World Models Increase Autonomy in Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Reinforcement learning (RL) is an appealing paradigm for training intelligent agents, enabling policy acquisition from the agent's own autonomously acquired experience. However, the training process of RL is far from automatic, requiring extensive human effort to reset the agent and environments. To tackle the challenging reset-free setting, we first demonstrate the superiority of model-based (MB) RL methods in such setting, showing that a straightforward application of MBRL can outperform all the prior state-of-the-art methods while requiring less supervision. We then identify limitations inherent to this direct extension and propose a solution called model-based reset-free (MoReFree) agent, which further enhances the performance. MoReFree adapts two key mechanisms, exploration and policy learning, to handle reset-free tasks by prioritizing task-relevant states. It exhibits superior data-efficiency across various reset-free tasks without access to environmental reward or demonstrations while significantly outperforming privileged baselines that require supervision. Our findings suggest model-based methods hold significant promise for reducing human effort in RL. Website: `https://sites.google.com/view/morefree`

1 Introduction

Reinforcement learning presents an attractive framework for training capable agents. At first glance, RL training appears intuitive and autonomous - once a reward is defined, the agent learns from its own automatically gathered experience. However, in practice, RL training often assumes the access to environmental resets that can require significant human effort to setup, which poses a significant barrier for real world applications of RL like robotics.

Most RL systems on real robots to date have employed various strategies to implement resets, all requiring a considerable amount of effort [17, 32, 34, 21]. In [21], which trains a dexterous hand to rotate balls, the practitioners had to (1) position a funnel underneath the hand to catch dropped balls, and (2) deploy a separate robot arm to pick up the dropped balls for resets, and (3) script the reset behavior. These illustrate that even for simple behaviors, proper implementation of reset mechanisms can result in significant human effort and time.

Rather than depending on human-engineered reset mechanisms, the agent can operate within a reset-free training scheme, learning to reset itself [4, 27, 25, 12] or train a policy capable of starting from diverse starting states [35]. However, the absence of resets introduces unique exploration challenges. Without periodic resets, the agent can squander significant time in task-irrelevant regions that require careful movements to escape and may overexplore, never returning from indefinite exploration. Recent unsupervised model-based RL (MBRL) approaches [20, 14] in the episodic setting have shown sophisticated exploration, high data-efficiency and promising results in long-horizon tasks. This prompts the question: *would MBRL agents excel in the reset-free RL setting?*

As an initial attempt, we first evaluate an unsupervised MBRL agent, out-of-the-box, in a reset-free Ant locomotion task. The ant is reset to the center of a rectangular arena, and is tasked with navigating to the upper right corner. The agent is reset only once at the start of training. The evaluation is episodic - the agent is reset at the start of each evaluation episode.

For the MBRL agent, we use PEG [14], which was developed to solve hard exploration tasks in the episodic setting. As seen in Figure 1, PEG, out of the box, outperforms prior state-of-the-art, model-free agent, IBC [15], tailored for the reset-free setting.

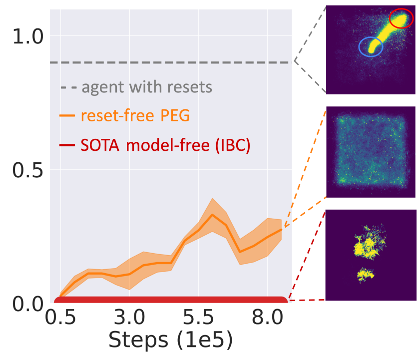

Figure 1: Performance and collected data of different agents on the reset-free Ant locomotion task.

In Figure 1, we plot state visitation heatmaps of the agents, where lighter colors correspond to more visitations. The oracle agent, with access to resets, explores the the "task-relevant" area between the initial and top right corner, which is ideal for training a policy that succeeds in episodic evaluation. IBC's heatmap (bottom) shows that it fails to explore effectively, never encountering the goal states in the top right region. In contrast, PEG exhaustively explores the entire space, as seen through its uniform heatmap. This results in an overexploration problem - PEG may devote considerable time on finding irrelevant states rather than concentrating on the task-relevant region of the task. This leads us to ask: **how can MBRL agents acquire more task-relevant data in the reset-free setting to improve its performance?**

We propose **Mo**del-based, **Re**set-**Free** (MoReFree), which improves two key mechanisms in model-based RL, exploration and policy optimization, to better handle reset-free training. Following the top row of Figure 2: to gather task-relevant data without resets, we define a training curriculum that alternates between temporally extended phases of task solving, resetting, and exploration. Next, as seen in the bottom row of Figure 2, we bias the policy training within the world model towards achieving task-relevant goals such as reaching initial states and evaluation states.

Our key contributions are as follows: **(1)** We demonstrate the viability of using model-based agents with strong exploration abilities for the reset-free setting as well as their inherent limitations. We address such limitations through the MoReFree framework which focuses exploration and policy optimization on task-relevant states. **(2)** We evaluate the out-of-the-box MBRL baseline and MoReFree against state-of-the-art reset-free methods in 8 challenging reset-free tasks ranging from manipulation to locomotion. Notably, both model-based approaches outperform prior state-of-the-art baselines in 7/8 tasks in final performance and data efficiency, all the while requiring less supervision (e.g. environmental reward or demonstrations). MoReFree outperforms the model-based baseline in the 3 hardest tasks. **(3)** We perform in-depth analysis of the MoReFree and baselines behaviors, and show that MoReFree explores the state space thoroughly while retaining high visitation counts in the task-relevant regions. Our ablations show that the performance gains of MoReFree come from the proposed design choices and justify the approach.

2 Related Work

Reset-free RL: There is a growing interest in researching reinforcement learning methods that can effectively address the complexities of reset-free training. [28] proposes a reset-free RL benchmark (EARL) and finds that standard RL methods like SAC [9] fail catastrophically in EARL. Multiple approaches have been proposed to address reset-free training, which we now summarize. One approach is to add an additional reset policy, to bring the agent back to suitable states for learning [4, 16, 27, 25, 15]. LNT [4] and [16] train a reset policy to bring the agent back to initial state distribution, supervised by dense rewards and demonstrations respectively. MEDAL [25, 26], train a goal-conditioned reset policy and direct it to reset goal states from demonstrations. IBC [15] defines a curriculum for both task and reset policies without requiring demonstrations. VaPRL [27] trains a single goal-conditioned policy to reach high value states close to the initial states. Instead of guiding the agent back to familiar states, R3L [35] and [31] learn to reset the policy to diverse initial states, resulting in a policy that is more robust to variations in starting states. However, such methods are

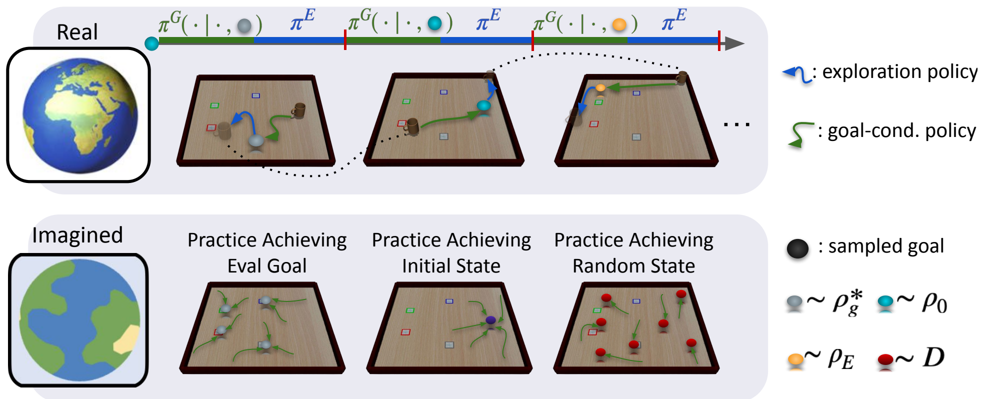

Figure 2: MoReFree is a model-based RL agent for solving reset-free tasks. **Top row:** MoReFree strikes a balance between exploring unseen states and practicing optimal behavior in task-relevant regions by directing the goal-conditioned policy to achieve evaluation states, initial state states (emulating a reset), and exploratory goals. **Bottom row:** MoReFree focuses the goal-conditioned policy training inside the world model on achieving evaluation states, initial states, and random replay buffer states to better prepare the policy for the aforementioned exploration scheme.

limited to tasks where exploration is unchallenging. The vast majority of reset-free approaches are model-free, with a few exceptions [19, 18]. Other works [8, 29] model the reset-free RL training process as a multi-task RL problem and require careful definition of the task distribution such that the tasks reset each other.

