World Models Increase Autonomy in Reinforcement Learning

Anonymous Authors[1]

Abstract

Reinforcement learning 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 provide forms of supervision (e.g. rewards / demonstrations) or resets to the agent. Ideal RL methods would eliminate the need for rewards, demonstrations and resets, facilitating fully autonomous training. We refer to this minimal training setting as the Demo-free, Reward-free, Reset-free RL (DR3L) paradigm. Unsurprisingly, RL with such limited assumptions presents significant challenges. To tackle the challenging DR3L setting, we propose a **Mo**del-based, **Re**set-**Free** (MoReFree) framework. It adapts two key mechanisms, exploration and policy learning, to handle DR3L tasks by prioritizing task-relevant states. MoReFree exhibits superior data-efficiency across various reset-free tasks without access to rewards and demonstrations while significantly outperforming privileged baselines that require supervision. 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 presence of two scaffolding mechanisms that can require significant human effort to setup: supervision in the form of rewards (or demonstrations), and environmental resets. These assumptions pose a significant barrier for real world applications of RL like robotics.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Most RL systems on real robots to date have employed various strategies to implement resets and rewards, all requiring a considerable amount of effort (Levine et al., 2016; Yahya et al., 2017; Zhu et al., 2019; Nagabandi et al., 2020). In Nagabandi et al. (2020), 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. For rewards, one notable example is Schenck & Fox (2018); they used expensive thermal cameras to measure fluid levels to train a vision-based pouring policy.

In summary, to implement RL in real world scenarios, a practitioner may have to add privileged sensors to estimate state for rewards, modify the environment to facilitate resets, and script reset controllers (potentially on another robot). These illustrate that even for simple behaviors, proper implementation of reward and reset mechanisms can result in significant human effort and time.

To enhance autonomy and reduce human burden during RL training, multiple lines of work have been proposed to remove the reliance on human-specified rewards, demonstrations, and reset functions. For instance, instead of requiring humans to set up additional privileged sensors for reward estimation, the agent can learn its own reward function from onboard sensors (Fu et al., 2018; Hartikainen et al., 2019; Eysenbach et al., 2021; Ma et al., 2022; Haldar et al., 2023) Similarly, rather than necessitating human-engineered reset mechanisms, the agent can operate within a reset-free training scheme, learning to reset itself (Eysenbach et al., 2017; Sharma et al., 2021a; 2023; Kim et al., 2023) or learning a policy with a diverse starting state distribution (Zhu et al., 2020).

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. Therefore, having an exploration strategy that balances between exploration of unseen regions and practicing optimal behavior in task-relevant regions is essential.

While these approaches aim to increase RL training autonomy by eliminating supervision or resets, few work as-

sume the absence of both. Methods that attempt to remove one assumption typically depend on another to compensate. For example, most reset-free approaches rely on human-supervised guidance, while most unsupervised RL methods assume episodic resets. This observation prompts the question, **Can we train an RL agent without requiring supervision or resets?**

We call the scenario, the **D**emo-free, **R**eward-free, **R**eset-free **RL** setting (DR3L). Recent work (Mendonca et al., 2021; Hu et al., 2023) in (episodic) unsupervised goal-conditioned model-based RL has shown promising results in training goal-conditioned policies without rewards or demonstrations. We build upon this work, extending it to the reset-free setting, resulting in the **Mo**del-based, **Re**set-**Free** (MoReFree) framework (see Figure 1). MoReFree improves two key mechanisms, exploration and goal-conditioned policy optimization, to enable non-episodic training.

Following the top row of Figure 1: to gather task-relevant data without resets, we define a training curriculum that alternates between temporally phases of extended task solving, resetting, and exploration. Next, as seen in the bottom row of Figure 1, we bias goal-conditioned policy training within the world model to achieve task-relevant goals such as reaching initial states and evaluation states.

Our key contributions are: **(1)** MoReFree, a novel model-based framework that focuses exploration and goal-conditioned policy training on task-relevant states to handle the challenging Demo-free, Reward-free, Reset-free RL (DR3L) setting. **(2)** We compare MoReFree against baselines on six challenging non-episodic, continuous control tasks ranging from pick-and-place manipulation to multi-legged ant locomotion. MoReFree notably outperforms baselines with privileged access to reward and demonstrations in final performance and data-efficiency in 5/6 tasks, highlighting the effectiveness of the model-based approach. **(3)** We conduct extensive analysis of MoReFree and baselines' exploration behavior. We find that MoReFree's exploration strategy explores the state space throughly while retaining high visitation counts around task-relevant states. Our ablations show that the performance gains of MoReFree are coming from the proposed design choices and therefore justify the overall approach.

2. Related Work

We give an overview of related lines of work that aim to remove human-engineered rewards and resets to reduce human supervision over the RL training process.

