A Smooth Sea Never Made a Skilled `SAILOR`: Robust Imitation via Learning to Search

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

The fundamental limitation of the behavioral cloning (BC) approach to imitation learning is that it only teaches an agent what the expert did at states the expert visited. This means that when a BC agent makes a mistake which takes them out of the support of the demonstrations, they often don't know how to recover from it. In this sense, BC is akin to *giving the agent the fish* – giving them dense supervision across a narrow set of states – rather than teaching them *to fish*: to be able to reason independently about achieving the expert's outcome even when faced with unseen situations at test-time. In response, we explore *learning to search* (L2S) from expert demonstrations, i.e. learning the components required to, at test time, plan to match expert outcomes, even after making a mistake. These include *(1)* a world model and *(2)* a reward model. We carefully ablate the set of algorithmic and design decisions required to combine these and other components for stable and sample/interaction-efficient learning of recovery behavior without additional human corrections. Across a dozen visual manipulation tasks from three benchmarks, our approach `SAILOR` consistently out-performs state-of-the-art Diffusion Policies trained via BC on the same data. Furthermore, scaling up the amount of demonstrations used for BC by 5-10× still leaves a performance gap. We find that `SAILOR` can identify nuanced failures and is robust to reward hacking.

1 Introduction

The workhorse of modern imitation learning (IL) is behavioral cloning (BC, Pomerleau [1988]). From training Diffusion Policies (DPs, Chi et al. [2023]) on per-task expert demonstrations collected via a variety of teleportation interfaces [Zhao et al., 2023, Chi et al., 2024, Wu et al., 2024] to Visual-Language-Action models (VLAs, Team et al. [2024], Kim et al. [2024], Intelligence et al. [2025]) trained on wider, multi-task datasets [Khazatsky et al., 2024, ONeill et al., 2024], we see the same *recipe* applied: collecting more data to train more expressive policy models. The latent hope here is that scaling will eventually lead to a "ChatGPT moment" for robotics [Vemprala et al., 2023].

However, even for the *simpler* problem of language modeling where one doesn't have to deal with the complexities of embodiment (e.g. meaningful, stochastic dynamics and physical safety concerns), simply scaling *next token prediction* (i.e. BC) was insufficient: we needed *interactive learning* in the form of Reinforcement Learning from Human Feedback (RLHF, Stiennon et al. [2020], Ouyang et al. [2022]) and more recently, Test-Time Scaling (TTS, Jaech et al. [2024], Guo et al. [2025]), to build robust systems. If internet-scale offline pretraining was insufficient to solve language modeling, it stands to reason that we'll need similar interactive learning algorithms to train (embodied) agents.

At heart, this is because even when they are trained on large amounts of data and over expressive policy classes, agents still sometimes make mistakes that take them out of the support of the offline data. This is a fundamental property of sequential decision-making: one has to deal with the consequences