Goal-conditioned Exploration: A common theme running through the aforementioned work is the instantiation of a curriculum, often through commanding goal-conditioned policies, to keep the agent in task-relevant portions of the environment while exploring. Closely related is the subfield of goal-conditioned exploration in RL, where a goal-conditioned agent selects its own goals during training time to generate data. There is a large variety of approaches for goal selection, such as task progress [1, 30], intermediate difficulty [6], value disagreement [33], state novelty [23, 22], world model error [14, 24], and more. Many goal-conditioned exploration methods use the “Go-Explore" [3] strategy, which first selects a goal and runs the goal-conditioned policy (“Go"-phase), and then switches to an exploration policy for the latter half of the episode (“Explore"-phase). PEG [14], which MoReFree uses, extends Go-Explore to the model-based setting, and utilizes the world model to plan states with higher exploration value as goals. However, such methods are not designed for the reset-free RL setting, and may suffer from *over-exploration* of task-irrelevant states.

Table 1: A conceptual overview of reset-free methods. Existing methods are model-free, and most of them require other forms of supervision (environmental reward or demonstrations or both). In performance, MoReFree improves over reset-free PEG, which significantly outperforms privileged baselines IBC, MEDAL and R3L.

Approach	MEDAL	IBC	VaPRL	R3L	reset-free PEG	MoReFree
Model-based	✗	✗	✗	✗	✓	✓
Other Supervision	✓	✓	✓	✗	✗	✗

We notice that the majority of all prior work are model-free and may suffer from poor sample efficiency and exploration issues. In contrast, our model-based approaches use world models to efficiently train policies and perform non-trivial goal-conditioned exploration with minimal supervision. See Table 1 for a conceptual comparison between prior work and two model-based methods (MoReFree and reset-free PEG).

3 Preliminaries

3.1 Reset-free RL

We follow the definition of reset-free RL from EARL [28], and extend it to the goal-conditioned RL setting. Consider the goal-conditioned Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{G}, \mathcal{A}, p, r, \rho_0, \rho_{g^*}, \gamma)$. At each time step t in the state $s_t \in \mathcal{S}$, a goal-conditioned policy $\pi(\cdot|s_t, g)$ under the goal command $g \in \mathcal{G}$ selects an action $a_t \in \mathcal{A}$ and transitions to the next state s_{t+1} with the probability $p(s_{t+1}|s_t, a_t)$, and gets a reward $r(s_t, a_t, g)$. ρ_0 is the initial state distribution, ρ_{g^*} is the evaluation goal distribution, and γ is the discount factor.

The learning algorithm $\mathbb{A}$ is defined: $\{s_i, a_i, s_{i+1}\}_{i=0}^{t-1} \mapsto (a_t, \pi_t)$, which maps the transitions collected until the time step t to the action a_t the agent should take in the non-episodic training and the best guess π_t of the optimal policy π^* on the evaluation goal distribution (ρ_{g^*}). In reset-free training the agent will only be reset to the initial state $s_0 \sim \rho_0$ once. The evaluation of agents is still episodic. The agent always starts from $s_0 \sim \rho_0$, and is asked to achieve $g \sim \rho_{g^*}$. The evaluation objective for a policy π is:

$$J(\pi) = \mathbb{E}_{s_0 \sim \rho_0, g \sim \rho_{g^*}, a_j \sim \pi(\cdot|s_j, g), s_{j+1} \sim p(\cdot|s_j, a_j)}[\sum_{j=0}^{T} \gamma^j r(s_j, a_j, g)], \tag{1}$$

where T is the total time steps during the evaluation. The goal of algorithm $\mathbb{A}$ during the reset-free training is to minimize the performance difference $\mathbb{D}(\mathbb{A})$ of the current policy π_t and the optimal policy π^*:

$$\mathbb{D}(\mathbb{A}) = \sum_{t=0}^{\infty} (J(\pi^*) - J(\pi_t)). \tag{2}$$

In summary, the algorithm $\mathbb{A}$ should output an action a_t that the agent should take in the non-episodic data collection and a policy π_t that can maximize $J(\pi_t)$ at every time step t based on all previously collected data.

3.2 Model-based RL setup

Recent goal-conditioned MBRL approaches like LEXA [20] and PEG [14] train goal-conditioned policies purely using synthetic data generated by learned world models. Their robust exploration demonstrate significant success in solving long-horizon goal-conditioned tasks. In the reset-free setting, strong exploration is crucial, as the agent can no longer depend on episodic resets to bring it back to task-relevant areas if it gets stuck. Therefore, we select PEG as the backbone MBRL agent for its strong exploration abilities and sample efficiency.

PEG [14] is a model-based Go-Explore framework that extends LEXA [20], an unsupervised goal-conditioned variant of DreamerV2 [11]. The following components are parameterized by θ and learned:

$$\begin{aligned} &\text{world model: } \widehat{\mathcal{T}}_\theta(s_t|s_{t-1}, a_{t-1}) && \\ &\text{goal conditioned policy: } \pi_\theta^G(a_t|s_t, g) && \text{goal conditioned value: } V_\theta^G(s_t, g) \\ &\text{exploration policy: } \pi_\theta^E(a_t|s_t) && \text{exploration value: } V_\theta^E(s_t) \end{aligned} \tag{3}$$

The world model is a recurrent state-space model (RSSM) which is trained to predict future states and is used as a learned simulator to train the policies and value functions. The goal-conditioned policy π^G is trained to reach random states sampled from the replay buffer. The exploration policy π^E is trained on an intrinsic motivation reward that rewards world model error, expressed through the variance of an ensemble [24]. Both policies are trained on simulated trajectory rollouts in the world model.

Algorithm 1 Go-Explore

```
Input: g, π_θ^G, π_θ^E
τ_g ← {}; τ_e ← {}
for t = 1 to H_G do:
    a_t ∼ π^G(· | s_t, g)
    s_{t+1} ∼ T(· | s_t, a_t)
    τ_g ← τ_g ∪ {s_t}
for t = 1 to H_E do:
    a_t ∼ π^E(· | s_t)
    s_{t+1} ∼ T(· | s_t, a_t)
    τ_g ← τ_g ∪ {s_t}
return τ_g, τ_e
```

► Self-supervised Goal-reaching Reward Function:

Rather than assuming access to the environmental reward, PEG learns its own reward function. PEG uses a dynamical distance function [13] as the reward function within world models, which predicts the number of actions between a start and goal state. The distance function is trained on random state pairs from imaginary rollouts of π^G. π^G is then trained to minimize the dynamical distance between its states and commanded goal state in imagination. See [20] for more details.

► **Phased Exploration via Go-Explore:** For data-collection, PEG employs the Go-Explore strategy. In the "Go"-phase, a goal is sampled from some goal distribution ρ. The goal-conditioned policy, conditioned on the goal is run for some time horizon H_G, resulting in trajectory τ^G. Then, in the "Explore"-phase, starting from the last state in the "Go"-phase, the exploration policy is run for H_E steps, resulting in τ^E. The interleaving of goal-conditioned behavior with exploratory behavior results in more directed exploration and informative data. This in turn improves accuracy of the world model, and the policies that train inside the world model. See Algorithm 1 and Algorithm 2 for pseudocode. The choice of goal distribution ρ is important for Go-Explore. In easier tasks, the evaluation goal distribution ρ_{g^*} may be sufficient. But in longer-horizon tasks, evaluation goals may be too hard to achieve. Instead, intermediate goals from an exploratory goal distribution ρ_E can help the agent explore. We choose PEG, which generates goals by planning through the world model to maximize exploration value (see [14] for details).