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

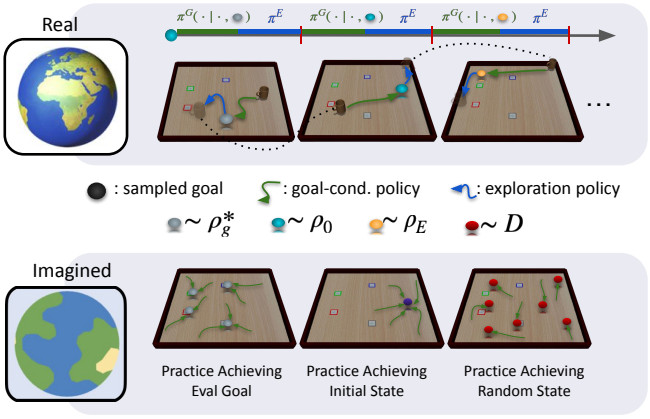

Figure 1. MoReFree is a model-based RL agent for solving reset-free tasks. **Top row:** MoReFree strikes a balance between exploring unseen states and practicing optimal behavior in task-relevant regions by directing the goal-conditioned policy to achieve evaluation states, initial state states (emulating 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.

image, VICE and C-Learning train reward classifiers over examples (Fu et al., 2018; Eysenbach et al., 2021) and agent data. The learned dynamical distance function (Hartikainen et al., 2019) 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 (Mendonca et al., 2021; Hu et al., 2023) to train the goal-conditioned policy.

Reset-free RL: There is a growing interest in researching reinforcement learning methods that can effectively address the complexities of reset-free training. Sharma et al. (2021b) proposes a reset-free RL benchmark (EARL) and finds that standard RL methods like SAC (Haarnoja et al., 2018) 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 (Eysenbach et al., 2017; Kim et al., 2022; Sharma et al., 2021a; 2022; Kim et al., 2023). LNT (Eysenbach et al., 2017) and Kim et al. (2022) train a reset policy to bring the agent back to initial state distribution, supervised by dense rewards and demonstrations respectively. MEDAL (Sharma et al., 2022; 2023), train a goal-conditioned reset policy and direct it to reset goal states from demonstrations. IBC (Kim et al., 2023) defines a curriculum for both task and reset policies without requiring demonstrations. VaPRL (Sharma et al., 2021a) 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 (Zhu et al., 2020) and

Xu et al. (2020) 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 limited to tasks where exploration is unchallenging. The vast majority of reset-free approaches are model-free, with a few exceptions (Lu et al., 2020b;a). Other works (Gupta et al., 2021; Smith et al., 2019) 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 (Baranes & Oudeyer, 2013; Veeriah et al., 2018), intermediate difficulty (Florensa et al., 2018), value disagreement (Zhang et al., 2020), state novelty (Pong et al., 2019; Pitis et al., 2020), world model error (Hu et al., 2023; Sekar et al., 2020), and more. Many goal-conditioned exploration methods use the "Go-Explore" (Ecoffet et al., 2021) 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 (Hu et al., 2023), 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. Goal-conditioned exploration methods frequently outperform non-goal-directed counterparts, due to their structured exploration behavior. However, such methods are not designed for the reset-free RL setting, and may suffer from *over-exploration* of task-irrelevant states.

Approach	Need Demo	Need Reward	Need Reset	Model-based
PEG	✗	✗	✓	✓
LiSP	✗	✓	✗	✓
MEDAL	✓	✓	✗	✗
IBC	✗	✓	✗	✗
R3L	✗	✗	✗	✗
MoReFree	✗	✗	✗	✓

Table 1. A conceptual overview of related methods. Most methods require some combination of demos, rewards, or resets to work. In terms of assumptions, only R3L matches MoReFree– neither requires demos, rewards, or resets. In performance, MoReFree significantly improves over privileged baselines IBC and MEDAL, which in turn outperform R3L in their experiments.

We notice that the majority of all prior work that aim to eliminate one form of assumption (e.g. reward) still depend on another (e.g. resets). R3L is a notable exception, and proposes a reward-free, reset-free RL approach for training robots. R3L's model-free approach is known to suffer from poor sample efficiency and exploration issues, limiting its usefulness and generality. In contrast, MoReFree uses world models to efficiently train policies and perform non-trivial goal-conditioned exploration with minimal assumptions. See Table 1 for a conceptual comparison between prior work and MoReFree.

3. Preliminaries

3.1. DR3L: Demo-free, Reward-free, Reset-free RL

We formalize the "**D**emo-free, **R**eward-free, **R**eset-free **RL**" (DR3L) paradigm, where the only human effort is task specification of initial states and goals. The agent is trained to achieve the goals from the initial states without resets nor action supervision (e.g. rewards or demonstrations). Building off of the definition of autonomous reinforcement learning (ARL) from EARL (Sharma et al., 2021b), we formalize this problem in a goal-conditioned setting. Unlike ARL, we do not assume access to any form of training supervision on how to reach such state distributions (rewards or demonstrations).

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 $p(s_{t+1}|s_t, a_t)$. Importantly, the environmental reward function $r(s, a, g)$ is unavailable during training time, and is only used during evaluation to score policies. 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. Similar to the "persistent RL" definition in EARL, the evaluation of DR3L 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 unsupervised goal-conditioned RL approaches like LEXA (Mendonca et al., 2021) and PEG (Hu et al., 2023) train goal-conditioned policies using learned dynamical distance rewards (Hartikainen et al., 2019). Their success at solving long-horizon goal-conditioned tasks show that it is possible to efficiently train policies without a given reward function. Recall that we desire a method that 1) does not require environmental rewards to train policies and 2) can train without resets. Therefore, we select PEG as the backbone MBRL agent to fulfill the first criteria, and will later adapt it to the non-episodic training setting to fullfill the second criteria.