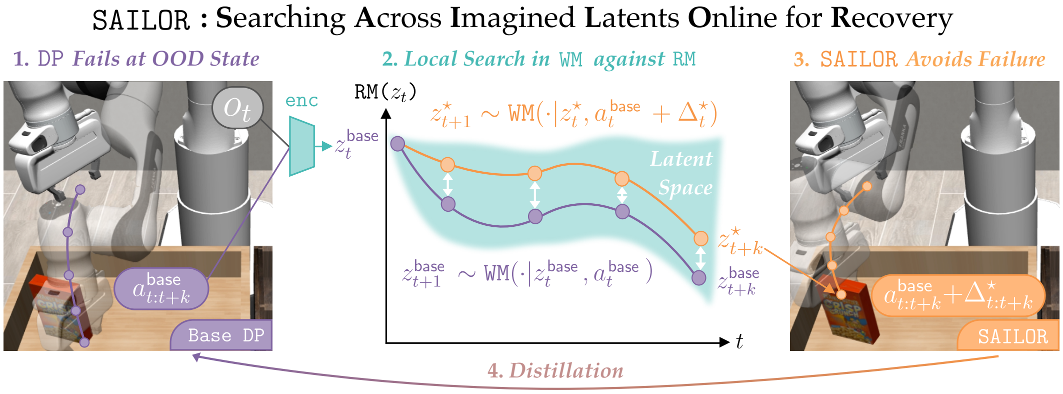

SAILOR : Searching Across Imagined Latents Online for Recovery
1. DP Fails at OOD State
2. Local Search in WM against RM
3. SAILOR Avoids Failure
o_t
enc
z_t^{base}
RM(z_t)
$z^\star_{t+1} \sim \text{WM}(\cdot|z^\star_t, a_t^{\text{base}} + \Delta^\star_t)$
Latent Space
$z^\star_{t+k}$
z^{base}_{t+k}
$z^{\text{base}}_{t+1} \sim \text{WM}(\cdot|z_t^{\text{base}}, a_t^{\text{base}})$
t
$a^{\text{base}}_{t:t+k}$
Base DP
$a^{\text{base}}_{t:t+k}+\Delta^\star_{t:t+k}$
SAILOR
4. Distillation

Figure 1: We introduce SAILOR, a method for *learning to search* from expert demonstrations. By learning world and reward models on a mixture of expert and base-policy data, we endow the agent with the ability to, at test time, reason about how to recover from mistakes that the base policy makes.

of their own prior actions. When faced with the resulting unseen situation, we'd like an agent that attempts to *recover* and match the expert's *outcome* (whenever it is possible to do so). The most direct approach to teach an agent this recovery behavior is to ask a human-in-the-loop to correct mistakes [Ross et al., 2011, Kelly et al., 2019, Spencer et al., 2020]. While simple, such an approach can be difficult to scale as it fundamentally makes *human* time the bottleneck for *robot* learning.

In an ideal world, we'd like our robots to be able to learn to recover from their *own* mistakes without additional human feedback. There are two fundamental capabilities an agent needs to reason at test-time about recovering from mistakes. The first is *prediction*: to understand the consequences of their proposed actions. The second is *evaluation*: to know which outcomes are preferable to others.

We propose an algorithmic paradigm that allows us to acquire both of these capabilities without requiring any sources of human data beyond the standard imitation learning pipeline. In other words, *a better recipe with the same ingredients*. In particular, rather than merely learning a policy from expert demonstrations, we propose learning a *local world model* (WM) and a *reward model* (RM) from demonstration and base policy data [Ren et al., 2024b]. By combining these components with a planning algorithm, we have the capability to, at test time, reason about how to recover from mistakes that the base policy makes by planning against our learned RM inside our learned WM. Thus, rather than mere imitation, we're ***learning to search*** (L2S, Ratliff et al. [2009]) from expert demonstrations. Our key insight is that ***we can infer the latent search process required to recover from local mistakes from the same source of human data required for the standard behavioral cloning pipeline***.

Put differently, in contrast to approaches like BC and DAgger that *give the agent the fish* – i.e. relying on a human teacher to demonstrate desired or recovery behavior, we focus on teaching the agent *to fish*: ***to develop the reasoning process required to match the expert's outcomes, even when faced with situations unseen in the training dataset***, often as a result of the agent's own earlier mistakes.

In our work, we focus on long-horizon visual manipulation tasks and therefore instantiate the L2S paradigm by using base Diffusion Policies [Chi et al., 2023], Dreamer World Models [Hafner et al., 2024], and the Model-Predictive Path Integral Control (MPPI, Williams et al. [2017]) Planner. Concretely, we learn a residual planner [Silver et al., 2018] that performs a *local search* at test time to correct mistakes the base policy makes. We call our composite architecture SAILOR: **Searching Across Imagined Latents Online for Recovery.** More specifically, our contributions are three-fold:

1. We demonstrate that across a dozen visual manipulation problems at three different dataset scales, SAILOR outperforms Diffusion Policies trained on the same demonstrations. Furthermore, we find that scaling up the number of demonstrations passed to DP by ≈ 5-10× often still leaves a performance gap. We are also significantly more interaction-efficient than model-free inverse RL methods that use state of the art Diffusion Policy RL algorithms like DPPO [Ren et al., 2024a].

