t)$; estimate $p_{\text{irr}}(o_t, a_t)$ with E_ϕ
    if $p_{\text{irr}}(o_t, a_t) \geq \tau_{\text{warning}}$ **then** execute π_t as described in Section 4.2; **break**
    Execute a_t and observe (s_{t+1}, o_{t+1}, r_t)
    Store transition in B_{e}; update π_e
  end for
  for $t \leftarrow 1$ to T_{forward} **do** ▷ Forward policy
    Observe o_t; sample $a_t \sim \pi_f(\cdot \mid s_t, g_k)$; estimate $p_{\text{irr}}(o_t, a_t)$ with E_ϕ
    if $p_{\text{irr}}(o_t, a_t) \geq \tau_{\text{warning}}$ **then break**
    Execute a_t and observe (s_{t+1}, o_{t+1}, r_t)
    Store transition in B_{gc} and B_{e}; update π_{gc}
  end for
  Update I from the trajectories in B_{gc}
  Label transitions in B_{e} using $\mathcal{M}$ and binary search
  Periodically update f_ψ and E_ϕ from the labeled transitions
end for

Algorithm 1 summarizes the overall training procedure. At the beginning of each iteration, our algorithm samples an informative goal g_k and an informative initial state s_{init} using the state information estimator described in Section 3.3. The agent follows π_r toward s_{init}, explores with π_e, and then follows π_f toward g_k. An early abort is triggered whenever $p_{\text{irr}} \geq \tau_{\text{warning}}$. After an early abort of the reset or exploration policy, our algorithm executes π_t toward a safe switching state s_{init}^* selected as described in Section 4.2, and training resumes at that state. After an early abort of the forward policy, the iteration ends without executing π_t. Our algorithm labels the collected transitions in B_{e} using $\mathcal{M}$ and binary search, and periodically samples the labeled transitions to update f_ψ and E_ϕ.

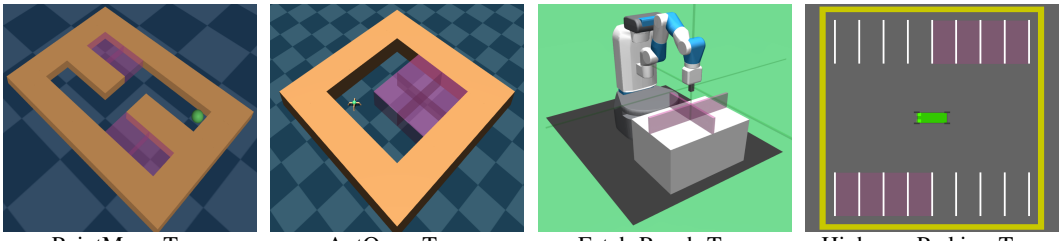
5 EXPERIMENTS

We design our experiments to investigate the following questions: (1) Can our algorithm achieve success rates comparable to or higher than those of prior ARL algorithms while requiring fewer manual resets in irreversible environments? (2) Can the irreversibility estimator predict entering irreversible states multiple steps in advance? (3) How do the irreversibility estimator and the transition policy each contribute to reducing manual resets?

To answer these questions, we compare our algorithm with six baselines: (1) Leave No Trace (LNT) (Eysenbach et al., 2017), (2) LNT-MG, (3) PAINT (Xie et al., 2022), (4) PAINT-MG, (5) R3L (Zhu et al., 2020), and (6) R3L-MG. LNT trains forward and reset policies and uses the reset value function to terminate the forward policy when recovery is predicted to be difficult. However, it requires a predefined reset reward function. PAINT learns a reversibility-aware value function and classifier from human-provided labels and terminates the current policy when the classifier predicts irreversibility. R3L uses random network distillation (RND) (Burda et al., 2019) to explore diverse initial states but does not explicitly identify irreversible transitions. Since these methods were originally developed for single-goal tasks, we additionally evaluate their multi-goal variants, denoted by "-MG." Each multi-goal variant periodically samples a goal from states stored in its replay buffer. Our algorithm predicts irreversibility risk from image observations and uses the transition policy to guide the agent from an aborted state to a safe switching state without a predefined reset reward function or human-provided reversibility labels.

5.1 ENVIRONMENTS

We evaluate the algorithms in four environments with irreversible states: PointMaze-Trap, AntOpen-Trap, Highway-Parking-Trap, and Fetch-Reach-Trap. In PointMaze-Trap and AntOpen-Trap, the purple areas indicate states from which the agent cannot recover. In Highway-Parking-Trap, the designated parking slots shown in purple indicate such states. In Fetch-Reach-Trap, the wall-shaped area on the tabletop indicates such states. We count a manual reset whenever external intervention is required after the agent enters one of these states.



PointMaze-Trap AntOpen-Trap Fetch-Reach-Trap Highway-Parking-Trap

Figure 3: Environments used in our experiments. Purple areas in PointMaze-Trap and AntOpen-Trap, purple parking slots in Highway-Parking-Trap, and the wall-shaped area in Fetch-Reach-Trap indicate irreversible states that require external intervention.

Overall performance. Figure 4 compares success rates and manual resets across four irreversible environments. The upper row shows that the single-goal variants of LNT, PAINT, and R3L generally achieve lower success rates than their multi-goal variants. Fixed goals limit the diversity of state-goal pairs, while periodically sampling goals from the replay buffer generally improves success. However, LNT-MG remains below the other multi-goal variants, and PAINT-MG achieves a lower success rate in AntOpen-Trap than in PointMaze-Trap or Highway-Parking-Trap. These results suggest that random goal diversity alone does not guarantee informative transitions.