PEG (Hu et al., 2023) is a model-based Go-Explore framework that extends LEXA (Mendonca et al., 2021), an unsupervised goal-conditioned variant of DreamerV2 (Hafner et al., 2020). The following components are parameterized by θ and learned:

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

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 (Sekar et al., 2020). Both policies are trained on simulated trajectory rollouts in the world model.

► **Self-supervised goal-reaching reward function:** In the absence of an environmental reward function, the agent must define its own reward. PEG uses the learned dynamical distance function (Hartikainen et al., 2019), 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 Mendonca et al. (2021) 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.

Algorithm 1 Go-Explore

Input: g, π^G_θ, π^E_θ
$\tau_g \leftarrow \{\}$; $\tau_e \leftarrow \{\}$
for $t = 1$ to H_G **do**:
  $a_t \sim \pi^G(\cdot \mid s_t, g)$
  $s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t)$
  $\tau_g \leftarrow \tau_g \cup \{s_t\}$
for $t = 1$ to H_E **do**:
  $a_t \sim \pi^E(\cdot \mid s_t)$
  $s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t)$
  $\tau_g \leftarrow \tau_g \cup \{s_t\}$
return τ_g, τ_e

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 Hu et al. (2023) for details). Note that in principle, we could use alternative goal-selection mechanisms mentioned in Section 2 like uniform density (Pong et al., 2019) or maximum entropy (Pitis et al., 2020). We choose PEG since it outperforms prior work in exploratory goal selection.

Algorithm 2 MBRL Backbone (PEG)

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

4. Method

We now introduce MoReFree, a model-based approach that does not require rewards, demonstrations nor resets for training goal-conditioned policies. Model-based approaches offer benefits like sample-efficient, self-supervised goal-conditioned policy training inside world models (e.g. LEXA (Mendonca et al., 2021)) and enhanced exploration capabilities (e.g. Plan2Explore (Sekar et al., 2020)). Therefore, a model-based approach with both properties, like PEG (i.e. combines LEXA with Go-Explore) appears promising at first glance for the DR3L formulation.

However, such model-based approaches were developed and evaluated for episodic RL settings, and may suffer when trained without resets. Indeed, in our later ablation experiments, we find that directly running the backbone MBRL agent (PEG) results in poor performance. To port model-based RL over to our DR3L setting, we must adapt its key mechanisms to handle the lack of resets. 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 (Hu et al., 2023) is a MBRL agent with strong goal-conditioned exploration abilities. However, 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 1 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

Input: π^G_θ, π^E_θ, world model $\widehat{\mathcal{T}}_\theta$, ρ_{g^*}, ρ_0, ρ_E
Generate a random number r in $[0, 1]$
if $r < \alpha$ **then**
  $g^*, g_0 \sim \rho_{g^*}, \rho_0$
  $\tau_{g^*}, \tau^1_e \leftarrow$Go-Explore$(g^*, \pi^G, \pi^E)$
  $\tau_{g_0}, \tau^2_e \leftarrow$Go-Explore$(g_0, \pi^G, \pi^E)$
else
  $g \sim \rho_E$
  $\tau_g, \tau^1_e \leftarrow$Go-Explore$(g, \pi^G, \pi^E)$
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 of 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 (Hafner et al., 2019), 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 1 bottom row).

4.3. Implementation details

Our work builds on the top of PEG (Hu et al., 2023), a model-based goal-conditioned exploration method, and 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 A.3 for more details and the MoReFree codebase in the supplemental.

5. Experiments

We evaluate MoReFree and three competitive baselines on six DR3L tasks. We aim to answer: 1) Does MoReFree outperform baselines in DR3L tasks in terms of sample efficiency and performance? 2) What sorts of behavior does MoReFree and baselines exhibit in such tasks? 3) Which components of MoReFree influence and contribute to its performance?

Baselines: We want to compare MoReFree against competitive baselines in the DR3L setting, but from our literature search, we could only find R3L as a suitable baseline. R3L

is reported to have poor performance relative to more competitive but privileged approaches that use rewards / demos (IBC, VaPRL, and MEDAL (Kim et al., 2023; Sharma et al., 2021a; 2022)). Hence, we choose to compare MoReFree against privileged state-of-the-art approaches in reset free RL (IBC, MEDAL) rather than compare against R3L. The baselines are implemented using their official codebases, see Appendix A.2 for details.

- **MEDAL** (Sharma et al., 2022) requires demonstrations and trains two policies, one for getting back to demo states and another that achieves task goals.
- **IBC** (Kim et al., 2023) is a competitive baseline that does not rely on demonstrations and makes the weaker assumption of having sparse environmental rewards. IBC outperforms prior work (e.g. MEDAL, VaPRL) by defining a bidirectional curriculum for the goal-conditioned forward and backwards (i.e. reset) policies.
- **Oracle** is SAC (Haarnoja et al., 2018) trained under the episodic setting on the environmental reward.