2. We carefully ablate the algorithmic and design decisions required to learn and combine the above components for stable and sample-efficient learning. Specifically, we find that "warm starting" the hybrid world model training period, doing online hybrid world model fine-tuning [Ross

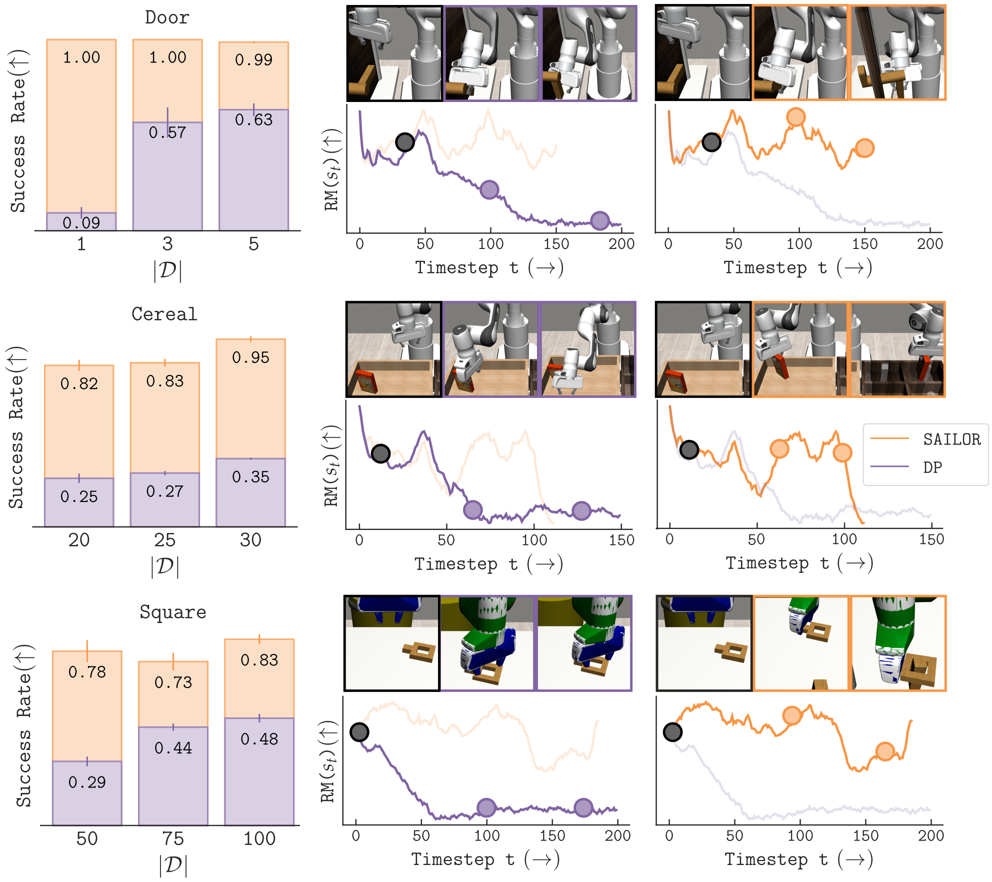


Figure 2: **Left:** we see `SAILOR` consistently out-perform diffusion policies trained on the same demos across twelve visual manipulation problems at multiple dataset scales. **Right**: our learned reward model is able to detect shared prefixes, base policy failure suffixes and `SAILOR`'s successful suffixes.

and Bagnell, 2012, Vemula et al., 2023, Ren et al., 2024b], and periodically distilling the learned search algorithm into the base policy (akin to expert iteration [Anthony et al., 2017, Sun et al., 2018] or Guided Policy Search [Levine and Koltun, 2013]) are critical for performance.

3. We investigate the fidelity and test-time scaling properties of our learned search algorithm. We find that our learned reward model is able to identify nuanced failures that occur at different stages of our long-horizon manipulation tasks and that our composite stack is robust to reward hacking.

2 Related Work

Imitation Learning. The simplest approach to imitation learning is *behavioral cloning* (BC, Pomerleau [1988], Chi et al. [2023]), where one learns a policy via maximizing the likelihood of expert actions at expert states. However, whether it is due to limited demonstrations [Swamy et al., 2022b], optimization error [Swamy et al., 2021], misspecification [Espinosa-Dice et al., 2025], or partial observability [Swamy et al., 2022a], a policy trained via BC will often make a mistake that takes it out of the support of the expert demonstrations, where it can continue to make errors, an issue known as *compounding errors* [Ross et al., 2011], which is unavoidable in general [Swamy et al., 2021].