Our algorithm instead uses the state information estimator to select goals and initial states according to their estimated reachability. It achieves success rates comparable to the strongest multi-goal baselines in PointMaze-Trap, AntOpen-Trap, and Fetch-Reach-Trap. In Highway-Parking-Trap, its final success rate is lower than those of PAINT-MG and R3L-MG. The warning and danger predictions in Figure 5 extend into nearby reversible parking slots, which may reduce opportunities to practice those goals.

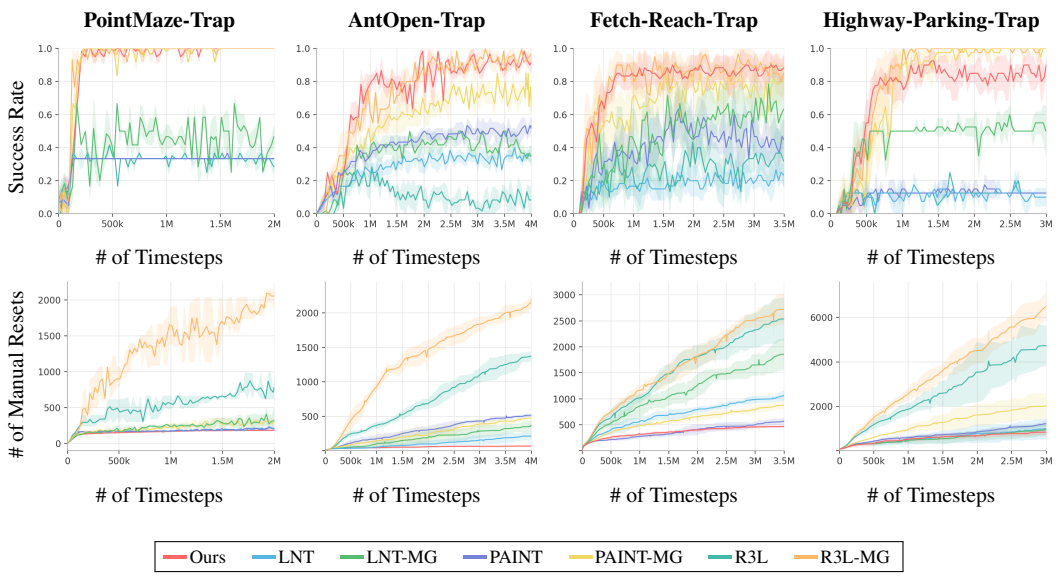


Figure 4: Success rates and manual resets across four irreversible environments. The upper and lower rows show success rates and the number of manual resets, respectively. Each curve reports the mean over five random seeds, and its shaded area indicates the corresponding standard deviation.

The lower row of Figure 4 shows the number of manual resets. The R3L variants require many manual resets because their RND-based exploration does not explicitly identify irreversible transitions. This is particularly evident for R3L-MG, which achieves high success rates in several environments but frequently enters irreversible states. LNT and LNT-MG use the reset value function to reduce manual resets, but rely on a predefined reset reward function and a narrow initial-state distribution, which is consistent with their lower success rates.

Even with environment-provided reversibility labels, PAINT-MG requires fewer manual resets than R3L-MG but more than our algorithm. PAINT terminates the current policy when its classifier predicts irreversibility and resumes training at the aborted state, which may lie close to an irreversible state. Our irreversibility estimator is trained on the N transitions preceding the first irreversible transition, so an early abort is triggered while the agent can still move away. The transition policy then guides the agent to a safe switching state before training resumes. This safe switching reduces manual resets without a predefined reset reward function or human-provided reversibility labels. Overall, our algorithm maintains competitive success rates while requiring fewer manual resets across the four environments.

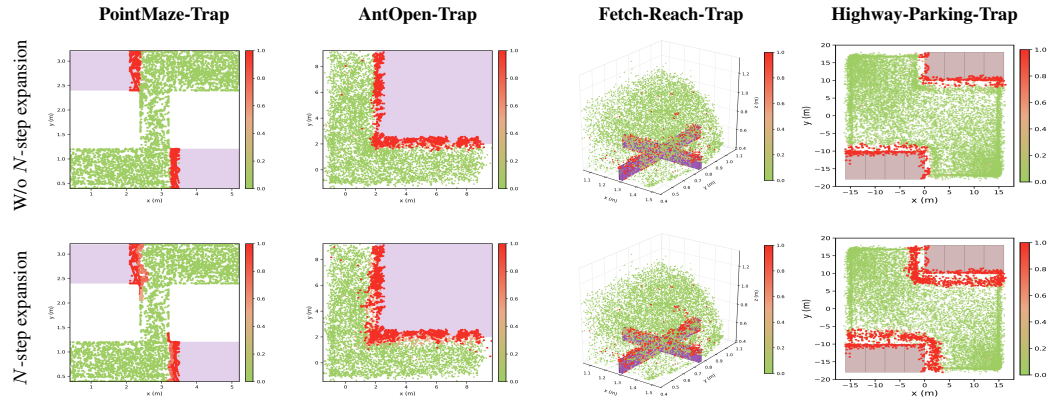


Figure 5: Effect of the N-step label expansion on irreversibility prediction. The upper row shows predictions without the expansion, and the lower row shows predictions with the expansion.