Note that all baselines enjoy some advantage over MoReFree since none of them were explicitly designed for the DR3L setting; MEDAL uses demonstrations and rewards, IBC uses rewards, and Oracle uses resets and rewards. See Table 1 for a conceptual comparison between MoReFree and prior work.

Environments: We evaluate MoReFree and baselines on 6 tasks (see Figure 2). We select five tasks from IBC's evaluation suite of six tasks; we omit Fetch Reach, because it is trivially solvable. Next, we contribute a difficult navigation task, Ant, which is adapted from the PEG codebase (Hu et al., 2023). We summarize key properties:

- **PointUMaze**: A point-mass agent navigates a U-shape maze through continuous acceleration commands. During evaluation, the agent starts from the bottom-left corner and is tasked to reach the top-left corner. This environment is taken from IBC's evaluation suite (Kim et al., 2023).
- **Tabletop Manipulation**: The agent needs to grab and move the mug to one of the four goal locations. The initial state is always fixed and the goal state is uniformly sampled from four fixed locations. The task is taken from IBC, and was originally part of the EARL benchmark (Sharma et al., 2021b).
- **Sawyer Door**: The agent controls a Sawyer robot arm to close the door in an open position. During the reset-free training, it needs to learn to close the door and open the door again to practice. The door is opened to 60 degrees for evaluation. This task is from the EARL benchmark also used by IBC.
- **Fetch Push**: The agent commands a Fetch robot arm to push the object initialized at the center of the table to goal locations sampled from a 15cm The environment is taken from IBC's evaluation suite, which modified the original environment from Plappert et al. (2018). To prevent the block from falling off the table, the IBC authors artificially limited the block position with block position constraints. This resulted in unrealistic jittering behavior near the limits. To avoid this, we removed the artificial joint constraints and surrounded the table with physical walls. Furthermore, we enable the usage of the grippers (disabled in IBC's version) to permit picking behaviors (i.e. useful for resetting), at the cost of increased action space and exploration difficulty.
- **Fetch Pick&Place**: Similar to the Fetch Push, except the Fetch robot arm needs to pick up the block and move it to a target location. The environment is also taken from IBC's evaluation suite. We also replaced the artificial block position limits with physical walls. To make the task more difficult, we modified the evaluation goal distribution to only include goals in the air. This requires the agent to learn picking behaviors to solve the task whereas goals on the table can be solved with pushing.
- **Ant**: The 4-legged ant agent needs to navigate in a square room to a given goal, which is uniformly located in the top-right corner. The initial state is at the center point with randomness. It is adapted from Hu et al. (2023), with changing the U-shape maze into a square room.

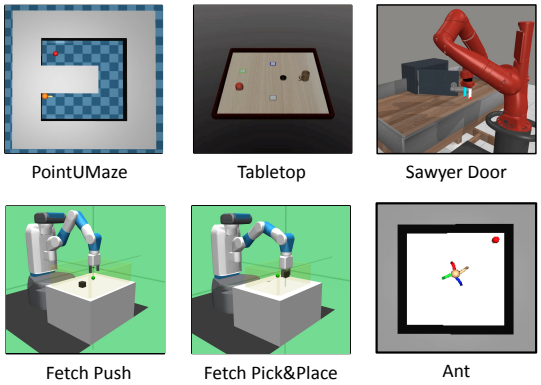

Figure 2. We evaluate MoReFree on six reset-free tasks ranging from navigation to manipulation.

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 A for more experimental details.

5.1. Results

As shown in Fig 3, MoReFree, without demonstrations or rewards, outperforms other baselines with privileged access to supervision in both final performance and sample

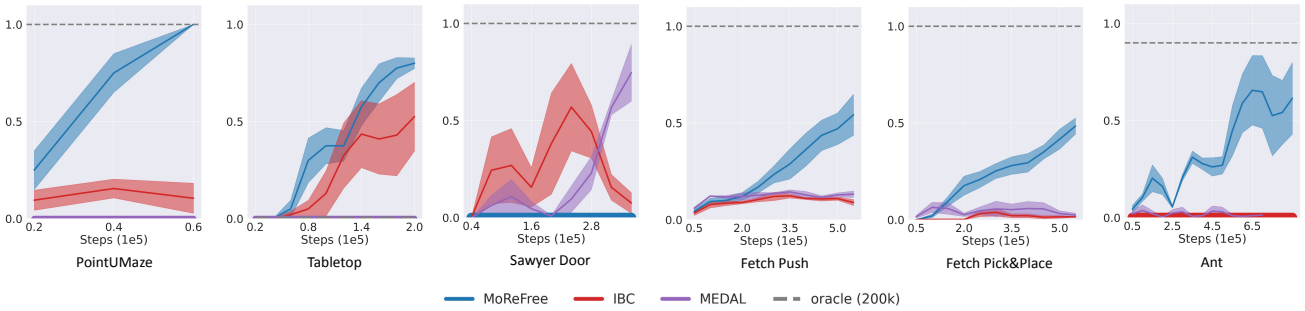

Figure 3. MoReFree significantly outperforms baselines with privileged access to demos (MEDAL) and task rewards (IBC) in 5/6 tasks. In 4 tasks, only MoReFree is able to learn meaningful behavior, showcasing MoReFree's sample efficiency.