Under the hood, the reason a BC policy makes mistakes is due to the *covariate shift* between the training input distribution (expert states) and the testing input distribution (learner states). Interacting with the environment allows us to generate samples from this test distribution. If we're able to further query the demonstrator in the loop, we can ask them for action labels at these states [Ross et al., 2011, Kelly et al., 2019, Spencer et al., 2020]. However, such approaches are often labor-intensive.

Learning to Search = RM + WM. In Learning to Search (L2S, Ratliff et al. [2009]), one learns the components (i.e., RM and a WM) required to, at test time, search for actions, rather than directly learning a policy. If there are no meaningful dynamics or stochasticity (e.g., append-only, autoregressive language generation), one can eschew the WM and plan over the entire horizon via repeated sampling and scoring under the RM, an approach known as Best-of-N (BoN, Brown et al. [2024]). Otherwise, one samples some plans, performs stochastic rollouts inside the WM, calling the RM along the way to estimate performance before updating the sampling distribution. We use the `MPPI` algorithm of Williams et al. [2017] as our search procedure due to the strong performance it has demonstrated when deployed inside a learned WM on robotics problems [Hansen et al., 2022, 2024]. In contrast to prior L2S approaches. `SAILOR` learns from expert demonstrations alone (i.e., without test-time access to a ground-truth simulator as in Silver et al. [2017], Brown and Sandholm [2019] or *any* information about the ground-truth reward function as in Hansen et al. [2022, 2024]).

After expending computation at test-time for a search procedure, it is often valuable to distill the search process back into the base policy, an approach known as *expert* or *dual policy iteration* [Anthony et al., 2017, Sun et al., 2018]. While such approaches have long demonstrated strong performance in continuous control [Levine and Koltun, 2013, Wang et al., 2025], they have attracted renewed interest in the context of LLM reasoning [Zelikman et al., 2022, Gandhi et al., 2024, Hosseini et al., 2024]. `SAILOR` can be seen as a generalization of these ideas to continuous control problems with stochastic dynamics, visual observations, and unknown reward functions, which none of the above can handle directly. We ablate the value of ExIt-like updates and find they provide some improvement in policy performance. Recent work by Wu et al. [2025b] applies similar ideas on real-world visual manipulation problems but focuses on mode selection rather than more general behavior correction. Another perspective on `SAILOR` is that it fuses the paradigms of residual RL [Silver et al., 2018, Yuan et al., 2025, Ankile et al., 2024] and L2S by *learning to search for residuals*.

3 Robust Imitation via *Learning to Search*

We adopt the framework of a Partially Observed Markov Decision Process (POMDP, Kaelbling et al. [1998]) and use $\mathcal{O}$, $\mathcal{A}$, and $\mathcal{Z}$ to denote the observation, action, and latent spaces, respectively. We also use $\gamma \in [0, 1]$ to denote the discount factor and k to denote our policy's planning horizon.

3.1 Why Learn to Search?

Before we delve into the practicalities of L2S, it is worth pausing for a moment to think about why L2S is an attractive algorithmic paradigm. After all, interactive approaches to imitation learning like DAgger [Ross et al., 2011] and inverse RL [Ziebart et al., 2008] are already provably robust to the compounding errors that can stymie offline algorithms like behavior cloning [Swamy et al., 2021]. Furthermore, approaches like inverse RL that learn rewards / *verifiers* from demonstrations can take advantage of lower sample complexity on problems for which *generation-verification gaps* exist [Swamy et al., 2025]. Beyond these benefits, the unique advantage of the L2S paradigm is a *computational* one: the ability to only need to plan at the states encountered at *test-time*, rather than potentially at all states in the underlying MDP during standard RL [Kearns et al., 2002]. In this sense, L2S lets us trade potentially redundant computation at train time for only what's necessary at test time. Furthermore, expert iteration-style distillation updates let us recycle these compute cycles.