Irreversibility prediction. To examine whether the irreversibility estimator can identify risk before the agent enters an irreversible state, Figure 5 visualizes its predictions with and without the N-step label expansion in four environments. Each sampled point is colored by the predicted probability that the selected action leads the agent into an irreversible state. In the upper row, without the expansion, predicted risk increases only near irreversible states, making the estimator less able to identify risk in advance and trigger an early abort. In the lower row, the expansion causes elevated predictions to extend to nearby reversible states, allowing an early abort before the agent enters an irreversible state. In Highway-Parking-Trap, warning and danger predictions extend into adjacent reversible parking slots and may reduce opportunities to practice those goals.

Ablation study. We examine the contributions of the irreversibility estimator and the transition policy in PointMaze-Trap and Fetch-Reach-Trap. Ours w/o Estimator and Transition removes both components, whereas Ours w/o Transition retains the estimator and removes the transition policy. Figure 6 shows that Ours w/o Estimator and Transition requires the most manual resets in both environments. The gap between the two variants shows that the estimator reduces manual resets by identifying risk in advance and triggering an early abort before the agent enters an irreversible state. The full method further reduces manual resets by guiding the agent from the aborted state to a safe switching state before training resumes. The gap between the full method and Ours w/o Transition shows that the transition policy also contributes to reducing manual resets.

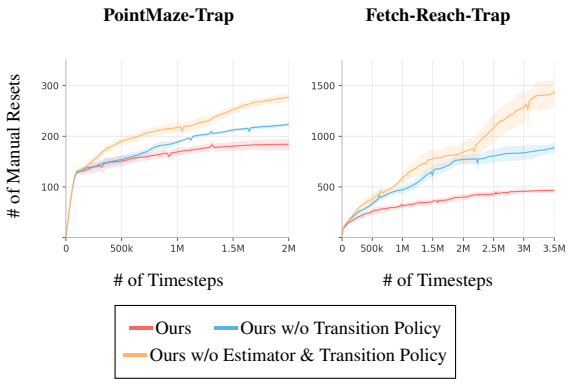


Figure 6: Ablation study on PointMaze-Trap and Fetch-Reach-Trap. Each curve reports the mean over five random seeds, and its shaded area indicates the corresponding standard deviation.

6 Conclusion

We propose an ARL algorithm that learns the state-action sequences leading an agent into irreversible states and predicts an irreversible transition several steps before it occurs. This prediction triggers an early abort while the agent can still move away, and the transition policy then guides the agent to a safe switching state before training resumes. Experimental results show that the learned estimator assigns risk to the states surrounding irreversible states, that both components reduce manual resets, and that our algorithm maintains competitive success rates across four navigation and manipulation tasks. Two directions remain for future work. First, our algorithm uses the predicted risk as a penalty. The same prediction can also serve as a reward in reversible states, and we plan to investigate whether the agent can learn a task from this reward alone. Second, the transition policy currently guides the agent to a single safe switching state. We expect that a curriculum of intermediate states between the aborted state and the safe switching state will let the agent move away from an irreversible state step by step and more safely. We believe that reducing the supervision required to identify irreversible states is a step toward ARL algorithms that generalize across diverse environments and tasks.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Yuri Burda, Harri Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

Yuanpei Chen, Chen Wang, Li Fei-Fei, and C Karen Liu. Sequential dexterity: Chaining dexterous policies for long-horizon manipulation. *arXiv preprint arXiv:2309.00987*, 2023.

Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. *arXiv preprint arXiv:1711.06782*, 2017.

Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, volume 2, pp. 1094–8, 1993.

Sang-Hyun Lee. Robust and scalable autonomous reinforcement learning in irreversible environments. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Sang-Hyun Lee and Seung-Woo Seo. Self-supervised curriculum generation for autonomous reinforcement learning without task-specific knowledge. *IEEE Robotics and Automation Letters*, 9(5): 4043–4050, 2024.

Youngwoon Lee, Shao-Hua Sun, Sriram Somasundaram, Edward S Hu, and Joseph J Lim. Composing complex skills by learning transition policies. In *International conference on learning representations*, 2019.

Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018.

Darshan Patil, Janarthanan Rajendran, Glen Berseth, and Sarath Chandar. Intelligent switching for reset-free rl. *arXiv preprint arXiv:2405.01684*, 2024.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pp. 1312–1320. PMLR, 2015.

Archit Sharma, Michael Ahn, Sergey Levine, Vikash Kumar, Karol Hausman, and Shixiang Gu. Emergent real-world robotic skills via unsupervised off-policy reinforcement learning. *arXiv preprint arXiv:2004.12974*, 2020.

Archit Sharma, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Autonomous reinforcement learning via subgoal curricula. *Advances in Neural Information Processing Systems*, 34:18474–18486, 2021.

Archit Sharma, Rehaan Ahmad, and Chelsea Finn. A state-distribution matching approach to non-episodic reinforcement learning. *arXiv preprint arXiv:2205.05212*, 2022.

Annie Xie, Fahim Tajwar, Archit Sharma, and Chelsea Finn. When to ask for help: Proactive interventions in autonomous reinforcement learning. *Advances in Neural Information Processing Systems*, 35:16918–16930, 2022.

Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4): 1307–1319, 2020.

Henry Zhu, Abhishek Gupta, Aravind Rajeswaran, Sergey Levine, and Vikash Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3651–3657. IEEE, 2019.

Henry Zhu, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine. The ingredients of real-world robotic reinforcement learning. *arXiv preprint arXiv:2004.12570*, 2020.