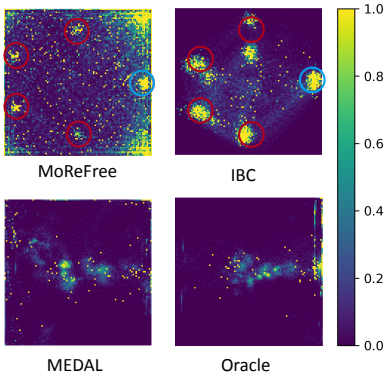

Figure 4. XY state visitation heatmap of the mug in Tabletop of various approaches. MoReFree's heatmap shows high state diversity while retaining high visitation counts near the start (blue circle) and evaluation state distributions (red circles).

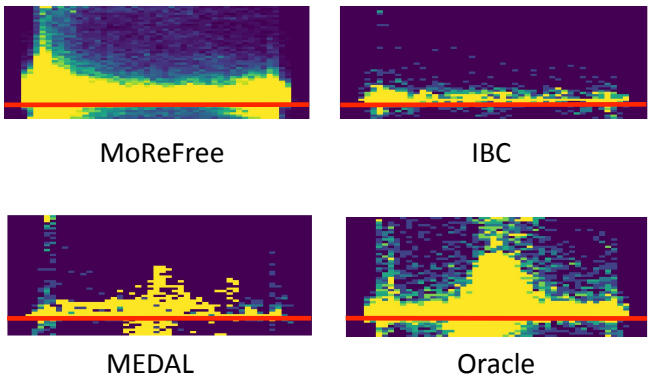

Figure 5. XZ view of object state visitation heatmap on Fetch Pick&Place. States above the red line mean the object is in the air. MoReFree picks up the object frequently and in diverse ways.

efficiency in 5/6 tasks. The gains are especially large in PointUMaze, Fetch Push and Fetch Pick&Place, and Ant; IBC and MEDAL do not learn anything within the given time steps. MoReFree learns 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. In Sawyer Door, MoReFree fails to learn. We investigated further in Appendix D and found that multiple Dreamer-based agents (DreamerV2, DreamerV3) struggle to solve this task even in the episodic setting. If Dreamer, the base MBRL algorithm that MoReFree extends, cannot solve the task in episodic setting, then it is unlikely MoReFree will work. See the website for videos of MoReFree and baselines.

Comparing exploration behavior. We perform qualitative analysis of MoReFree and baselines' exploration behaviors. First, we visualize the replay buffer states of different agents to characterize their exploration behavior. Figure 4 shows heatmaps on XY state visitations of the mug on Tabletop Manipulation task, where increasing intensity corresponds to higher visitations. Notably, MoReFree showcases two desirable properties: diverse exploration shown through numerous medium/high intensity cells scattered uniformly across the plane, and distinct high visitation groups clustered around the initial (blue circle) and goal states (red circle). This suggests that Back-and-forth Go-Explore is creating a curriculum that simultaneously encourages exploration, reset, and evaluation behavior. In contrast, the baselines are one or both properties. IBC's bidirectional curriculum also encourages it to shuttle back and forth between evaluation and initial states, resulting in distinct clusters around initial and evaluation states. However, IBC, MEDAL, and Oracle all fail to explore well; their heatmaps are mostly populated with low visitation cells.

We perform the heatmap visualization of the XZ dimensions of the block position for Fetch Pick&Place in Figure 5. All states above the red line are in the air, implying the agent has learned to pick up the block. MoReFree's heatmap shows numerous high visitation cells above the red line, and populated uniformly along the horizontal axis of the table. This means that MoReFree has learned how to pick up the block from diverse initial states and move it to diverse states in the air. In contrast, IBC and MEDAL have sparser

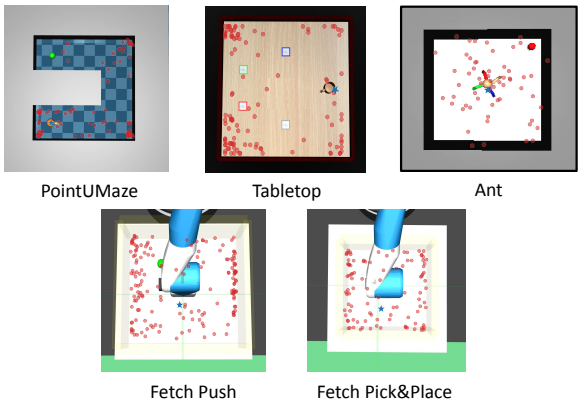

Figure 6. We visualize the start position (red dots) of successful "Back" trajectories of MoReFree's Back-and-Forth Go-Explore, where π^G is directed to reset the environment.

coverage above the red line. We visualize heatmaps for other environments in Appendix B and find similar trends where MoReFree exploration is superior.

Analyzing Back-and-forth Go-Explore. Next, we more closely investigate the qualitative behavior of MoReFree's Back-and-forth Go-Explore. To investigate 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 right before a successful "Back" trajectory is executed in Figure 6. The starting locations (red dots) of the agent / object are in corners or next to walls in all environments. This observation (that all starting locations are in corners) 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[1].

