

A Training Procedure Details

Algorithm 2 describes how RSA selects an informative goal state g_k at iteration k based on the agent’s learning progress. This procedure corresponds to `get_informative_goal($B_r, \mathcal{G}_{1:k-1}, I(s, g)$)` in Algorithm 1 of the main paper. RSA first checks whether the previous goal state g_{k-1} was achieved. If so, it samples a batch of N_s initial states from the reset buffer B_r and computes the average probability of reaching g_{k-1} from those states. If the average probability lies within the range $[\lambda_1, \lambda_2]$, the goal state g_{k-1} is considered still informative and is reused. Otherwise, RSA samples N_g candidate goal states from the previously collected goal state set $\mathcal{G}_{1:k-1}$ to search for a new informative goal state. For each candidate goal \bar{g}_j , RSA computes the average probability of reaching it from the sampled initial states and returns the first goal state whose probability lies within the same range. If the previous goal state g_{k-1} was not achieved, RSA reuses g_{k-1} as the goal state for the k th iteration.

Algorithm 2 Sample g_k with `get_informative_goal($B_r, \mathcal{G}_{1:k-1}, I(s, g)$)`

```

1: Given: Final state within forward rollout  $s_T$  and previous goal state  $g_{k-1}$ 
2: if previous goal state  $g_{k-1}$  is achieved then
3:   Sample candidate initial states  $\{\bar{s}_i\}_{N_s} \sim B_r$ 
4:   if  $\lambda_1 \leq \frac{1}{N_s} \sum_{i=1}^{N_s} I(\bar{s}_i, g_{k-1}) \leq \lambda_2$  then
5:     return goal state  $g_{k-1}$ 
6:   else
7:     Sample candidate goal states  $\{\bar{g}_j\}_{N_g} \sim \mathcal{G}_{1:k-1}$ 
8:     for  $j \leftarrow 1 \dots N_g$  do
9:       if  $\lambda_1 \leq \frac{1}{N_s} \sum_{i=1}^{N_s} I(\bar{s}_i, \bar{g}_j) \leq \lambda_2$  then
10:        return goal state  $\bar{g}_j$ 
11:   else
12:     return goal state  $g_{k-1}$ 

```

In the early stages of training, the state information estimator and the successor features may struggle to accurately identify informative or irreversible states. Using them prematurely could result in biased identification and suboptimal performance. To mitigate this issue, RSA initializes both models to classify all states as informative and reversible, and utilizes a warm-up period during which the agent takes random actions to explore the environment without performing any updates. During this period, RSA does not rely on either model, as they have not yet been trained sufficiently. We empirically observed that these simple techniques are highly effective for collecting diverse and unbiased experiences and stabilizing early training.

As described in Algorithm 1 of the main paper, RSA computes surrogate rewards that penalize the agent for entering irreversible states. The surrogate reset and forward reward functions, \hat{r}_g^r and \hat{r}_g^f , are defined as follows:

$$\hat{r}_g^r(s, a, s') = \begin{cases} r_g^i(s, a, s'), & s \notin S_{\text{irr}} \\ \mathcal{R}_{\min} - \epsilon, & s \in S_{\text{irr}} \end{cases} \quad \text{and} \quad \hat{r}_g^f(s, a, s') = \begin{cases} r_g(s, a, s'), & s \notin S_{\text{irr}} \\ \mathcal{R}_{\min} - \epsilon, & s \in S_{\text{irr}} \end{cases}$$

where r_g^i is the intrinsic reward generated by an off-the-shelf exploration algorithm, r_g is the extrinsic reward given by the experiment, and \mathcal{R}_{\min} is the lower bound of both rewards, and ϵ is a small positive constant. To penalize visits to irreversible states, we ensure that these states receive a reward of $\mathcal{R}_{\min} - \epsilon$, which is strictly lower than the minimum reward assigned to reversible states. Note that RSA identifies the set of irreversible states without task-specific knowledge.