3.2 What is Learning to Search?

In `SAILOR`, we *learn a search algorithm* that generates residual plans to correct a nominal plan generated by an arbitrary base policy. Concretely, after learning a world model `WM`, reward model `RM`, and critic `V` from a combination of expert demonstrations and base policy rollouts, we perform repeated stochastic rollouts of potential residual plans inside our `WM`, scoring the latent states with `R` and `V`, before selecting the plan with the highest estimated score. We then execute the first step of the corrected plan in the real world before re-planning in the style of *model predictive control* (MPC). More formally, at inference time, we attempt to solve the following *local search* problem:

$$\Delta^{\star}_{t:t+k} = \operatorname*{argmax}_{\Delta_{t:t+k}} \mathbb{E}_{\texttt{WM}} \left[\sum_{h=0}^{k-1} \gamma^{h} \texttt{RM}(z_{t+h}) + \gamma^{k} \texttt{V}(z_{t+k}) \middle| o_t, a^{\textsf{base}}_{t:t+k} + \Delta_{t:t+k} \right]. \tag{1}$$

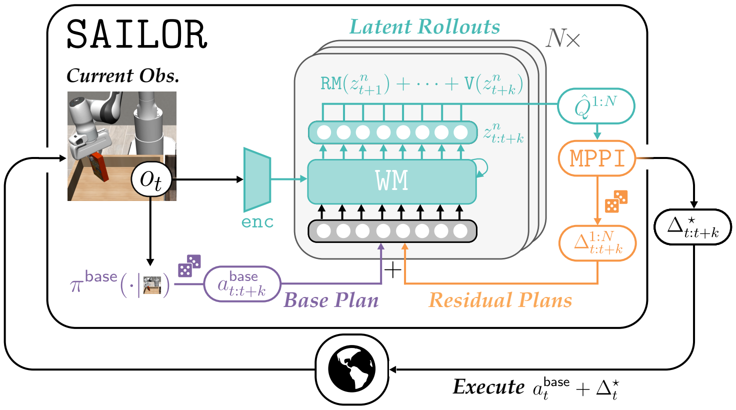

SAILOR
Current Obs.
Latent Rollouts
$N\times$
$\mathtt{RM}(z^n_{t+1}) + \cdots + \mathtt{V}(z^n_{t+k})$
$\hat{Q}^{1:N}$
$z^n_{t:t+k}$
MPPI
WM
o_t
enc
$\Delta^{1:N}_{t:t+k}$
$\Delta^{\star}_{t:t+k}$
$\pi^{\text{base}}(\cdot|$
$a^{\text{base}}_{t:t+k}$
+
Base Plan
Residual Plans
Execute $a^{\text{base}}_t + \Delta^{\star}_t$

Figure 3: At inference time, `SAILOR` performs a search for residual plans to correct mistakes in the base policy's nominal plan in the latent world model `WM` against the learned reward model `RM` and critic `V`. It then executes the first step of the best corrected plan before re-planning, MPC-style.

Algorithm 1 `SAILOR` (Inference)

```
**Input:** Base Policy $\pi^{\text{base}}$, World Model $\mathtt{WM}$, Reward Model $\mathtt{RM}$, Critic Network $\mathtt{V}$, Obs. $o_t$,
Sample nominal plan from base: $a^{\text{base}}_{t:t+k} \sim \pi^{\text{base}}(o_t)$.
**for** iteration $j$ in $1 \ldots J$ **do**
    // Can perform in parallel
    **for** residual plan $n$ in $1 \ldots N$ **do**
        Sample $\Delta^n_{t:t+k} \sim \mathcal{N}(\mu^j, \sigma^j)$.
        Execute plan in WM: $z^n_{t:t+k} \sim \mathtt{WM}(o_t, a^{\text{base}}_{t:t+k} + \Delta^n_{t:t+k})$.
        Compute $\hat{Q}$s: $\hat{Q}^n \leftarrow \sum_{h=0}^{k-1} \gamma^h \mathtt{RM}(z^n_{t+h}) + \gamma^k \mathtt{V}(z^n_{t+k})$.
    **end for**
    // Update mean and std
    $\mu^{j+1}, \sigma^{j+1} \leftarrow \mathtt{MPPI\_update}(\mu^j, \sigma^j, \hat{Q}^{1:N}, \Delta^{1:N}_{t:t+k})$.
**end for**
**Return** corrected plan $a^{\star}_{t:t+k} \sim \mathcal{N}(a^{\text{base}}_{t:t+k} + \mu^J, \sigma^J)$.
```

We then execute the first of these actions, $a^{\text{base}}_t + \Delta^{\star}_t$, in the environment before re-planning on top of the fresh observation and base plan. We describe this process in Alg. 1 and visualize it in Fig. 3. We now describe each of these components before describing the phases of training.