Algorithm 2 MBRL Backbone (PEG)

1: **Input:** π_θ^G, π_θ^E, world model $\widehat{\mathcal{T}}_\theta$, goal distribution ρ
2: **for** episode $i = 1$ to N **do**:
3: sample a goal $g \sim \rho$
4: $\tau_g, \tau_e \leftarrow$Go-Explore($g, \pi^G, \pi^E$)
5: $\mathcal{D} \leftarrow \mathcal{D} \cup \tau^g \cup \tau^e$
6: update $\widehat{\mathcal{T}}_\theta$ with $\mathcal{D}$
7: update π_θ^G and π_θ^E with $\widehat{\mathcal{T}}_\theta$ in imagination

4 Method

As motivated in Section 1 and Figure 1, the direct application of PEG to the reset-free setting shows promising performance but suffers from over-exploration of task-irrelevant states. To adapt model-based RL to the reset-free setting, we introduce MoReFree, a model-based approach that improves PEG to handle the lack of resets and overcome the over-exploration problem. MoReFree improves two key mechanisms of MBRL for reset-free training: exploration and policy training.

4.1 Back-and-Forth Go-Explore

First, we introduce MoReFree's procedure for collecting new datapoints in the real environment. PEG [14] already has strong goal-conditioned exploration abilities, but was developed for solving episodic tasks. Without resets, PEG's Go-Explore procedure can undesirably linger in unfamiliar but task-irrelevant portions of the state space. This generates large amounts of uninformative trajectories, which in turn degrades world model learning and policy optimization.

MoReFree overcomes this by periodically directing the agent to return to the states relevant to the task (i.e. initial and evaluation goals). We call this exploration procedure "Back-and-Forth Go-Explore", where we sample pairs of initial and evaluation goals and ask the agent to cycle back and forth between the goal pairs, periodically interspersed with exploration phases (see Figure 2 top row).

Now, we define the "Back-and-Forth Go-Explore" strategy as seen in Algorithm 3. First, we decide whether to perform initial/evaluation state directed exploration. With probability α, we sample goals (g^*, g_0) from ρ_{g^*}, ρ_0 respectively. Then, we execute the Go-Explore routine for each goal. We name Go-Explore trajectories conditioned on initial state goals as "Back" trajectories, and Go-Explore trajectories conditioned on evaluation goals as "Forward" trajectories. With probability $1 - \alpha$, we execute exploratory Go-Explore behavior by sampling exploratory goals from PEG.

Algorithm 3 Back-and-Forth Go-Explore

1: **Input:** π_θ^G, π_θ^E, world model $\widehat{\mathcal{T}}_\theta$, ρ_{g^*}, ρ_0, ρ_E
2: Generate a random number r in $[0, 1]$
3: **if** $r < \alpha$ **then**
4: $g^*, g_0 \sim \rho_{g^*}, \rho_0$
5: $\tau_{g^*}, \tau_e^1 \leftarrow$Go-Explore($g^*, \pi^G, \pi^E$)
6: $\tau_{g_0}, \tau_e^2 \leftarrow$Go-Explore($g_0, \pi^G, \pi^E$)
7: **else**
8: $g \sim \rho_E$
9: $\tau_g, \tau_e^1 \leftarrow$Go-Explore($g, \pi^G, \pi^E$)
10: **end if**

By following this exploration strategy, the agent modulates between various Go-Explore strategies, alternating between solving the task by pursuing evaluation goals, resetting the task by pursuing initial states, and exploring unfamiliar regions via exploratory goals.

4.2 Learning to Achieve Relevant Goals in Imagination

Next, we describe how MoReFree trains the goal-conditioned policy in the world model. To train π^G, MoReFree samples various types of goals and executes $\pi^G(\cdot \mid \cdot, g)$ inside the world model to generate "imaginary" trajectories. The trajectory data is scored using the learned dynamical distance reward mentioned in Section 3.2 , and the policy is updated to maximize the expected return. This procedure is called imagination [10], and allows the policy to be trained on vast amounts of synthetic trajectories to improve sample efficiency.

First, we choose to sample evaluation goals from ρ_{g^*} since the policy will be evaluated on its evaluation goal-reaching performance. Next, recall that Back-and-Forth Go-Explore procedure also samples initial states from ρ_0 as goals for the Go-phase to emulate resetting behavior. Since we would like π^G to succeed in such cases so that the task is reset, we will also sample from ρ_0. Finally, we sample random states from the replay buffer to increase π^G's ability to reach arbitrary states. The sampling probability for each goal type is set to $\alpha/2, \alpha/2, 1-\alpha$ respectively. In other words, MoReFree biases the goal-conditioned policy optimization procedure to focus on achieving task-relevant goals (i.e. evaluation and initial states), as they are used during evaluation and goal-conditioned exploration to condition the goal-reaching policy (see Figure 2 bottom row).

4.3 Implementation Details

Our work builds on the top of PEG [14], and we use its default hyperparameters for world model, policies, value functions and temporal reward function. We set the length of each phase for Go-Explore (H_G, H_E) to half the evaluation episode length for each task. We set the default value of $\alpha = 0.2$ for all tasks (never tuned). See Appendix C.3 for more details and the supplemental for MoReFree code.

5 Experiments

We evaluate two MBRL methods (PEG [14] and our extension MoReFree) and four competitive reset-free baselines on eight reset-free tasks. We aim to address the following questions: 1) Do MBRL approaches work well in reset-free tasks in terms of sample efficiency and performance? 2) What limitations arise from running MBRL in the reset-free setting, and does our proposed solution MoReFree address them? 3) What sorts of behavior do MoReFree and baselines exhibit in such tasks, and are our design choices for MoReFree justified?

Baselines: All baselines except for R3L are implemented using official codebases, see Appendix C.2 for details.

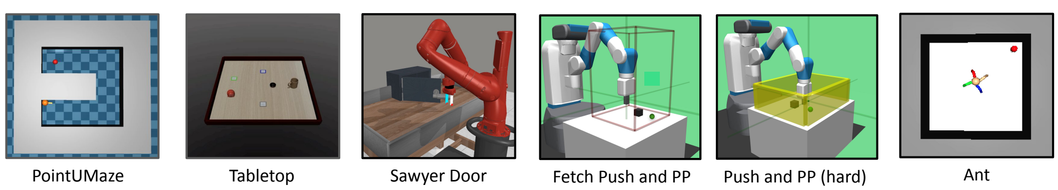

Figure 3: We evaluate MoReFree on eight reset-free tasks ranging from navigation to manipulation. PP is short for Pick&Place.

- **reset-free PEG** is a straightforward extension of PEG [14] to the reset-free setting.
- **MEDAL** [25] requires demonstrations and trains two policies, one for returning to demonstration states and another that achieves task goals.
- **IBC** [15] is a competitive baseline that outperforms prior reset-free work (e.g. MEDAL, VaPRL) by defining a bidirectional curriculum for the goal-conditioned forward and backwards (i.e. reset) policies trained using the environmental reward.
- **R3L** [35] trains two policies, one for achieving task goals and another that perturbs the agent to novel states. Notably, it is the only baseline that operates without any additional assumptions (i.e. environmental rewards, demonstrations, and resets).

• **Oracle** is SAC [9] trained under the episodic setting on the environmental reward.

Note that most baselines enjoy some advantage over two MBRL methods: MEDAL, IBC and Oracle use ground truth environmental reward, while MEDAL also uses demonstrations and Oracle uses resets. See Table 1 for a conceptual comparison between MoReFree and prior work.

Environments: We evaluate MoReFree and baselines on eight tasks (see Figure 3). We select five tasks from IBC's evaluation suite of six tasks; (Fetch Reach is omitted because it is trivially solvable). Next, we increased the complexity of the two hardest tasks from IBC, Fetch Push and Fetch Pick&Place, by extending the size of the workspace, replacing artificial workspace limits (which cause unrealistic jittering behavior near the limits, see the website for videos) with real walls, and evaluating on harder goal states (i.e. Pick&Place goals only in the air rather than including ones on the ground). In addition, we contributed a difficult locomotion task, Ant, which is adapted from the PEG codebase [14]. All methods are run with 5 seeds, and the mean performance and standard error are reported. During the evaluation, the performance on tasks with randomly sampled goals from ρ_{g^*} is measured by averaging over 10 episodes. See Appendix C for more experimental details.