The surrogate reset and forward reward functions encourage the agent to avoid irreversible states, as demonstrated in standard RL settings by Xie et al. [8]. We extend this to goal-conditioned RL settings. The superscripts r and f of the reward functions and the corresponding subscripts of the policies are omitted for brevity, as the subsequent equations hold for both the reset and forward cases. Since the minimum value of the reward functions is $\mathcal{R}_{\min} - \epsilon$, the lower bound of the goal-conditioned

value function is

$$Q^\pi(s, a, g) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_g(s_t, a_t, s_{t+1}) \mid S_0 = s, A_0 = a \right] \geq \sum_{t=0}^{\infty} \gamma^t (\mathcal{R}_{\min} - \epsilon) = \frac{\mathcal{R}_{\min} - \epsilon}{1 - \gamma},$$

where the lower bound is achieved if $s \in S_{\text{irr}}$. Let $s^+ \notin S_{\text{irr}}$ denote a reversible state and $s^- \in S_{\text{irr}}$ denote an irreversible state. The goal-conditioned value function for a reversible state $s^+ \notin S_{\text{irr}}$ can be written as follows:

$$\begin{aligned} Q^\pi(s^+, a, g) &= r_g(s^+, a, s') + \gamma \mathbb{E}_{a' \sim \pi(\cdot | s')} [Q^\pi(s', a', g)] \\ &\geq \mathcal{R}_{\min} + \gamma \frac{\mathcal{R}_{\min} - \epsilon}{1 - \gamma} \\ &= \epsilon + (\mathcal{R}_{\min} - \epsilon) + \gamma \frac{\mathcal{R}_{\min} - \epsilon}{1 - \gamma} = \epsilon + \frac{\mathcal{R}_{\min} - \epsilon}{1 - \gamma} = \epsilon + Q^\pi(s^-, a', g). \end{aligned}$$

The above result demonstrates that the goal-conditioned value function for reversible states is greater than that for irreversible states on deterministic MDPs. We refer the reader to Xie et al. [8] for additional analysis on stochastic MDPs.

B Implementation Details

The forward and reset policies of RSA can be trained using any reinforcement learning and exploration algorithms. In our experiments, we use Soft Actor-Critic (SAC) [4] and Random Network Distillation (RND) [2], as both are stable and well-established. RND involves a target network $f(s) : S \rightarrow \mathbb{R}^k$ and a predictor network $\hat{f}(s) : S \rightarrow \mathbb{R}^k$. The target network is randomly initialized and remains fixed throughout training, while the predictor network is trained to predict the output of the target network by minimizing the expected prediction error $\|\hat{f}(s) - f(s)\|^2$. The key idea of RND is that the prediction error is higher for novel states than for frequently visited states. Leveraging this idea, RND defines the prediction error as the intrinsic reward function r^i . We implement the goal-conditioned intrinsic reward function r_g^i by extending the intrinsic reward function r^i proposed in RND to goal-conditioned RL settings, incorporating a goal state g as an additional input to both the target and predictor networks. The goal-conditioned intrinsic reward function is then given by $r_g^i = \|\hat{f}(s, g) - f(s, g)\|^2$. This goal-conditioned extension enables the intrinsic reward to vary across different goal states, even for the same state, based on how frequently the state has been visited with respect to each goal. The SFs, used to encode behaviors exhibited in irreversible states, are trained to minimize the temporal-difference (TD) error on subsequent states [1, 6]. We normalize the feature vectors of the SFs using a running mean and standard deviation.

The main models of RSA, including the forward policy, reset policy, state information estimator, and SFs, consist of two hidden layers with 512 units and ReLU activations. Both the forward and reset policies output the mean and standard deviations of a Gaussian distribution over continuous actions. The state information estimator includes an additional softmax layer to estimate the probability of reaching goal states from initial states. We use the Adam optimizer [5] to update the parameters of all models except the state information estimator, for which we use the RMSProp optimizer [7]. Our experiments are conducted on two machines: a PC equipped with an RTX 5080 GPU and a workstation with four RTX 4090 GPUs. Each RSA run uses a fraction of a single GPU.

Hyperparameter	Value
Batch Size	512
Adam β_1	0.9
Adam β_2	0.999
Learning Rate	0.0003
Discount Factor	0.99
Temperature	0.4
Target Update Interval	1
Target Smoothing Coefficient	0.005
λ_1	0.1
λ_2	0.6
λ_3	-0.1

Table 2: Hyperparameters

To ensure fair comparisons, the main models in the baselines are implemented using the same network architecture as those in RSA and trained with the same algorithms. We performed a grid search for the hyperparameters and applied the same values to both RSA and the baselines. Table 2 summarizes the key hyperparameters used in our experiments.