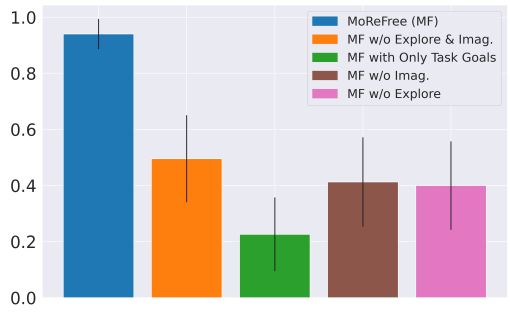

Figure 7. Ablations on 5 variants of MoReFree over all 4 environments with normalized performance. MoReFree is significantly better than the backbone MBRL agent, and all components of MoReFree (exploration and policy optimization) are important.

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

5.2. Ablation study

We seek to analyze MoReFree's components' contributions to performance. First, we verify that the performance of MoReFree can be attributed to its contributions rather than the vanilla MBRL agent (PEG). To do so, we define two ablations. First, **MF w/o Explore & Imag.** removes all of our features (Back-and-Forth Go-Explore and biased goal sampling in imagination) and reduces to PEG, the backbone MBRL agent. Next, **MF with Only Task Goals** sets $\alpha = 1$, which results in the Back-and-Forth Go-Explore and goal sampling in imagination to only sample from initial and evaluation goal distributions. In Figure 7, we run ablations over all environments and plot the normalized final performance. MoReFree (blue) substantially outperforms both ablations (orange, green), showing that MoReFree's performance gain is more than just using a strong MBRL backbone and that sampling all 3 types of goals is important.

Next, 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 in Figure 7. This is somewhat intuitive, as the two components rely on each other. 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 DR3L setting. See Appendix C for details.

6. Conclusion and future work

As a step towards fully autonomous training, we propose MoReFree, a model-based approach to solving tasks without demonstrations, rewards, nor resets. By focusing model-based exploration and goal-conditioned policies on task-relevant states, we are successfully able to train unsupervised goal-conditioned model-based RL agents without resets. Our experiments show that for a majority of tasks, MoReFree substantially outperforms baselines that do assume access to rewards and demonstrations. Despite MoReFree's overall success, MoReFree is not without limitations. MoReFree is a model-based approach, and as such inherits all of its disadvantages. For example, we believe Sawyer Door is a task where learning the dynamics is harder than learning the policy (see Appendix D), disadvantaging MBRL approaches. We hope MoReFree inspires future efforts in increasing autonomy in RL.

7. Impact Statement

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.

References

Baranes, A. and Oudeyer, P.-Y. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73, 2013. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2012.05.008. URL https://www.sciencedirect.com/science/article/pii/S0921889012000644.

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

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

Eysenbach, B., Salakhutdinov, R., and Levine, S. C-learning: Learning to achieve goals via recursive classification. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=tc5qisoB-C.

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

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

Gupta, A., Yu, J., Zhao, T. Z., Kumar, V., Rovinsky, A., Xu, K., Devlin, T., and Levine, S. 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)*, pp. 6664–6671. IEEE, 2021.

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

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

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

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

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

Hu, E. S., Chang, R., Rybkin, O., and Jayaraman, D. Planning goals for exploration. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6qeBuZSo7Pr.

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

Kim, J., Cho, D., and Kim, H. J. Demonstration-free autonomous reinforcement learning via implicit and bidirectional curriculum. *arXiv preprint arXiv:2305.09943*, 2023.

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

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

Lu, K., Mordatch, I., and Abbeel, P. Adaptive online planning for continual lifelong learning, 2020b. URL https://openreview.net/forum?id=HkgFDgSYPH.

Ma, Y. J., Sodhani, S., Jayaraman, D., Bastani, O., Kumar, V., and Zhang, A. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.

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

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

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

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

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

Schenck, C. and Fox, D. Perceiving and reasoning about liquids using fully convolutional networks. *The International Journal of Robotics Research*, 37(4-5):452–471, 2018.

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

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

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

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

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

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

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

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

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

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

Zhu, H., Gupta, A., Rajeswaran, A., Levine, S., and Kumar, V. 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.

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

A. Experimental Details

A.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.

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.

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.

Fetch Push The state space is 25 dimensional and action space is 4 dimensional. Different from the original Fetch Push task, in our case walls are added to prevent the object from dropping out of the table. The workspace of the robot arm is also limited. The object is always initialized to a fixed location, and goal distribution during the evaluation is $\mathcal{U}(-0.15, 15)$. Fetch Push used in IBC (Kim et al., 2023) paper, the object 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 object and keep pushing it towards the limit boundary, see videos on our project website [2]). 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 object is in corners, it will need to operate the gripper to drag the object 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. See a visual difference between our Pick&Place and IBC's in Figure 8.