5.1 Results

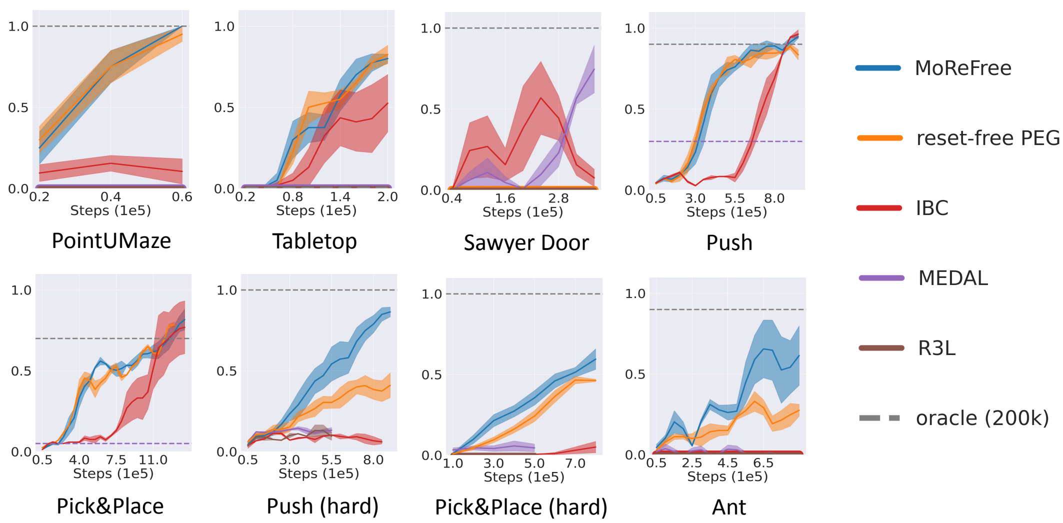

Figure 4: MBRL methods (MoReFree and reset-free PEG) significantly outperform baselines in 7/8 tasks. In 4 tasks, only MBRL methods are able to learn meaningful behavior, showcasing MBRL's sample efficiency. MoReFree outperforms PEG in the 3 most difficult tasks.

As shown in Fig 4, two model-based methods (MoReFree and reset-free PEG), without demonstrations or access to environmental reward, outperform other baselines with privileged access to supervision in both final performance and sample efficiency in 7/8 tasks. We observe that the two MBRL methods learn good behaviors: the pointmass agent hugs the wall of the UMaze to minimize travel time and the Fetch robot deftly pushes and picks up the block into multiple target locations. MoReFree is always competitive with or outperforms PEG, with large gains in the 3 hardest tasks: Push (hard) by 45%, Pick&Place (hard) by 13% and Ant (hard) by 36%. We observe that MoReFree learns non-trivial reset behaviors such as picking and pushing blocks back into the center of the table for the hard variants of the Fetch manipulation tasks. See the website for videos of MoReFree and baselines.

In many tasks, the baselines fail to learn at all. We believe this is due the low sample budget, which may be too low for the baselines to fully explore the environment and learn the proper resetting behaviors necessary to train the actual task policy. In Appendix G, we increased the training budget by $3\times$ for the IBC baseline and it still fails, underscoring the difficulty of the tasks and the sample-efficiency gains of MoReFree and MBRL. On the other hand, we noticed that one environment, Sawyer Door, seemed particularly hard for MBRL agents to solve. We hypothesize that the dynamics of the task are hard to model, resulting in performance degradation for model-based approaches (see Appendix F for more analysis).

5.2 Analysis

To explain the performance differences between MoReFree and baselines, we closely analyze the exploration behaviors.

MoReFree focuses on task-relevant states. In Figure 5 we visualize the state visitation heatmaps of methods in various environments, and also compute the percentage of "task-relevant" states (initial and goal regions, highlighted with white borders). We highlight two trends. First, the heatmaps show that MoReFree and PEG explore thoroughly while baselines have more myopic exploration patterns, as seen in the Ant heatmaps at the top.

Next, performance differences between PEG and MoReFree are intuitively explained by the amount of task-relevant data collected by each agent. In easier environments like Push or Pick&Place where both PEG and MoReFree encounter similar amounts of task-relevant states, the performance is roughly similar between PEG and MoReFree. But in harder environments (Ant, Push (hard), Pick&Place (hard)) with larger state spaces and more complicated resetting dynamics, MoReFree collects $1.3-5\times$ more task-relevant data and has large performance gains over PEG. By experiencing more task-relevant states and training policies on them in imagination, MoReFree policies are more suited towards succeeding at the episodic evaluation criteria. See Appendix D for additional visualizations.

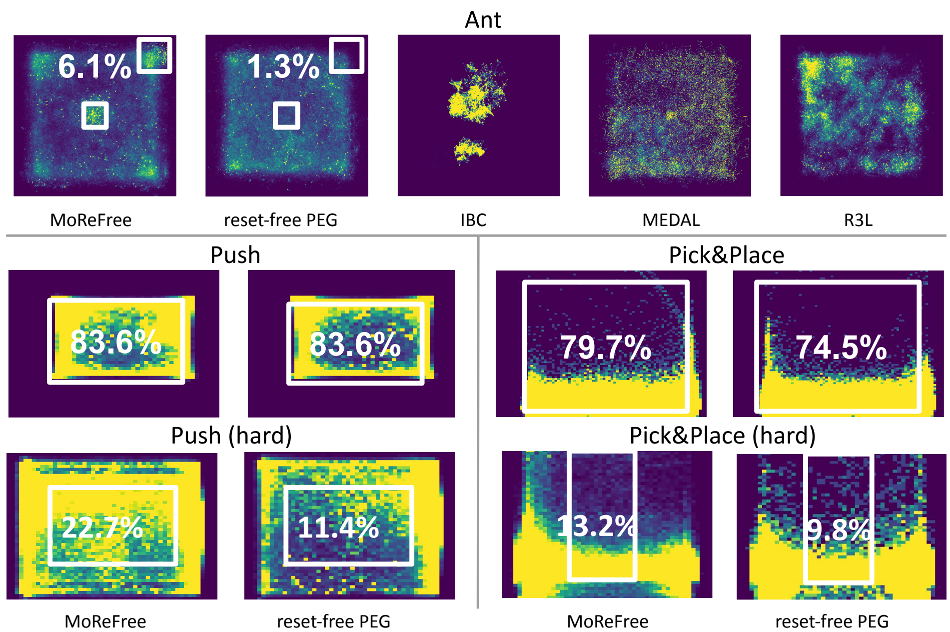

Figure 5: State visitation heatmaps of different agents. White areas are task-relevant states (including initial and goal state distributions) and we overlay the percentages of task-relevant states. MBRL methods explore more and in harder environments, MoReFree experiences more task-relevant states.

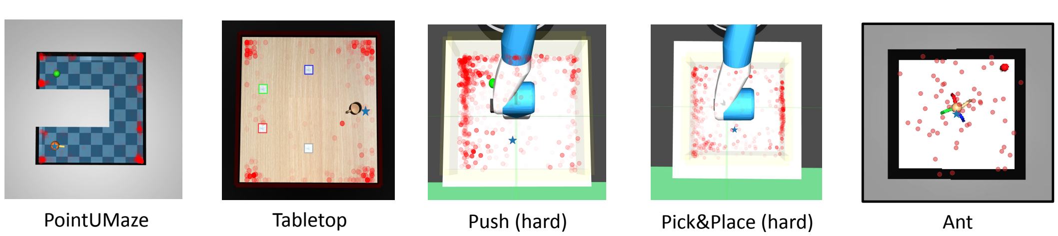

Figure 6: We visualize the start position (red dots) of successful "Back" trajectories of MoReFree's Back-and-Forth Go-Explore, where π^G is directed to reset the environment. The color intensity of the dots correspond to state density over the last 100 steps.

MoReFree effectively resets. Next, we investigate the qualitative behavior of MoReFree's Back-and-Forth Go-Explore. To see if "Back" trajectories help free the agent from the sink states, we analyze the replay buffer of MoReFree for the environments, and plot the starting locations of the agent / object up to 100 timesteps before a successful "Back" trajectory is executed in Figure 6. The color intensity of the dots correspond to state density over the last 100 steps (i.e. dark red means the agent / object has rested there for a while). We observe that the starting locations (red dots) of the agent / object are in corners or next to walls in all environments. This suggests that these areas act as sink states, where the agent / object would remain for long and waste time. We observe that MoReFree learns reset behaviors like picking the block out of corners and walls in Fetch Push and Fetch Pick&Place. See detailed videos of the reset behavior on the website.