C Experimental Details

C.1 Environments

This section provides additional information about the navigation and manipulation tasks used in our experiments. All tasks are either used directly from Gymnasium-Robotics [3] or adapted from it. Note that in all tasks, the reset reward is not the extrinsic reward provided by the environment, but rather the intrinsic reward generated by the exploration algorithm. Table 3 presents an overview of the main properties of these tasks. The following paragraphs describe each task in more detail.

Task	Type	State Space \mathcal{S}	Action Space \mathcal{A}	Goal Space \mathcal{G}	Episode Length
PointIMaze	Reversible	\mathbb{R}^4	\mathbb{R}^2	\mathbb{R}^2	200
AntOpen	Reversible	\mathbb{R}^{29}	\mathbb{R}^8	\mathbb{R}^2	300
HandReach	Reversible	\mathbb{R}^{63}	\mathbb{R}^{20}	\mathbb{R}^{15}	100
PointIMaze-Trap	Irreversible	\mathbb{R}^4	\mathbb{R}^2	\mathbb{R}^2	200
AntOpen-Trap	Irreversible	\mathbb{R}^{29}	\mathbb{R}^8	\mathbb{R}^2	300
HandManipulate	Irreversible	\mathbb{R}^{61}	\mathbb{R}^{20}	\mathbb{R}^7	200

Table 3: Evaluation Tasks.

PointIMaze: PointIMaze does not contain any irreversible states. A state represents the agent’s position and velocity along the X and Y axes, while an action corresponds to the forces applied to the agent in those directions. A key challenge in this task is that the maze contains several dead ends, which make it difficult for the agent to gain experience reaching diverse goal states. Evaluation episodes are constructed by pairing the initial and goal states shown in Figure 8, excluding pairs with identical positions.

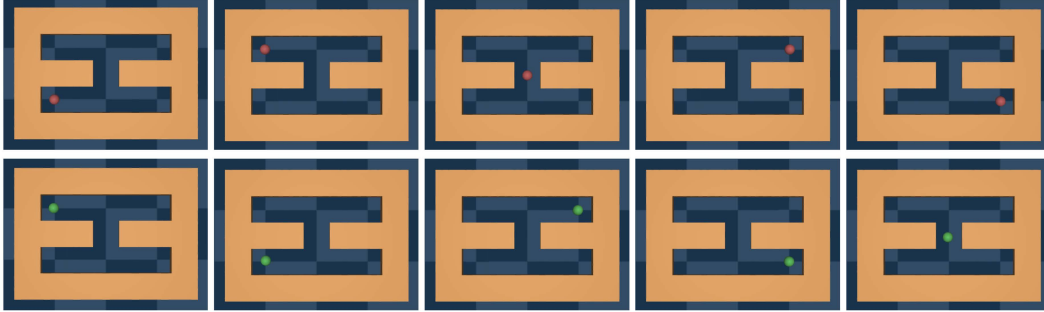


Figure 8: Visualization of goal and initial states used for evaluation in PointIMaze. The top and bottom rows show the goal and initial states, respectively.

PointIMaze-Trap: PointIMaze-Trap is a variant of PointIMaze that introduces irreversible states. It shares the same state and action spaces as PointIMaze. While the agent can continue to move after colliding with brown boxes, it cannot move further after colliding with purple boxes unless external

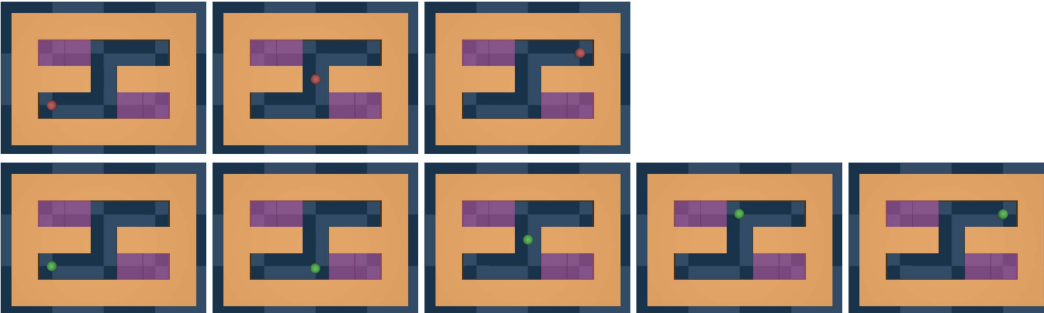


Figure 9: Visualization of goal and initial states used for evaluation in PointIMaze-Trap. The top and bottom rows show the goal and initial states, respectively.