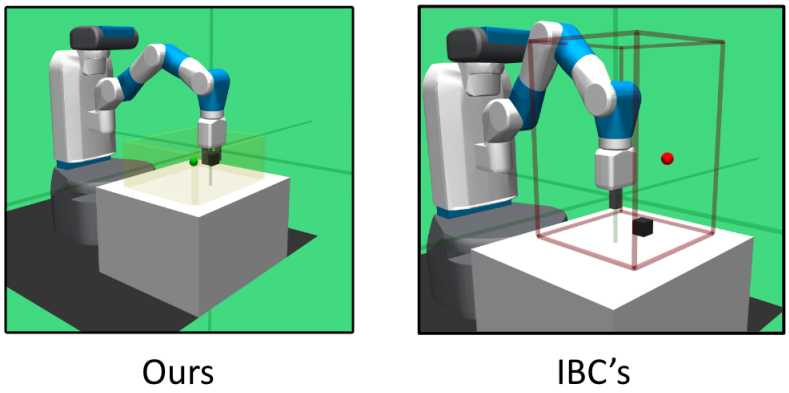

Figure 8. We use walls (yellow) to limit the workspace of the object and prevent it from falling, while IBC adds joint constraints on the object (visualized as the red frame).

Fetch Pick&Place We add walls in the same way as we did for Fetch Push. 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

[2] https://sites.google.com/view/morefree

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.

Ant We adapt the AntMaze task from environments[3] 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.

A.2. Baseline Implementations

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.

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 (Hu et al., 2023), we extend their codebase [6] 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.

A.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.

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

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

A.4. 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.

[3] https://github.com/edwhu/mrl
[4] https://github.com/snu-larr/ibc_official
[5] https://github.com/architsharma97/medal
[6] https://github.com/penn-pal-lab/peg

B. More Visualizations on Replay Buffer

We visualize the replay buffer of different agents on more tasks. See Figure 9 for xy location data of the agent in PointUMaze and Figure 10 for xy location data of the object in Fetch Push and Fetch Pick&Place. Overall, we see MoReFree explores the whole state space better, and has much more interactions with the object. Meanwhile, due to back-and-forth procedure, MoReFree collects many data near initial / goal states, which are important for the evaluation.

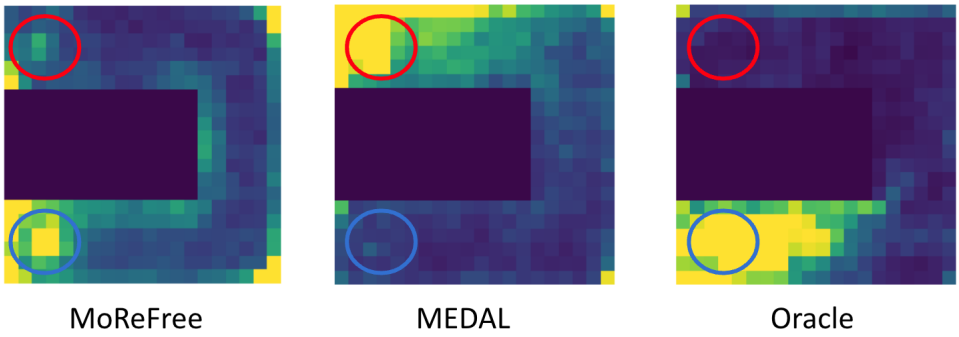

Figure 9. State visitation heatmap on point maze. MoReFree has special focuses on both initial state (blue circles) corner and goal state (red circles), while explore much uniformly. MEDAL collects lots of data near the goal state and little data on the initial state. Both MEDAL and Oracle explore less extensively.

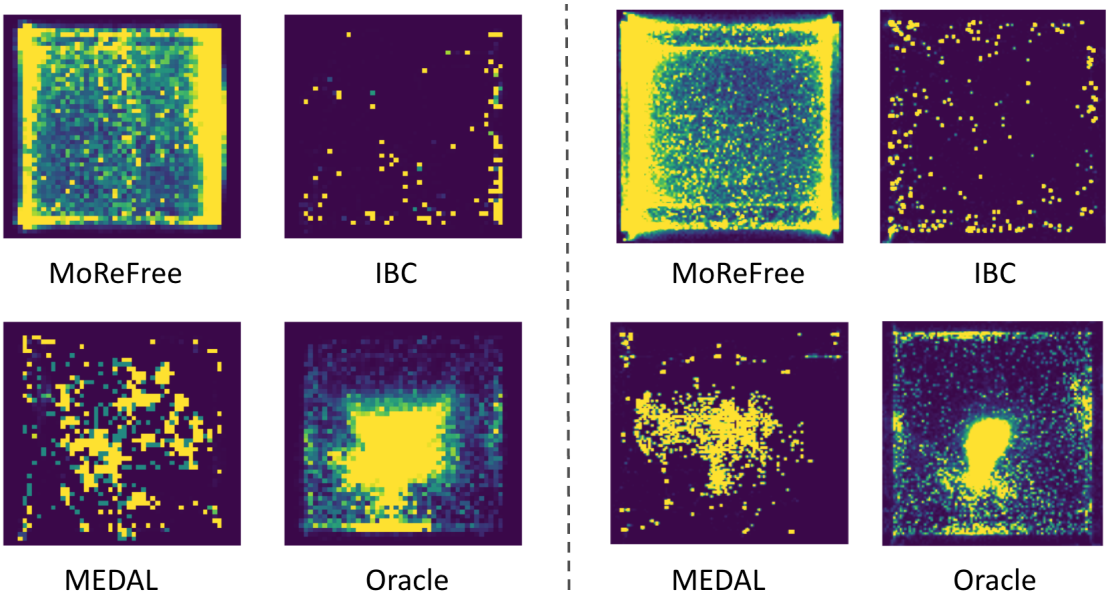

Figure 10. Object state visitation heatmap on Fetch Push (left) and Fetch Pick&Place (right) of different agents. MoReFree better explores the whole state space, while IBC and MEDAL do not have too much interactions with the object, thus lighted areas are scattered everywhere.