5.3 Ablations

To justify our design choices, we ablate the two mechanisms of MoReFree, the back-and-forth exploration and goal-conditioned policy training, and plot the results in Figure 7 First, removing all mechanisms (**MF w/o Explore & Imag.**) reduces to PEG, and we can see a large gap in performance. Next, **MF with Only Task Goals** sets $\alpha = 1$, which causes an extreme bias towards task-relevant states in the exploration and policy training. This also degrades performance, due to the need for strong exploration in the reset-free setting. Examinations of more values for α can be found in Appendix C.3.

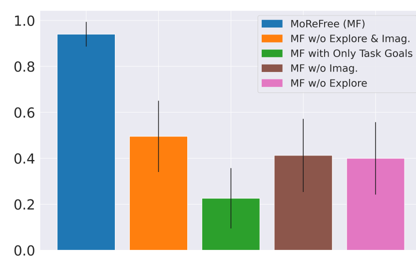

Figure 7: Ablations on 5 variants of MoReFree over 5 environments with normalized final performance.

Finally, we isolate individual components of MoReFree. First, we disable Back-and-Forth Go-Explore by disallowing the sampling of initial or evaluation goals during Go-Explore. Only exploratory goals are used in Go-Explore for this ablation (named **MF w/o BF-GE**). Next, in **MF w/o Imag.** we turn off the initial / evaluation goal sampling in imagination, so only random replay buffer goals are used to train π^G. We see that both variants perform poorly. This is somewhat intuitive, as the two components rely on each other. Having both forms a synergistic cycle where 1) the goal-conditioned policy's optimization is more focused towards reaching initial / goal states, and 2) the exploration is biased towards reaching initial / goal states by using the goal-conditioned policy we just optimized in step 1. If we remove one without the other, then the cycle breaks down. In **MF w/o Imag.**, Back-and-Forth Go-Explore will suffer since π^G trained on random goals cannot reliably reach initial / evaluation goals. In **MF w/o BF-GE**, the exploration strategy will not seek initial / evaluation states, resulting in an inaccurate world model and degraded policy optimization. In summary, the ablations show that MoReFree's design is sound and is the major factor behind its success in the reset-free setting. See Appendix E for details.

6 Conclusion and Future Work

As a step towards reset-free training, we adapt model-based methods to the reset-free setting and demonstrate their superior performance. Specifically, we show that the out-of-the-box, unsupervised MBRL method substantially outperforms the state-of-the-art model-free baselines tailored for the reset-free setting while being more autonomous (requires less supervision like environmental reward or demonstrations). We then identify a limitation of unsupervised MBRL in the reset-free setting (over-exploration on task-irrelevant states), and propose MoReFree to address such limitations by focusing model-based exploration and goal-conditioned policy training on task-relevant states. We conduct a through experimental study of MoReFree and baselines over 8 tasks, and show considerable performance gains over the MBRL baseline and prior state-of-the-art reset-free methods. Despite its overall success, MoReFree is not without limitations. Being a model-based approach, it inherits all associated disadvantages. For example, we believe Sawyer Door is a task where learning the dynamics is harder than learning the policy (see Appendix F), disadvantaging MBRL approaches. Next, MoReFree uses a fixed percentage of task-relevant goals for exploration and imagination, whereas future work could consider an adaptive curriculum. Finally, scaling MoReFree to high-dimensional observations would be a natural extension. We hope MoReFree inspires future efforts in increasing autonomy in RL.

References

[1] A. Baranes and P.-Y. Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73, 2013.

[2] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[3] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

[4] B. Eysenbach, S. Gu, J. Ibarz, and S. Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. *arXiv preprint arXiv:1711.06782*, 2017.

[5] B. Eysenbach, R. Salakhutdinov, and S. Levine. C-learning: Learning to achieve goals via recursive classification. In *International Conference on Learning Representations*, 2021.

[6] C. Florensa, D. Held, X. Geng, and P. Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528. PMLR, 2018.

[7] J. Fu, A. Singh, D. Ghosh, L. Yang, and S. Levine. Variational inverse control with events: A general framework for data-driven reward definition. *Advances in neural information processing systems*, 31, 2018.

[8] A. Gupta, J. Yu, T. Z. Zhao, V. Kumar, A. Rovinsky, K. Xu, T. Devlin, and S. Levine. Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6664–6671. IEEE, 2021.

[9] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[10] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[11] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[12] S. Haldar, J. Pari, A. Rai, and L. Pinto. Teach a robot to fish: Versatile imitation from one minute of demonstrations. *arXiv preprint arXiv:2303.01497*, 2023.

[13] K. Hartikainen, X. Geng, T. Haarnoja, and S. Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. *arXiv preprint arXiv:1907.08225*, 2019.

[14] E. S. Hu, R. Chang, O. Rybkin, and D. Jayaraman. Planning goals for exploration. In *The Eleventh International Conference on Learning Representations*, 2023.

[15] J. Kim, D. Cho, and H. J. Kim. Demonstration-free autonomous reinforcement learning via implicit and bidirectional curriculum. In *International Conference on Machine Learning*. PMLR, 2023.

[16] J. Kim, J. hyeon Park, D. Cho, and H. J. Kim. Automating reinforcement learning with example-based resets. *IEEE Robotics and Automation Letters*, 7(3):6606–6613, 2022.

[17] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[18] K. Lu, A. Grover, P. Abbeel, and I. Mordatch. Reset-free lifelong learning with skill-space planning. *arXiv preprint arXiv:2012.03548*, 2020.

[19] K. Lu, I. Mordatch, and P. Abbeel. Adaptive online planning for continual lifelong learning, 2020.

[20] R. Mendonca, O. Rybkin, K. Daniilidis, D. Hafner, and D. Pathak. Discovering and achieving goals via world models, 2021.

[21] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112. PMLR, 2020.

[22] S. Pitis, H. Chan, S. Zhao, B. Stadie, and J. Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *International Conference on Machine Learning*, pages 7750–7761. PMLR, 2020.

[23] V. H. Pong, M. Dalal, S. Lin, A. Nair, S. Bahl, and S. Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

[24] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak. Planning to explore via self-supervised world models. In *ICML*, 2020.

[25] A. Sharma, R. Ahmad, and C. Finn. A state-distribution matching approach to non-episodic reinforcement learning. *arXiv preprint arXiv:2205.05212*, 2022.

[26] A. Sharma, A. M. Ahmed, R. Ahmad, and C. Finn. Self-improving robots: End-to-end autonomous visuomotor reinforcement learning. *arXiv preprint arXiv:2303.01488*, 2023.

[27] A. Sharma, A. Gupta, S. Levine, K. Hausman, and C. Finn. Autonomous reinforcement learning via subgoal curricula. *Advances in Neural Information Processing Systems*, 34:18474–18486, 2021.

[28] A. Sharma, K. Xu, N. Sardana, A. Gupta, K. Hausman, S. Levine, and C. Finn. Autonomous reinforcement learning: Formalism and benchmarking. *arXiv preprint arXiv:2112.09605*, 2021.

[29] L. Smith, N. Dhawan, M. Zhang, P. Abbeel, and S. Levine. Avid: Learning multi-stage tasks via pixel-level translation of human videos. *arXiv preprint arXiv:1912.04443*, 2019.

[30] V. Veeriah, J. Oh, and S. Singh. Many-goals reinforcement learning. *ArXiv*, abs/1806.09605, 2018.

[31] K. Xu, S. Verma, C. Finn, and S. Levine. Continual learning of control primitives: Skill discovery via reset-games. *Advances in Neural Information Processing Systems*, 33:4999–5010, 2020.

[32] A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, and S. Levine. Collective robot reinforcement learning with distributed asynchronous guided policy search. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 79–86. IEEE, 2017.

[33] Y. Zhang, P. Abbeel, and L. Pinto. Automatic curriculum learning through value disagreement. *Advances in Neural Information Processing Systems*, 33:7648–7659, 2020.

[34] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3651–3657. IEEE, 2019.

[35] H. Zhu, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, and S. Levine. The ingredients of real-world robotic reinforcement learning. *arXiv preprint arXiv:2004.12570*, 2020.

A Broader Impacts

As we increase the autonomy of RL agents, the possibility of them acting in unexpected ways to maximize reward increases. The unsupervised exploration coupled alongside the learned reward functions further add to the unpredictability; neither mechanisms are very interpretable. As such, we expect research into value alignment, interpretability, and safety to be paramount as autonomy in RL improves.