intervention is provided. This scenario is analogous to a vehicle that, after an accident, can no longer operate without repairs. A key challenge in this task is that the agent must identify the purple boxes, corresponding to irreversible states, without access to privileged information. Evaluation episodes are constructed using the initial and goal states shown in Figure 9, excluding pairs in which the two states occupy the same position.

AntOpen: AntOpen does not contain any irreversible states. A state includes the agent’s position, orientation, velocity, and joint information, while an action corresponds to the torques applied to the joints. Because the agent in this task exhibits more complex dynamics than in PointIMaze, it is difficult to obtain informative initial and goal states through random sampling. Evaluation episodes are constructed by pairing the initial and goal states shown in Figure 10, excluding pairs with identical positions.

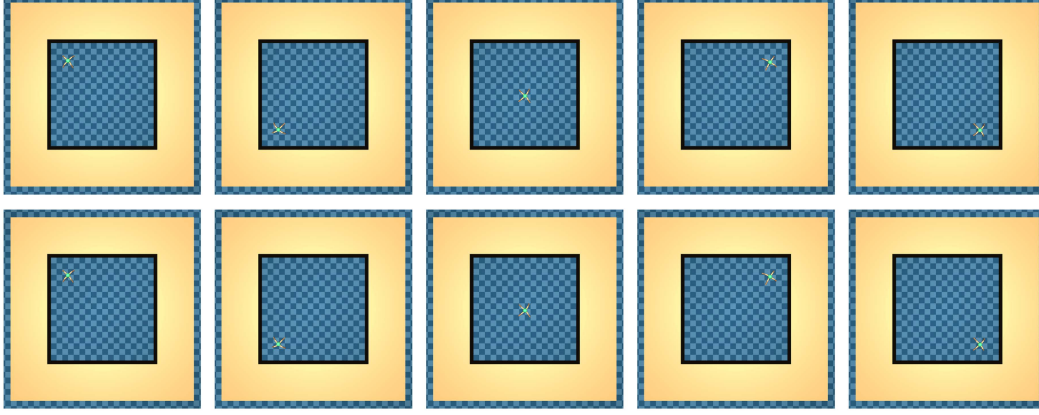


Figure 10: Visualization of goal and initial states used for evaluation in AntOpen. The top and bottom rows show the goal and initial states, respectively.

AntOpen-Trap: AntOpen-Trap is a variant of AntOpen that introduces irreversible states. It shares the same state and action spaces as AntOpen. Once the agent enters the purple boxes, which are designated as irreversible states, it cannot move further without external intervention. This scenario is analogous to a robot that cannot move on its own due to internal failure or physical damage. The agent has no access to privileged information for identifying these irreversible states. Evaluation episodes are constructed using the initial and goal states shown in Figure 11, excluding pairs in which the two states occupy the same position.

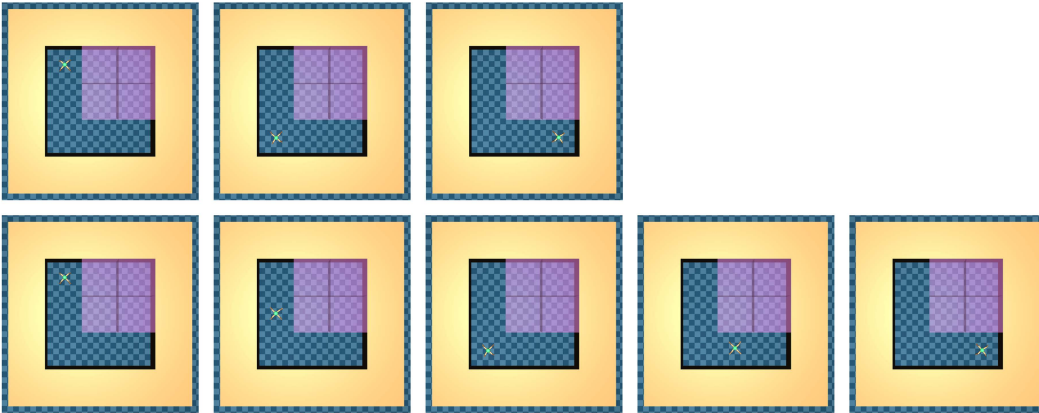


Figure 11: Visualization of goal and initial states used for evaluation in AntOpen-Trap. The top and bottom rows show the goal and initial states, respectively.