A TRANSITION POLICY

Algorithm 2 provides the detailed procedure after an early abort. It selects a safe switching state, executes π_{gc} toward the selected state, and stores the collected transitions in B_{gc} and B_{e}.

Algorithm 2 Transition Policy

Require: goal g_k and aborted state s_t

Sample candidate states $\mathcal{C}_k$ from B_{e} labeled reversible by $\mathcal{M}$ and identified as low-risk by E_ϕ
$\mathcal{S}_k^* \leftarrow \{s \in \mathcal{C}_k \mid \lambda_{\mathrm{low}} \le I(s, g_k) \le \lambda_{\mathrm{high}}\}$
$\mathcal{R}_k^*(s_t) \leftarrow \{s \in \mathcal{S}_k^* \mid \lambda_{\mathrm{low}} \le I(s_t, s) \le \lambda_{\mathrm{high}}\}$
Sample $s_{\mathrm{init}}^* \sim \mathrm{Unif}(\mathcal{R}_k^*(s_t))$
for $j \leftarrow 1$ to $T_{\mathrm{transition}}$ **do**
    Sample $a_j \sim \pi_{\mathrm{gc}}(\cdot \mid s_j, s_{\mathrm{init}}^*)$
    Execute a_j and observe (s_{j+1}, o_{j+1}, r_j)
    Store transition in B_{gc} and B_{e}; update π_{gc} and I
end for

B IMPLEMENTATION DETAILS

B.1 POLICY NETWORKS AND IRREVERSIBILITY ESTIMATOR

We train the goal-conditioned policy π_{gc} and the exploration policy π_e using Soft Actor–Critic (Haarnoja et al., 2018). Their actor and critic networks use two hidden layers of 512 units with ReLU activations. Each actor parameterizes a tanh-squashed diagonal Gaussian distribution, and each critic uses two Q-networks. We optimize the actors and critics with Adam (Kingma & Ba, 2015). We separately optimize the state information estimator $I(s, g)$ with RMSProp (Tieleman & Hinton, 2012).

The irreversibility estimator E_ϕ takes the current image observation and action as input. Its image encoder f_ψ has four convolutional layers with stride two. Adaptive average pooling reduces the final 64 feature maps to 1×1 each, producing a 64-dimensional image representation. We concatenate this representation with the action and pass it to E_ϕ, which has two hidden layers with ReLU activations and a sigmoid output. We train f_ψ and E_ϕ jointly with Adam using the binary cross-entropy objective defined in Section 4.1 of the main paper. During policy training, we periodically update both networks as new labeled transitions become available. These updates do not use human-provided reversibility labels. Table 2 summarizes both sets of settings.

We conducted all experiments on a server with eight NVIDIA RTX PRO 5000 Blackwell GPUs, each with approximately 48 GB of memory. Each run used part of a single GPU.

Table 2: Implementation settings. Left: settings shared by the policies and the state information estimator. Right: irreversibility estimator settings.

Hyperparameter	**Value**
Replay batch size	256
Adam β_1	0.9
Adam β_2	0.999
Discount factor	0.99
Initial SAC temperature	0.4
Target update interval	1
Actor/critic hidden layers	2×512
Replay-buffer capacity	5×10^4
Warm-up environment steps	10^5
$(\lambda_{\mathrm{low}}, \lambda_{\mathrm{high}})$	$(0.1, 0.6)$

Parameter	**Value**
Image resolution	84×84 RGB
Encoder channels	$32, 64, 64, 64$
Convolution stride	2
Image feature size	64
E_ϕ hidden layers	2×512
Hidden activation	ReLU
Output activation	Sigmoid
Optimizer	Adam
Learning rate	0.0001
Loss	Binary cross-entropy

B.2 VLM Labeling and Offline Fine-Tuning

We fine-tune Qwen3-VL models (Bai et al., 2025) using LoRA (Hu et al., 2022), and denote each fine-tuned VLM by $\mathcal{M}$. We collect training images from simulated trajectories with varied initial states, random actions, and multiple random seeds. We automatically derive binary labels from the simulated outcomes: 1 for an image observation of an irreversible state and 0 for an image observation of a reversible state. We complete this VLM adaptation before ARL training. During ARL training, the VLM generates reversibility labels from image observations without querying human-provided reversibility labels.

We keep the pretrained VLM weights fixed and train rank-16 LoRA adapters in the attention projection layers, with a scaling factor of 32 and a dropout rate of 0.05. The training objective masks the image and prompt tokens, so the loss is computed only on the assistant response tokens. We train each adapter for three epochs using BF16, gradient checkpointing, fused AdamW (Loshchilov & Hutter, 2019), a 3% warm-up, and a cosine learning-rate schedule. For all environments, we accumulate gradients over four steps to increase the effective batch size without requiring larger per-device batches. The same image–prompt–label format and binary-response objective are used throughout fine-tuning.

Table 3 reports the base VLM, the number of fine-tuning images, and the resulting labeling performance on 2,000 held-out images that were not used for fine-tuning.

Table 3: Base VLM, the numbers of reversible and irreversible fine-tuning images, and the resulting labeling performance on 2,000 held-out images. Performance is reported as accuracy / irreversible-state recall (%).