B Extended Related Work

Learned Reward Functions: Instead of requiring the environment to provide a reward function, the agent can learn its own reward function from onboard sensors and data. Given human specified example states, e.g. a goal image, VICE and C-Learning train reward classifiers over examples [7, 5] and agent data. The learned dynamical distance function [13] learns to predict the number of actions between pairs of states. The dynamical distance function is used by unsupervised MBRL approaches like LEXA and PEG [20, 14] to train the goal-conditioned policy.

C Experimental Details

C.1 Environments

PointUMaze: The state space is 7D and the action space is 2D. The initial state is $(0, 0)$, which located in the bottom-left corner, and noise sampled from $\mathcal{U}(-0.1, 0.1)$ is added when reset. The goal during the evaluation is always located in at the top-left corner of the U-shape maze. The maximum steps during the evaluation is 100. Hard reset will happen after every $2e5$ steps. In the whole training process we performed, it only reset once at the beginning of the training. Taken from the IBC [15] paper.

Tabletop: The state space is 6D, and the action space is 3D. During the evaluation, four goal locations are sampled in turn, the initial state of the agent is always fixed and located in the center of the table. The maximum steps during the evaluation is 200. Hard reset will happens after every $2e5$ steps. In the whole training process we performed, it only reset once at the beginning of the training. Taken from the EARL [28] benchmark and also used in the IBC paper.

Sawyer Door: The state space is 7D and the action space is 4D. The position of door is initialized to open state (60 degree with noise sampled from $(0, 18)$ degree) and the goal is always to close the door (0 degree). The arm is initialized to a fixed location. Maximum number of steps is 300 for the evaluation. Hard reset will happen after every $2e5$ steps. In the whole training process we performed, it resets twice. Taken from the EARL [28] benchmark and also used in the IBC paper.

Fetch Push and Pick&Place: The state space is 25D and action space is 4D. These are taken from the IBC paper. Authors converted the original Fetch environments to a reversible setting by defining a constraint on the block position. The initial and goal distributions are identical to the original Fetch Push and Pick&Place. More details can be found in the IBC paper.

Push (hard): Different from the original Fetch Push task, in our case walls are added to prevent the block from dropping out of the table. The workspace of the robot arm is also limited. The block is always initialized to a fixed location, and goal distribution during the evaluation is $\mathcal{U}(-0.15, 15)$. Fetch Push used in the IBC paper, the block is limited by joint constraint, which shows unrealistic jittering behaviors near the limits (we observe such phenomenon by running model-based go-explore, the exploration policy prefers to always interact with the block and keep pushing it towards the limit boundary, see videos on our project website [1]). Meanwhile, the gripper is blocked, which makes the task easier. In our case, we release the gripper and it can now open and close again which add two more dimension of the state space. We found it is important to release the gripper in our version of Push task, when the block is in corners, it will need to operate the gripper to drag the block escape from corners. The maximum steps the agent can take in 50 during the evaluation. Hard reset will happen after every $1e5$ steps. In the whole training process we performed, it resets 5 times in total.

[1] `https://sites.google.com/view/morefree`

Pick&Place (hard): We add walls in the same way as we did for Push (hard). We make it more difficult by only evaluating the agent on goals that are in the air. Then it has to learn to perform picking behavior properly, whereas goals on the ground can just be solved by pushing. The goal will be uniformly sampled from a $5 \times 5 \times 10$ cm cubic area above the table. It has the same observation space, action space, initial state and maximum steps with Fetch Push described above. Hard reset will happens after every $1e5$ steps. In the whole training process we performed, it resets 5 times in total. See the visual difference between our Pick&Place and IBC's in Figure 3. Since the workspace of the robot is limited within the walls as well in Push (hard) and Pick&Place (hard), when the block gets stuck in corners, the robot needs to precisely move to the corner and bring the block back. In contrast, the robot in IBC's version can move to everywhere, being able to create various circumstance to solve such difficult position.

Ant: We adapt the AntMaze task from environments[2] codebase of PEG and change the shape of the maze to square, also change the evaluation goal distribution to be a uniform distribution $\mathcal{U}(2, 3)$ for both x and y location, which lies on the top-left corner of the square. The ant is always initialized to the center point (0, 0) of the square to start from, with uniform noise ($\mathcal{U}(-0.1, 0.1)$) added. The state space is 29D and the action space is 8D. The maximum steps for evaluation is 500. Hard reset will happen after every $2e5$ steps. In the whole training process we performed, it reset 4 times in total.

C.2 Baseline Implementations

reset-free PEG: We extend the official implementation of PEG[3] to reset-free setting by 1) removing the reset of environments; 2) optimizing the goal distribution every $H_G + H_E$ steps; 3) keeping all other hyperparameters the same as MoReFree.

IBC: We use the official implementation from authors[4] and keep hyperparameters unchanged.

MEDAL: We follow the official implementation of MEDAL[5] and use the deafult setting for experiments. Since MEDAL requires demonstrations, for tasks from EARL benchmark, demonstrations are provided. For other environments, we generate demonstrations by executing the final trained MoReFree to collect data. 30 episodes are generated for each task.

R3L: We implement R3L agent by modifying the FBRL agent from MEDAL codebase. The backward policy is replaced by an exploration policy trained using the random network distillation (RND) objective [2]. The RND implementation we follow is from DI-engine[6].

Oracle: This is a episodic SAC agent, we use the implementation from MEDAL codebase and keep all the hyper-parameters unchanged.

MoReFree: Our agent is built on the model-based go-explore method PEG [14], we extend their codebase by adding back-and-forth goal sampling procedure and training on evaluation initial and goal states in imagination goal-conditioned policy training. See our codebase in the supplemental.

C.3 Hyperparameters

Train ratio (i.e. Update to Data ratio) is an important hyper-parameter in MBRL. It controls how frequently the agent is trained. Every n steps, a batch of data is sampled from the replay buffer, the world model is trained on the batch, and then policies and value functions are trained in imagination. In all our experiments, we only vary n on different tasks. See the table below for different values on different tasks we used through experiments. MoReFree also introduces a new parameter α, which we keep $\alpha = 0.2$ for all tasks and did not tune it at all. All other hyperparameters we keep the same as the original code base.

Different values for α. We examine different values of α in MoReFree on Fetch Push task, which affects how much MoReFree focuses on task-relevant goals in exploration and imagination. In Figure 8, we see that introducing a moderate amount of task-relevant goals (α=0.2, α=0.5) results in

[2] `https://github.com/edwhu/mrl`
[3] `https://github.com/penn-pal-lab/peg`
[4] `https://github.com/snu-larr/ibc_official`
[5] `https://github.com/architsharma97/medal`
[6] `https://opendilab.github.io/DI-engine/12_policies/rnd.html`

Table 2: Different train ratio we used for different tasks. We keep all other hyperparameters the same as default ones.

PointUMaze	2	Sawyer Door	5	Tabletop	1	Fetch Push	2
Fetch Pick&Place	2	Push (hard)	2	Pick&Place (hard)	2	Ant	2

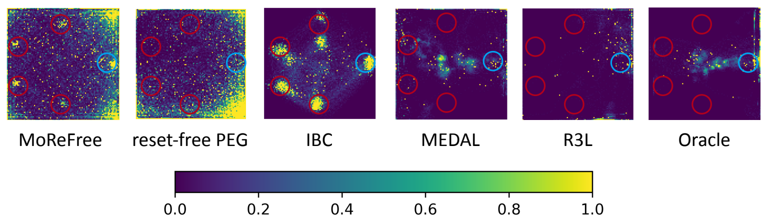

Figure 9: XY state visitation heatmap of the mug in Tabletop of various approaches. MoReFree's heatmap shows high state diversity while retaining high visitation counts near the task-relevant states (red circles are goal states, the blue circle is the initial state). reset-free PEG also shows diverse exploration, but it over-explores the bottom-right corner which is entirely task-irrelevant. IBC's bi-directional curriculum leads the exploration shuttles between the initial state and goal states, but fails to explore well. All other methods fail to explore, visited states mostly cluster in few spots.

sensible performance, while too many task-relevant goals (α=0.7, α=1.0) degrades performance. We use the same value of alpha, 0.2, across all tasks, which showcases MoReFree 's consistency.