HandReach: HandReach is a reversible manipulation task referred to as Hand-Reach-v0 in Gymnasium-Robotics [3]. A state consists of the joint and finger states of the agent, which has 24

degrees of freedom. An action corresponds to the absolute angular positions of the actuated joints. The goal state is 15-dimensional and specifies the target position of each fingertip. We include this manipulation task to demonstrate that our algorithm is task-agnostic and applicable to a wide range of tasks beyond navigation. Figure 12 illustrates the sets of initial and goal states used to construct the evaluation episodes.

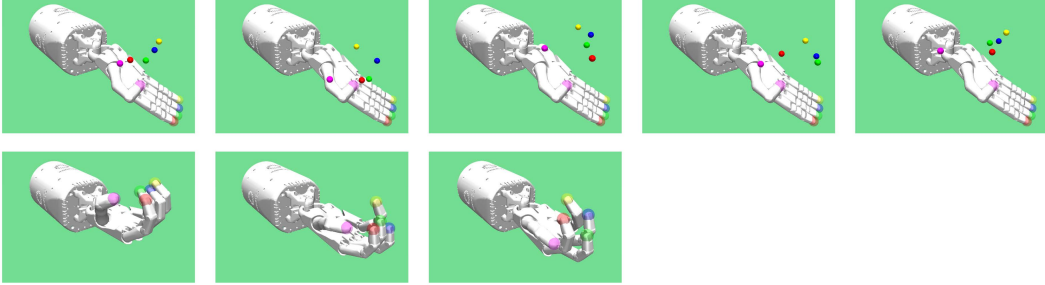


Figure 12: Visualization of goal and initial states used for evaluation in HandReach. The top and bottom rows show the goal and initial states, respectively.

HandManipulate: HandManipulate is an irreversible manipulation task referred to as HandManipulateEggRotate-v0 in Gymnasium-Robotics [3]. A state consists of the joint and finger states of the agent, along with the orientation and velocities of the egg-shaped object. The action space is identical to that of HandReach. The goal state is 4-dimensional and defines the target orientation of the object. Dropping the object results in an irreversible state, in which the agent can no longer pick up or manipulate it without external intervention. Figure 13 illustrates the sets of initial and goal states used to construct the evaluation episodes.

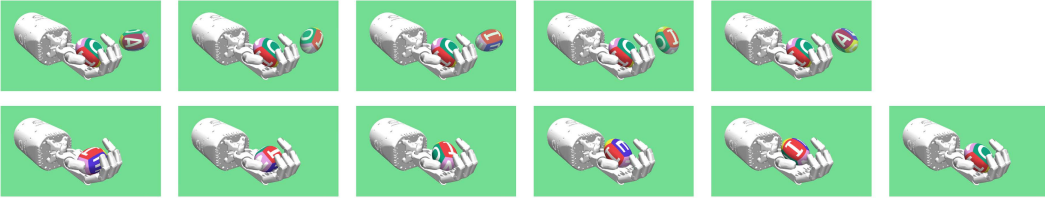


Figure 13: Visualization of goal and initial states used for evaluation in HandManipulate. The top and bottom rows show the goal and initial states, respectively.

C.2 Additional Results

We claim that the SFs for irreversible state-action pairs are lower than those for reversible pairs. This claim is based on our observation that, in irreversible states, an agent loses control over goal-relevant features regardless of the actions it takes. To validate this claim, we compare the SFs for irreversible state-action pairs with those for reversible ones. Figure 14 illustrates how the SFs for irreversible