Base Policy. We assume access to a *base policy* π^{base} that can generate k-step plans given an observation o_t: $a^{\text{base}}_{t:t+k} \sim \pi^{\text{base}}(o_t)$. This could be an arbitrary policy pretrained on a wide set of data like a VLA [Team et al., 2024, Intelligence et al., 2025, Kim et al., 2024] or a task-specific Diffusion Policy (DP, Chi et al. [2023]) trained via behavioral cloning. We adopt the latter for our experiments for simplicity, but note that the general `SAILOR` framework does not require doing so. [1] As is standard practice [Chi et al., 2023], we use a ResNet-18 [He et al., 2016] encoder for our DP that takes in both image observations and the proprioceptive state of the robot and generates 8-step plans.

World Model. To avoid the complexity of modeling the dynamics of and planning directly in the space of high-dimensional observations, we adopt the Recurrent State-Space Model architecture (RSSM, Hafner et al. [2019, 2024]). This means our world model is composed of three key components, i.e. $\mathtt{WM} = \{\mathtt{enc}, \mathtt{f}, \mathtt{dec}\}$. The first, the *encoder* $\mathtt{enc} : \mathcal{Z} \times \mathcal{O} \times \mathcal{A} \to \mathcal{Z}$ encodes the observation into latent space i.e. $z_t \approx \mathtt{enc}(z_{t-1}, a_{t-1}, o_t)$. The second, the *latent dynamics model* $\mathtt{f} : \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$ predicts the next latent after taking an action, i.e. $z_t \approx \mathtt{f}(z_{t-1}, a_{t-1})$. The third, the decoder $\mathtt{dec} : \mathcal{Z} \to \mathcal{O}$ tries to reconstruct the observation from the latent state, i.e. $\mathtt{dec}(z_t) \approx o_t$. In `SAILOR`, ***we train all three of these components on hybrid data***, i.e. a mixture of demonstrations $\mathcal{D}$ and on-policy rollouts $\mathcal{B}$. As argued by Ross and Bagnell [2012], Vemula et al. [2023], Ren et al. [2024b], while such a

[1] Explicitly, we make no assumptions on the base policy, other than the ability to fine-tune it via behavioral cloning (i.e., maximum likelihood estimation) for an optional *expert iteration* subroutine.

Algorithm 2 `SAILOR` (Training)

```
 1: Input: Base Policy $\pi^{\text{base}}$, Expert Demos $\mathcal{D}$
 2: // Phase I: Warm Start
 3: Collect on-policy rollouts: $\mathcal{B} \leftarrow \{(o_t, a_t) \sim \pi^{\text{base}}\}$.
 4: // Co-train components on hybrid data
 5: $\texttt{WM} \leftarrow \text{argmin}_{\texttt{WM}}\, \ell(\mathcal{D}, \mathcal{B}, \texttt{WM}), \texttt{RM} \leftarrow \text{argmin}_{\texttt{RM}}\, \ell(\mathcal{D}, \mathcal{B}, \texttt{RM}), \texttt{V} \leftarrow \text{argmin}_{\texttt{V}}\, \ell(\mathcal{D}, \mathcal{B}, \texttt{V})$.
 6: // Phase II: Online Fine-Tuning
 7: for iteration $j$ in $1 \dots J$ do
 8:     Collect on-policy rollouts: $\mathcal{B} \leftarrow \mathcal{B} \cup \{(o_t, a_t) \sim \texttt{SAILOR}\}$.
 9:     $\texttt{WM} \leftarrow \text{argmin}_{\texttt{WM}}\, \ell(\mathcal{D}, \mathcal{B}, \texttt{WM}), \texttt{RM} \leftarrow \text{argmin}_{\texttt{RM}}\, \ell(\mathcal{D}, \mathcal{B}, \texttt{RM}), \texttt{V} \leftarrow \text{argmin}_{\texttt{V}}\, \ell(\mathcal{D}, \mathcal{B}, \texttt{V})$.
10:     // Optional Phase III: Expert Iteration
11:     if $j \,\%\, m == 0$ then
12:         Relabel $\mathcal{B}^{\text{distill}} \leftarrow \{(o_t, \texttt{SAILOR}(o_t)) | o_t \in \mathcal{B}\}$.
13:         Distill $\pi^{\text{base}} = \text{argmin}_{\pi}\, \ell(\mathcal{B}^{\text{distill}}, \pi)$.
14:     end if
15: end for
16: Return $\pi^{\text{base}}$, WM, RM, V.
```

world model is only *locally accurate* on the expert's and learner's state distributions, a policy that looks good when evaluated in such a model is guaranteed to do well in the real world. [2]