C.4 Results Clarification

In Push and Pick&Place results, we retrieved the final performance of MEDAL directly from the IBC paper (dashed purple lines) and did not have time to run R3L in these two environments. R3L is shown to be a lot worse than MEDAL in the MEDAL paper and performs obviously bad in other tasks shown in Figure 4. In Push (hard) and Pick&Place (hard), we ran R3L and MEDAL with less budget since other methods clearly outperform and their learning curves do not show any evidence for going up.

C.5 Resource Usage

We submit jobs on a cluster with Nvidia 2080, 3090 and A100 GPUs. Our model-based experiments take 1-2 days to finish, and the model-free baselines take half day to one day to run.

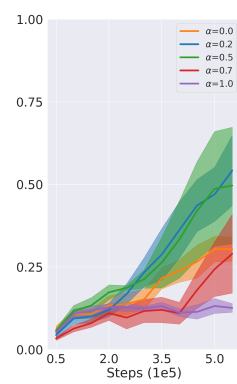

Figure 8: Performance of MoReFree with different values of α in Push (hard).

D More Visualizations on Replay Buffer

We visualize the replay buffer of different agents on more tasks. See Figure 9 for XY location of the mug in Tabletop, Figure 11 for XY location data of the agent in PointUMaze, Figure 10 for XZ location of the block in Pick&Place (hard) and Figure 12 for XY location data of the block in Push (hard) and Pick&Place (hard). Overall, we see MoReFree explores the whole state space better. Meanwhile, due to back-and-forth procedure, MoReFree collects more data near initial / goal states, which are important for the evaluation. However, IBC, MEDAL, R3L and Oracle all fail to explore well; their heatmaps are mostly populated with low visitation cells.

E Detailed Ablations

We report learning curves for each variant agent we ablate in Section 5.3 on every task in Figure 13. Since MoReFree does not learn at all in Saywer Door task, we exclude the ablation for it. In each

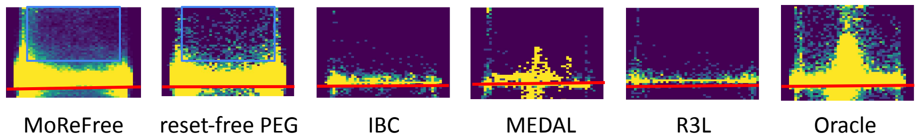

Figure 10: XZ state visitation heatmap of the block in Pick&Place (hard). States above the red line are in the air, which are crucial for solving the picking task. Two MBRL methods collect more data diversely in the air, while other reset-free methods barely pick up the block.

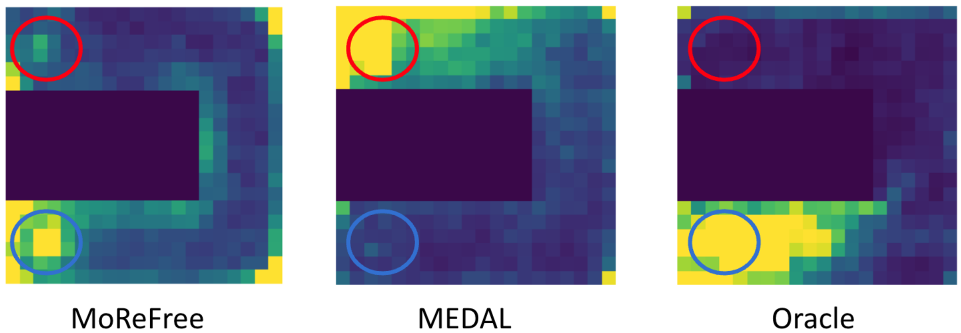

Figure 11: State visitation heatmap on point maze. MoReFree has special focuses on both initial state (blue circles) corner and goal state (red circles), while explore much uniformly. MEDAL collects lots of data near the goal state and little data on the initial state. Both MEDAL and Oracle explore less extensively.

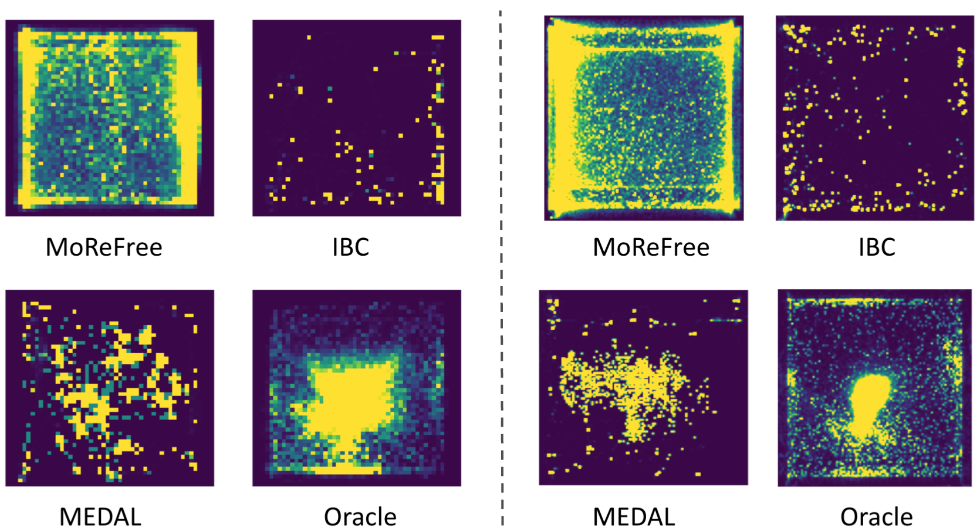

Figure 12: Block state visitation heatmap on Fetch Push (left) and Fetch Pick&Place (right) of different agents. MoReFree better explores the whole state space, while IBC and MEDAL do not have too much interactions with the block, thus lighted areas are scattered everywhere.

task, MoReFree is better or on par with all other ablations. Through learning curves, we see different components contribute differently on different tasks.

We further analyze the ablation on PointUMaze as an example by visualizing the replay buffer of different variants, see Figure 14. In the performance on PointUMaze from Figure 13, sampling exploratory goals for data collection is important (MF w/o Explore & Imag. outperforms other ablations). But we see in 14, MF w/o Explore & Imag. does not have focus on the initial / goal state which we care about for the evaluation, which makes it slightly worse than MoReFree. MF with Only Task Goals has a strong preference on initial / goal state, we think it is because in the later phase of

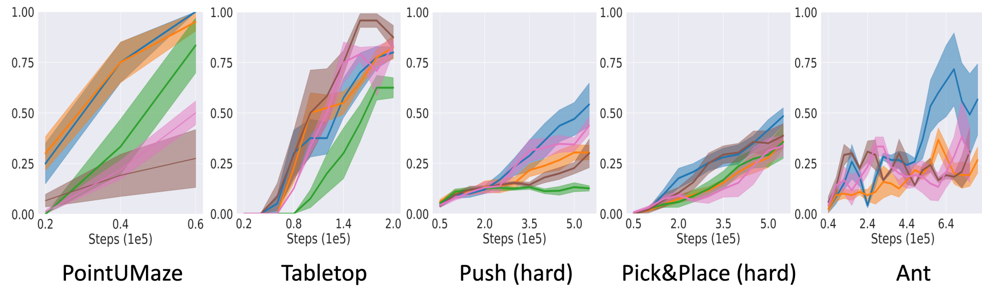

Figure 13: Learning curves of ablation study on 5 tasks. We see different components contribute differently in different tasks. For instance, in Tabletop, **MF w/o Imag.** even performs better than MoReFree, maybe because the whole state space can be explored quickly, then randomly sampling states from the replay buffer as goals for training already has good coverage on evaluation initial / goal states.

the training when the agent is able to solve the task, it goes back-and-forth consistently to collect data. But in the early phase of the training, it might lack exploration which causes the degraded performance compare with MoReFree. MF w/o Explore and MF w/o Imag. only either go to initial / goal state for data collection and do not practice on it during the imagination training, or practice without really going, which both does not form the positive cycle, and end up with poor performance.