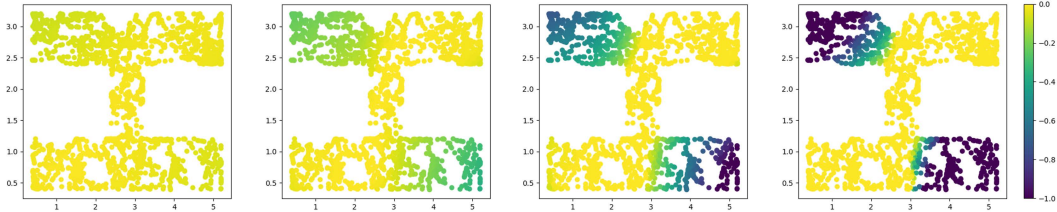


Figure 14: Changes in irreversible states identified over training. Each column shows how the irreversible states are identified as training progresses from left to right. The color of each state indicates the SFs for the corresponding state-action pair.

state–action pairs evolve during training on PointIMaze-Trap. As shown in Figure 3 of the main paper, this task involves irreversible states located in the top-left and bottom-right corridors. The visualization results show that, at the beginning of training, the SFs for both irreversible and reversible state–action pairs have similar values, all higher than the reversibility threshold. This occurs because we initialize the SFs such that all states are treated as reversible, allowing the agent to collect diverse and unbiased experiences. As training progresses, the SFs for state–action pairs in the top-left and bottom-right corridors gradually decrease below those for other pairs. These results indicate that the irreversible states identified by RSA are consistent with our intuitive notion of irreversibility.

We also investigate whether RSA can correctly identify irreversible states in stochastic tasks. To this end, we implemented a new stochastic task, PointIMaze-Trap-Stochastic, a variant of PointIMaze-Trap. In this task, the agent transitions to the next state not based on the action it actually executes but on an action perturbed by uniform noise in the range $[-0.5, 0.5]$. Figure 15 visualizes the irreversible states identified in PointIMaze-Trap-Stochastic in the same manner as Figure 14. In PointIMaze-Trap, only states located in or very near the top-left and bottom-right corridors were identified as irreversible, whereas in PointIMaze-Trap-Stochastic, states located farther from these corridors were also identified as irreversible. Given that uncertainty in state transitions expands the region where states are more likely to enter irreversible areas, these results suggest that RSA can reliably identify irreversible states even in stochastic tasks. We hypothesize that this robustness arises because the SFs in RSA are optimized to minimize the temporal-difference (TD) error over subsequent states. In other words, the SFs inherently capture transition stochasticity by updating expectations over transitions instead of assuming deterministic dynamics.

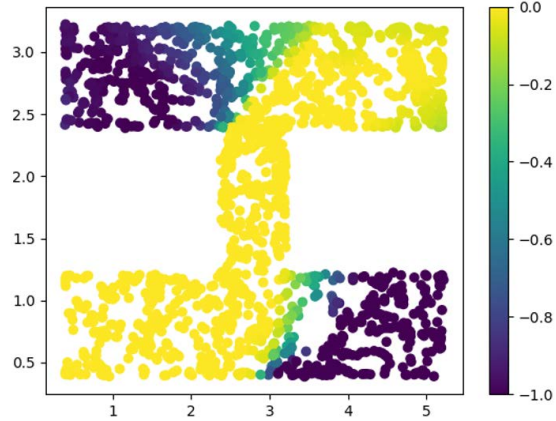


Figure 15: Irreversible states identified in PointIMaze-Trap-Stochastic. The color of each state indicates the SFs for the corresponding state-action pair.

D Societal Impact

RSA enables real-world agents to autonomously collect data and improve their performance without human supervention. This allows us to enhance a wide range of robotic systems, particularly in settings where human intervention is impractical or costly.

However, such autonomous agents may exhibit unintended behaviors. This risk is particularly significant in safety-critical applications, such as military robots and autonomous vehicles, where these behaviors could lead to physical harm or operational failures. We believe that alignment techniques, which are designed to ensure that agents act in accordance with human intent, should be actively studied and developed alongside RSA.

References

- [1] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.
- [2] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, pages 4003–4019, 2019.
- [3] Rodrigo de Lazcano, Kallinteris Andreas, Jun Jet Tai, Seungjae Ryan Lee, and Jordan Terry. Gymnasium robotics, 2024. URL <http://github.com/Farama-Foundation/Gymnasium-Robotics>.

- [4] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.
- [5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [6] Hao Liu and Pieter Abbeel. Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, pages 6736–6747. PMLR, 2021.
- [7] Tijmen Tieleman, Geoffrey Hinton, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.
- [8] 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.