Environment	Base VLM	Training images	Accuracy / Recall
PointMaze-Trap	Qwen3-VL-2B-Instruct	2,000/2,000	99.15 / 95.02
AntOpen-Trap	Qwen3-VL-4B-Instruct	2,000/2,000	99.70 / 99.77
Fetch-Reach-Trap	Qwen3-VL-2B-Instruct	2,000/2,000	100.00 / 100.00
Highway-Parking-Trap	Qwen3-VL-4B-Instruct	2,000/2,000	98.70 / 98.98

C Supervision Requirements Compared with Prior ARL Algorithms

Throughout the paper, task-specific knowledge refers to the information a learning algorithm requires, for each environment and task, in order to identify irreversible states. It takes two forms in prior ARL algorithms. Some algorithms assume access to privileged state information that is not available from image observations alone. Others require supervision throughout training. Our algorithm uses neither form: it identifies irreversible states from image observations alone, and it completes the VLM adaptation before training begins. The comparison table in the main paper summarizes these requirements.

LNT (Eysenbach et al., 2017) requires supervision throughout training. It triggers an early abort based on the value function of a reset policy, and that policy is trained with a reset reward function evaluated at every step. A designer defines this function for each task. For Cliff Cheetah and Cliff Walker, it combines the distance from the origin, an indicator of whether the agent is standing, and a control penalty. For Pusher, it replaces the task's puck-to-goal distance with the puck-to-start distance. LNT also defines its reset state set by a hand-chosen threshold on this reward. These definitions encode what counts as successful recovery in each task.

PAINT (Xie et al., 2022) removes the need for a reset reward function, but requires supervision throughout training in a different form. Its training loop queries reversibility labels for newly collected states and retrains the classifier on the accumulated data. Binary search reduces the number of queries to $O(\log T)$ per trajectory, but the queries still occur as training proceeds. PAINT reports $3{,}260 \pm 12$, $1{,}021 \pm 69$, $2{,}083 \pm 149$, and $8{,}748 \pm 3{,}762$ queried labels in Maze, Tabletop Organization, Peg Insertion, and Half-Cheetah, respectively, and its discussion identifies the need for thousands of reversibility labels as a limitation.

Our algorithm performs the visual adaptation offline. In our experiments, the labels are derived automatically from the simulator; in a real-world deployment, they would be annotated once before training begins. Human involvement therefore ends before ARL training starts, whereas PAINT requires a human to remain available throughout training and to respond at each intervention. We next examine how much offline data and prompt specificity this adaptation needs.

D Additional VLM Experiments

D.1 Fine-Tuning Data Size

To test whether the VLM requires a large environment-specific dataset, we fine-tune separate LoRA adapters using class-stratified subsets of the same offline image pool. We evaluate each adapter on the same 2,000 held-out images used in Table 3. The zero-shot condition uses the base VLM without LoRA fine-tuning, and all other optimization settings are held fixed. Because false negatives can cause the agent to enter an irreversible state, we report both overall accuracy and irreversible-state recall.

Table 4 shows that the zero-shot condition labels irreversible states poorly, and that performance rises sharply with a few hundred images. Our configuration uses 2,000 images per class, and none of these images are queried from a human during ARL training.

Table 4: Effect of the number of offline fine-tuning images on VLM labeling performance. Each entry reports accuracy / irreversible-state recall (%). Each condition uses an equal number of reversible and irreversible images.

Environment	**Zero-shot**	**250**	**500**	**1,000**	**2,000**
PointMaze-Trap	62.4 / 31.7	88.2 / 74.6	93.5 / 85.1	96.8 / 90.3	99.15 / 95.02
AntOpen-Trap	58.1 / 24.3	85.7 / 71.2	92.1 / 84.8	96.2 / 93.5	99.70 / 99.77
Fetch-Reach-Trap	54.6 / 18.9	90.4 / 82.1	95.8 / 92.7	98.5 / 97.4	100.00 / 100.00
Highway-Parking-Trap	60.2 / 28.5	84.9 / 72.3	91.6 / 83.7	95.4 / 91.8	98.70 / 98.98

D.2 Prompt Specificity

We evaluate two prompt conditions while keeping the image-label data and VLM optimization settings fixed. The *task-specific* condition specifies the camera setup, the coordinate system, the environment appearance, the agent dynamics, and a geometric decision rule for each environment. The *task-agnostic* condition identifies only the controllable agent and the visual cue associated with the irreversible area, together with the binary output rule. We fine-tune a separate LoRA adapter for each condition and use the same prompt during fine-tuning and evaluation.

The task-agnostic prompt uses a common structure across environments:

```
You are given an image of the current environment.  The
controllable agent is [AGENT]. The irreversible area is
indicated by [CUE]. Output 1 if the agent is in or has
contacted the irreversible area; otherwise output 0.  Answer
with only 0 or 1.
```

The environment-specific text inserted into the two placeholders is summarized in Table 5. The same cue is used in all four environments, and the agent description names only the controllable body. This prompt provides no task reward, state coordinates, transition dynamics, reset rule, reachability information, or hand-engineered model of how the agent should recover.

The complete prompts used in the task-specific condition are provided below.

Table 5: Environment-specific visual grounding used in the task-agnostic prompt.

Environment	[AGENT]	[CUE]
PointMaze-Trap	small sphere	purple area
AntOpen-Trap	green Ant body center	purple area
Fetch-Reach-Trap	robot gripper	purple area
Highway-Parking-Trap	green ego vehicle center	purple area

PointMaze-Trap.