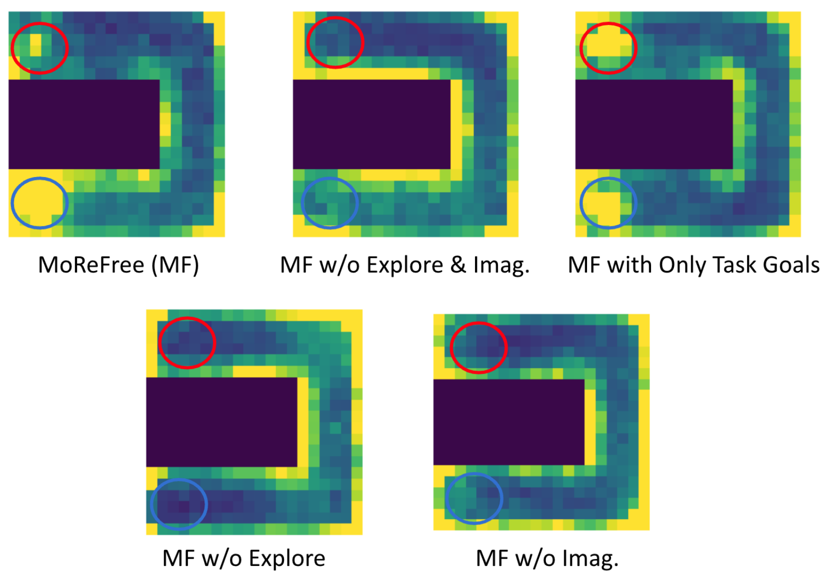

Figure 14: State visitation heatmap on PointUMaze task of all ablations. Red circles are evaluation goal states and blues are initial states. We see MoReFree collect good amount of data near initial / goal states while stronger exploration. MF w/o Explore and MF w/o Imag. could not gather task-relative data, which further causes poor performance.

F MBRL on Sawyer Door

We investigate why two MBRL methods fail on Sawyer Door tasks. Note that MoReFree is able to solve intermediate goals such as closing the door in some angles, but is unable to solve the original IBC evaluation goal (see website for more videos).

We simplify Sawyer Door task by limiting the movement range of the robot to a box and also having a block holds the door to prevent it from opening it too much, see Figure 15. Although MBRL methods are trained on the simplified environment, we see learning curves on Sawyer Door are completely flat in Figure 4, compared with other baselines trained on the original task. We wonder why MBRL methods can show the same performance and gain benefits as it does in other environments.

MoReFree and reset-free PEG use DreamerV2 as backbone agents and extend it to reset-free settings. We hypothesize that Dreamer itself, even under the episodic setting with task reward function, would not work well. If that's the case, then MBRL methods in the reset-free setting with self-supervised reward function would almost certainly not work either. For example, if the backbone agent cannot model the dynamics precisely, then policy learning, dynamical distance reward learning, will be degraded.

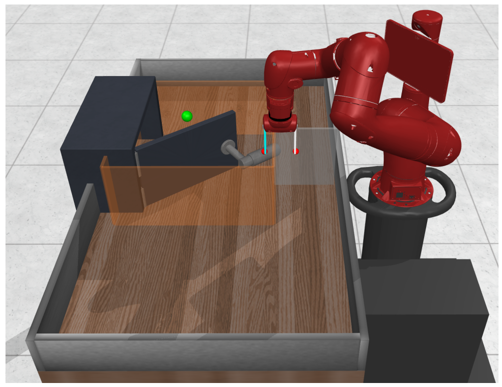

Figure 15: Simplified version of Sawyer Door. Orange walls show the limited workspace for the robot arm, and a grey wall is added to limit the movement of the door. The door can only move to maximum 60 degrees.

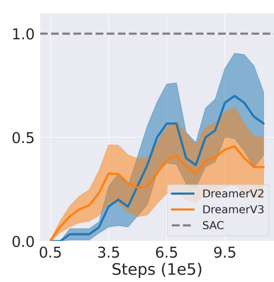

Figure 16: Performance of DreamerV2 and V3 on episodic Sawyer Door task. SAC can solve the task in 200k steps, while after 1 million steps MBRL is still not able to steadily solve the task.

We then run the underlying MBRL backbones under the episodic setting. Figure 16 shows DreamerV2 [7], and Dreamerv3 [8] struggle to solve the task, while model-free method SAC can steadily solve the task after 200k steps. This might be a potential reason that MBRL methods do not work on the more difficult reset-free setting. We hypothesize that the combination of the sparse environmental reward and dynamics of the door result in a hard prediction problem for world modelling approaches. We leave further investigation for the future work.

G More Analysis on Fetch Environments

Although IBC gains good final performance in Push and Pick&Place, it starts learning late compared with MBRL methods and fails entirely in our harder versions. We suspect IBC might need more computational budget to start learning in harder tasks. Thus we train IBC with two millions environment steps and results in Figure 17 show that it still fails to solve the harder version of Push.

[7] `https://github.com/danijar/dreamerv2`

[8] `https://github.com/danijar/dreamerv3`

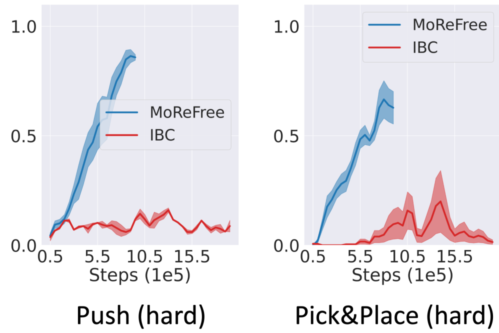

Figure 17: Longer training of IBC in our Fetch tasks, where the state space is larger and artificial constraints are replaced with surrounded walls. IBC still can not learn meaningful behaviors.

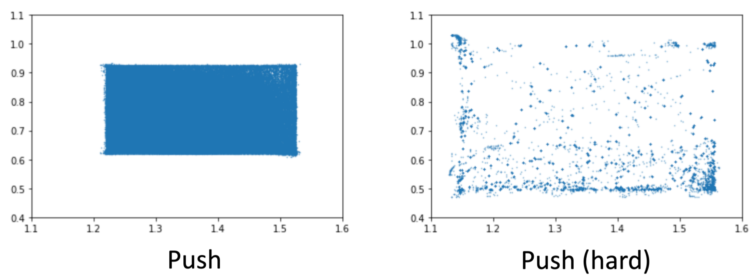

Figure 18: XY location of the block collected by IBC on Push (hard) and its original version (Push). IBC covers the whole state space very well in Push while fails in Push (hard), where the block stays for long time in corners or areas next to walls.

Figure 18 shows 600k data of the obejct (XY view) collected by IBC on our Push (hard) and IBC's Push. We see the block stays in corners or next to walls a lot in Push (hard), while goes everywhere and covers the whole space in IBC's Push, indicating object interaction is more difficult in Push (hard) due to the larger state space, surrounded walls and limited work space. In IBC's Push, the block can bounce back when it hits the limit of joint constraints. However, in Push (hard), the block needs to be explicitly brought back from the corner or walls, requiring more sophisticated behaviors. Meanwhile, larger size of the limited area (our version is $3\times$ larger than IBC's.) also increases the difficulty of the task.

H Analysis on R3L

R3L trains two policies, one for reaching the goal and another that brings the agent to novel states. The goal-reaching policy is trained using a learned classifier to classify the goal state and other states. Original R3L takes images as inputs, thus the trained classifier can successfully classify goal images from random state images. In our work, we use low-dimensional state input. Outputs of the trained classifier on the whole state space of PointUMaze is shown in Figure 19. We see that the classifier learns to output higher values for states close to the goal state (red dot) and lower values for states further away. Nonetheless, due to the smoothness of the output scope, states near the initial state (blue circle) that are numerically closer but spatially further to the goal state also have higher values. R3L agent trained using such reward function will always tend to follow states with higher values to the corner instead of going forward. See the website for more videos. These trained reward functions are misleading for learning reasonable policies which result in poor performance we see in Figure 4.

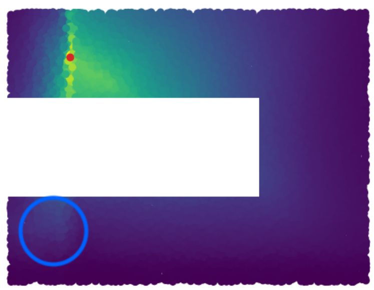

Figure 19: Outputs of the learned classifier on the whole state space. Due to the smoothness of the output scope, states near the initial state (blue circle) also have higher values.