C. Detailed Ablations

We report learning curves for each variant agent we ablate in Section 5.2 on every task in Figure 11. Since MoReFree does not learn at all in Saywer Door task, we exclude the ablation for it. In each 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 12. In the performance on PointUMaze from Figure 11, sampling exploratory goals for data collection is important

Figure 11. 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.

(MF w/o Explore & Imag. outperforms other ablations). But we see in 12, 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 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.

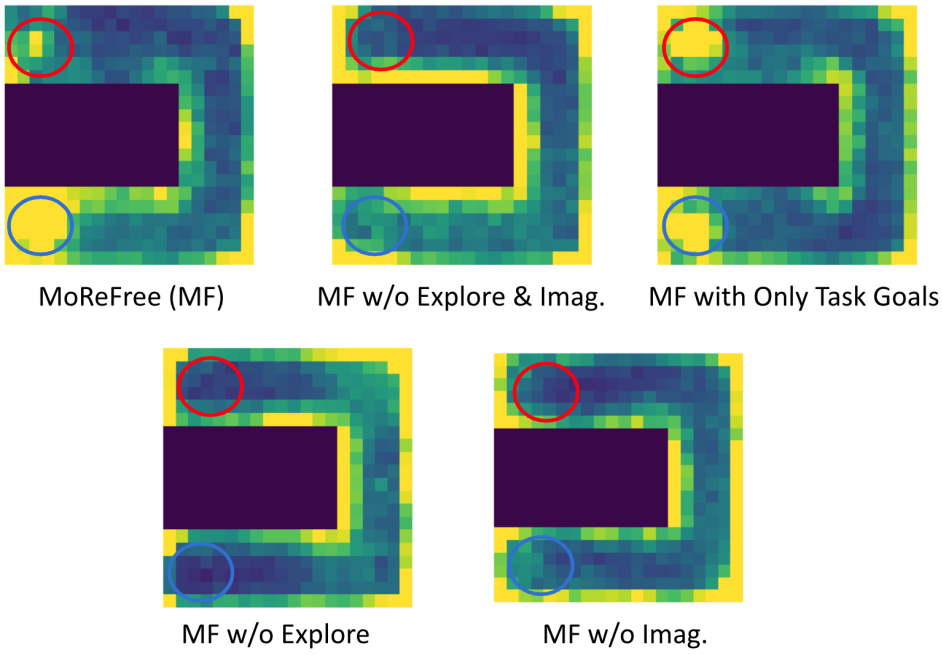

Figure 12. State visitation heatmap on PointUMaze task of all ablations. Red circles are evaluation goal states and blues are initial states. We see MoReFree collect good amount of data near initial / goal states while stronger exploration. MF w/o Explore and MF w/o Imag. could not gather task-relative data, which further causes poor performance.

D. MBRL on Sawyer Door

We investigate why MoReFree fails 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 have the robot hold the door to prevent it from opening it too much, see Figure 13. Although MoReFree is trained on the simplified environment, we see the learning curve of MoReFree on Sawyer Door is completely flat in Figure 3, compared with other baselines trained on the original task. We wonder why MoReFree can show the same performance and gain benefits as it does in other environments.

MoReFree uses DreamerV2 as a backbone agent and extends 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 MoReFree in 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.

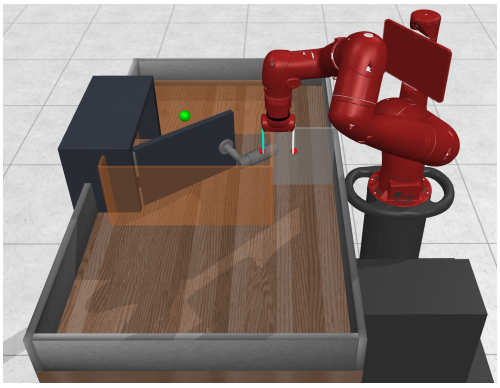

Figure 13. Simplified version of Sawyer Door. Orange walls show the limited workspace for the robot arm, and a grey wall is added to limit the movement of the door. The door can only move to maximum 60 degrees.

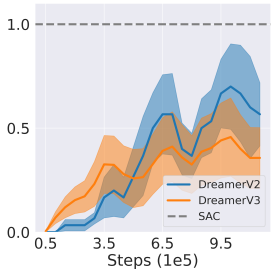

Figure 14. Performance of DreamerV2 and V3 on episodic Sawyer Door task. SAC can solve the task in 200k steps, while after 1 million steps MBRL is still not able to steadily solve the task.

We then run the underlying MBRL backbones under the episodic setting. Figure 14 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 MoReFree does not work on more difficult setting, which is reset-free and reward-free. 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.

[7] https://github.com/danijar/dreamerv2
[8] https://github.com/danijar/dreamerv3