TASK Determine whether the current state in PointMaze-Trap is reversible or irreversible.
INPUT A single 84×84 RGB image from a fixed top-down camera. The image shows a compact 7-row by 5-column grid maze.
ENVIRONMENT Light orange or tan blocks are solid maze walls. Dark navy-blue or gray tiles are safe corridors. Lavender or purple tiles form two irreversible areas, one in the northwest corridor and one in the southeast corridor. The walls physically block the agent, whereas the colored tiles do not act as physical barriers.
AGENT AND PHYSICS The controllable agent is a small green translucent sphere. The agent is a two-dimensional point mass driven by continuous forces along the horizontal axes. Its velocity and inertia can carry it across the boundary of an irreversible area after an action. Once the agent center enters either area, the state remains irreversible and the agent cannot autonomously leave it without a manual reset.
VISUAL CUES Judge the position of the entire sphere relative to the purple tiles. When the sphere center is inside an irreversible area, approximately half or more of its silhouette lies over the purple tiles. Nearby purple tiles alone do not make the state irreversible.
IRREVERSIBLE CONDITION Output 1 if the center of the sphere is inside either purple area. Treat at least half of the sphere overlapping the purple tiles as evidence that its center has entered the area.
REVERSIBLE CONDITION Output 0 if the sphere center remains on a safe corridor tile. Output 0 when the sphere only touches a purple boundary or less than half of its silhouette overlaps the purple tiles.
DECISION RULE Use the sphere center and overall silhouette. Ignore the distance to the goal and the proximity of maze walls. Do not label a state irreversible only because the agent is approaching a purple area.
OUTPUT FORMAT Answer with exactly one digit: 0 for reversible or 1 for irreversible.

AntOpen-Trap.

TASK Determine whether the current state in AntOpen-Trap is reversible or irreversible.
INPUT A single 84×84 RGB image from a fixed top-down camera. The image shows an open square maze with world coordinates ranging approximately from -2 to 10 along both horizontal axes.
ENVIRONMENT Dark brown blocks form the maze boundaries. Blue checkerboard tiles are safe floor. Lavender or purple checkerboard tiles form the irreversible area in the upper-right part of the maze. This area does not physically block the agent.
AGENT AND PHYSICS The controllable agent is an Ant quadruped with a round green body center and four articulated legs. Eight continuous joint torques move the legs, while contact forces, momentum, and body orientation determine the motion of the body center. A leg can extend across the boundary while the body center remains outside it. Once the body center enters the irreversible area, the state remains irreversible and the agent cannot autonomously recover without a manual reset.
VISUAL CUES Use the round green body center as the agent position. Flesh, tan, or yellow-brown leg segments are not the body center. Blue floor indicates a reversible location, whereas the purple upper-right area is irreversible.
IRREVERSIBLE CONDITION Output 1 if the green body center is inside the upper-right purple area. The implemented condition is satisfied when the body center has crossed both boundaries, $x > 2.5$ and $y > 2.5$.
REVERSIBLE CONDITION Output 0 if the green body center remains outside the purple area. Output 0 when only the legs or feet touch purple tiles or when the body center remains on the blue floor near the boundary.

DECISION RULE Judge only the green body center. Ignore leg overlap, leg tips, and the Ant silhouette outside the body center. Do not label a state irreversible only because continued motion could later carry the Ant into the purple area.

OUTPUT FORMAT Answer with exactly one digit: 0 for reversible or 1 for irreversible.

Fetch-Reach-Trap.

TASK Determine whether the current state in Fetch-Reach-Trap is reversible or irreversible.

INPUT A single 84×84 RGB image containing two vertically stacked views of the same state. The upper panel is a rotated top-down view of the table, and the lower panel is a side view of the table and partition.

ENVIRONMENT The white rectangular surface is the table. A cross-shaped partition stands on the table. The partition is lavender or purple before contact and becomes bright orange-red when contact occurs.

AGENT AND PHYSICS The controllable agent is the black rectangular gripper at the end of the robot arm. Each continuous action commands a bounded three-dimensional Cartesian displacement of the gripper, while its orientation remains fixed and its fingers remain closed. MuJoCo contact between either gripper finger and the partition triggers the irreversible event. Contact ends the current interaction and requires a manual reset before training can continue.

VISUAL CUES The black rectangular gripper may appear in either view, although it can be difficult to see or outside part of the frame. The partition color is the most reliable cue: lavender or purple before contact and orange-red during contact. Use the two views together to determine whether the gripper touches the partition.

IRREVERSIBLE CONDITION Output 1 if the partition is orange-red. Output 1 if either view clearly shows the gripper touching the partition.

REVERSIBLE CONDITION Output 0 if the partition remains lavender or purple and the gripper is separated from it. Output 0 when the gripper is approaching the partition without contact.

DECISION RULE Check the partition color first. Use visible contact between the gripper and partition as an additional cue. Do not infer contact solely from proximity when the partition remains lavender or purple.

OUTPUT FORMAT Answer with exactly one digit: 0 for reversible or 1 for irreversible.

Highway-Parking-Trap.

TASK Determine whether the current state in Highway-Parking-Trap is reversible or irreversible.

INPUT A single 84×84 RGB image from a top-down camera that follows the ego vehicle. The ego vehicle remains near the image center and appears large in an approximately 15-m local field of view. Most of the parking lot and the global map boundaries can be outside the image.

ENVIRONMENT Dark gray floor is the drivable parking surface. White or light gray lines mark parking slots. Solid purple or magenta areas indicate irreversible parking slots and their associated areas. Yellow wall segments may appear near the image boundary, but wall contact is not used for the irreversible label.

AGENT AND PHYSICS The controllable agent is the green rectangular ego vehicle. Continuous actions control acceleration and steering. The vehicle follows a modified bicycle model: its speed and heading determine its motion, and it cannot translate sideways independently. The purple areas do not block the vehicle. Once the vehicle center enters an irreversible area, external intervention is required before training can continue.

VISUAL CUES Judge the center of the green vehicle rather than its corners or full footprint. Purple immediately beneath the center indicates that the vehicle is in an irreversible parking slot. The irreversible area can extend into a narrow gray approach lane beside a purple slot row. A wide open gray corridor or a non-purple slot indicates a reversible location.

IRREVERSIBLE CONDITION Output 1 if the ego-vehicle center lies on solid purple fill. Output 1 if the center lies in the narrow gray approach lane directly beside a purple slot row and at least one of the following cues is present: a continuous purple band lies within approximately one vehicle length of the center; a purple slot row is separated from

the center by only a thin gray strip; or purple occupies approximately one quarter or more of the local image around the ego vehicle.

REVERSIBLE CONDITION Output 0 if the vehicle center lies in the wide central corridor and purple is absent or distant. Output 0 if the vehicle center lies in a safe non-purple parking slot. Output 0 when only a vehicle edge or corner overlaps purple and the center remains outside the irreversible area.

DECISION RULE Judge the ego-vehicle center and its local relation to the gray floor and purple areas. Do not infer global coordinates or compass directions from the ego-follow image. Ignore walls and wall collisions. Do not use a small or distant purple patch as sufficient evidence of an irreversible state.

OUTPUT FORMAT Answer with exactly one digit: 0 for reversible or 1 for irreversible.

Table 6 shows that the task-agnostic condition performs comparably to the task-specific condition in all four environments. Removing the coordinate system, the agent dynamics, and the geometric decision rule therefore does not degrade labeling performance. Together with Appendix D.1, this indicates that the VLM adaptation needs neither a large dataset nor a hand-written description of what makes a state irreversible.

Table 6: Effect of prompt specificity on VLM labeling performance. Each entry reports accuracy / irreversible-state recall (%).

Environment	Task-specific	Task-agnostic
PointMaze-Trap	99.15 / 95.02	98.70 / 93.21
AntOpen-Trap	99.70 / 99.77	99.25 / 98.94
Fetch-Reach-Trap	100.00 / 100.00	99.80 / 99.60
Highway-Parking-Trap	98.70 / 98.98	97.95 / 97.84

E Environment and Evaluation Details

The main paper introduces the four irreversible environments used in our experiments. Here, we specify the irreversible condition and evaluation states for each task. A manual reset is required whenever the agent enters an irreversible state. Fetch-Reach-Trap is adapted from the multi-goal Fetch environments (Plappert et al., 2018) and uses MuJoCo (Todorov et al., 2012). Highway-Parking-Trap is adapted from HighwayEnv (Leurent, 2018). Table 7 summarizes the state, action, and goal spaces and episode lengths of the evaluation tasks. The state dimension excludes the goal, which is provided separately to the goal-conditioned policy.

Table 7: State, action, and goal spaces and episode lengths of the evaluation tasks.

Task	State Space $\mathcal{S}$	Action Space $\mathcal{A}$	Goal Space $\mathcal{G}$	Episode Length
PointMaze-Trap	$\mathbb{R}^4$	$\mathbb{R}^2$	$\mathbb{R}^2$	200
AntOpen-Trap	$\mathbb{R}^{29}$	$\mathbb{R}^8$	$\mathbb{R}^2$	300
Fetch-Reach-Trap	$\mathbb{R}^{10}$	$\mathbb{R}^4$	$\mathbb{R}^3$	50
Highway-Parking-Trap	$\mathbb{R}^6$	$\mathbb{R}^2$	$\mathbb{R}^6$	50

PointMaze-Trap. The point agent enters an irreversible state when it reaches either purple area. Figure 7 shows three goal states and five initial states used for evaluation. Excluding pairs at the same position yields 12 initial–goal combinations. Goals are marked in red and initial states in green.

AntOpen-Trap. The Ant enters an irreversible state when its body center reaches the purple area; contact by a leg alone does not count. Figure 8 shows three goal states and five initial states used for evaluation. Excluding pairs at the same position yields 12 initial–goal combinations.

Fetch-Reach-Trap. In Fetch-Reach-Trap, a purple cross-shaped partition divides the workspace into four cells. The partition has a height of $0.1333\,\mathrm{m}$. The gripper enters an irreversible state when

it touches the partition. Figure 9 shows four evaluation states, each used as both an initial state and a goal state. Pairing each state with the other three goals yields 12 ordered initial–goal combinations.

Highway-Parking-Trap. The vehicle enters an irreversible state when its center reaches a purple parking slot; collisions with the surrounding wall remain reversible. Figure 10 shows eight goal states and two initial states used for evaluation. In each row, the four goal states are paired with the initial state shown at the right. These pairings yield eight initial–goal combinations.

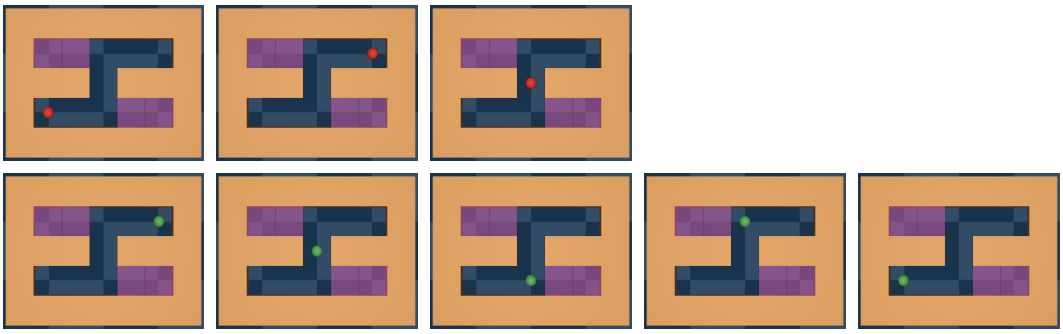


Figure 7: Evaluation states in PointMaze-Trap. The top row shows the three goal states, and the bottom row shows the five initial states.

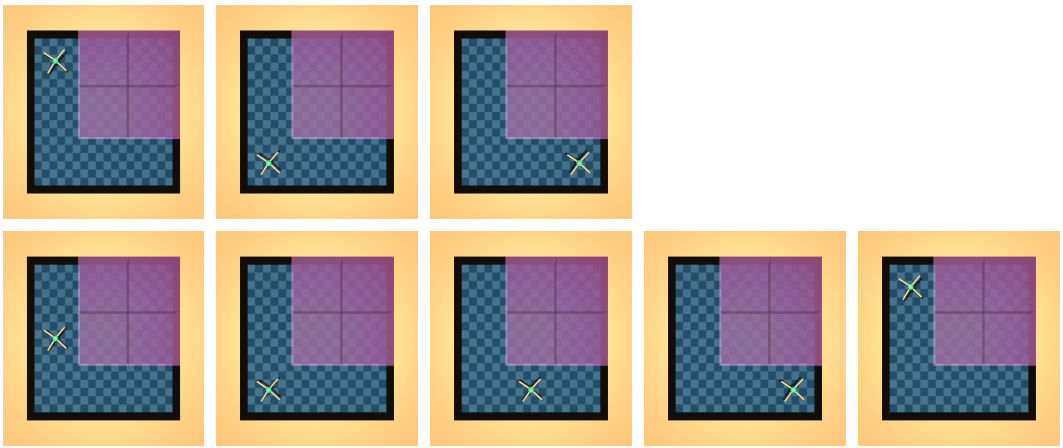


Figure 8: Evaluation states in AntOpen-Trap. The top row shows the three goal states, and the bottom row shows the five initial states.

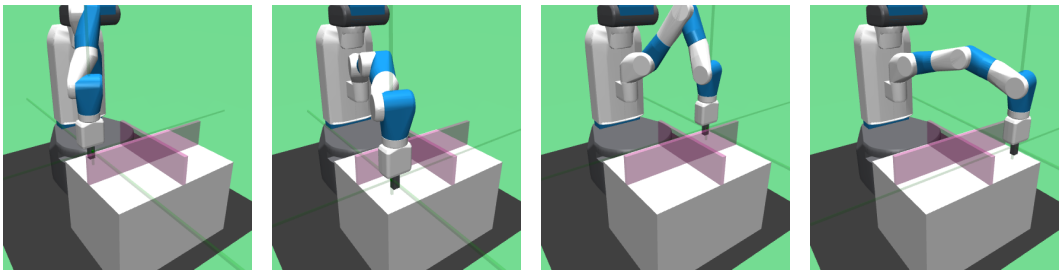


Figure 9: Evaluation states in Fetch-Reach-Trap. Each of the four states serves as both an initial state and a goal state.

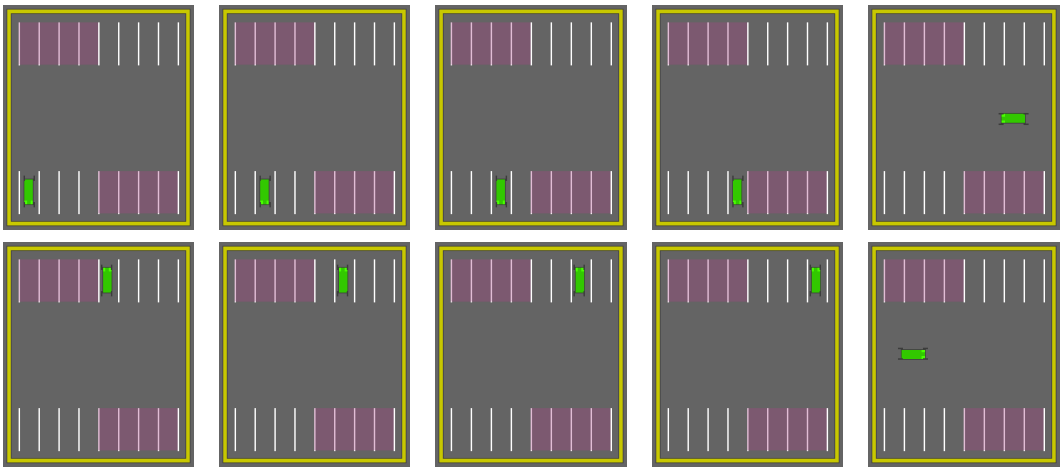


Figure 10: Evaluation states in Highway-Parking-Trap. Each row shows four goal states followed by their paired initial state.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. *arXiv preprint arXiv:1711.06782*, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870, 2018.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Edouard Leurent. An environment for autonomous driving decision-making. `https://github.com/Farama-Foundation/HighwayEnv`, 2018.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5—rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

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*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Annie Xie, Fahim Tajwar, Archit Sharma, and Chelsea Finn. When to ask for help: Proactive interventions in autonomous reinforcement learning. *Advances in Neural Information Processing Systems*, 35:16918–16930, 2022.