Reward Model. Intuitively speaking, we train a reward model $\texttt{RM} : \mathcal{Z} \rightarrow \mathbb{R}$ to score the latent states of our world model by how "expert-like" they are, which allows us to plan to match expert outcomes in the imagined future. More formally, we train a *discriminator* between the latent embeddings of expert and learner rollouts using the moment-matching loss proposed by Swamy et al. [2021]:

$$\ell(\mathcal{D}, \mathcal{B}, \texttt{RM}) = \mathbb{E}_{(z,a,o) \sim \mathcal{B}}[\texttt{RM}(\texttt{enc}(z, a, o))] - \mathbb{E}_{(z,a,o) \sim \mathcal{D}}[\texttt{RM}(\texttt{enc}(z, a, o))]. \tag{2}$$

As proposed by Swamy et al. [2021], we also add in a gradient penalty Gulrajani et al. [2017] to the above to stabilize training. We iteratively update the RM over the course of training to ensure that we are able to detect mistakes made by the current iteration of the composite SAILOR stack, similar to inverse RL procedures [Ziebart et al., 2008, Ho and Ermon, 2016]. While we do not dwell on the theoretical implications thereof, we note in passing that minimizing the above across a set of potential reward functions corresponds to bounding the performance difference between the expert and the learner under any of these rewards, thereby avoiding compounding errors [Swamy et al., 2021].

Critic. To enable truncated horizon rollouts in our WM of k steps, we learn a critic V that acts as a terminal cost estimate. The critic V is trained to predict bootstrapped λ-returns (Sutton et al. [1998]):

$$\mathbf{v}_t^{\lambda} = \texttt{RM}(z_t) + \gamma\Big((1-\lambda)\texttt{V}(z_t) + \lambda \mathbf{v}_{t+1}^{\lambda}\Big), \quad \mathbf{v}_k^{\lambda} = \texttt{V}(z_{t+k}). \tag{3}$$

In our experiments, we train an ensemble of 5 critic networks (Ball et al. [2023], Chen et al. [2021]). When computing terminal cost estimates, we take the mean of 2 randomly sampled critics and subtract an uncertainty penalty proportional to the standard deviation across the entire ensemble.

The Three Phases of Training a Seaworthy SAILOR. As outlined in Algorithm 2, training proceeds in three phases. In *Phase I: Warm-Start* (Lines 2-5), we perform rollouts with the base policy π^{base} to pre-fill the buffer $\mathcal{B}$, before co-training the world model WM, reward model RM, and critic V on a mixture of data from $\mathcal{B}$ and the expert demonstrations $\mathcal{D}$. In *Phase II: Online Fine-Tuning* (Lines 6-15), we instead perform rollouts with the entire SAILOR stack, periodically performing hybrid training of the WM, RM, and V. [3] There is an optimal subroutine of *Phase II*, *Phase III: Expert Iteration* (Lines 11-14), in which we distill the outputs of test-time search into the base policy to avoid the need to expend compute if the learner ends up in a similar situation in the future. More formally, we take

[2] While we do not investigate this in our experiments, the above theory still holds if the world model is trained on a wide data distribution that covers both the expert and learner's visitation distribution. Thus, a particularly interesting future direction is to train a powerful *foundation world model* [Agarwal et al., 2025, Bruce et al., 2024, Parker-Holder et al., 2024] on internet-scale robotics datasets [Khazatsky et al., 2024, ONeill et al., 2024], before plugging it into the SAILOR framework to allow for wider-ranging recovery from mistakes.

[3] One can consider training on buffer $\mathcal{B}$ as a variant of Follow the Regularized Leader [McMahan, 2011], the sort of *no-regret algorithm* one would analyze to prove guarantees about our algorithm, as in Ren et al. [2024b].

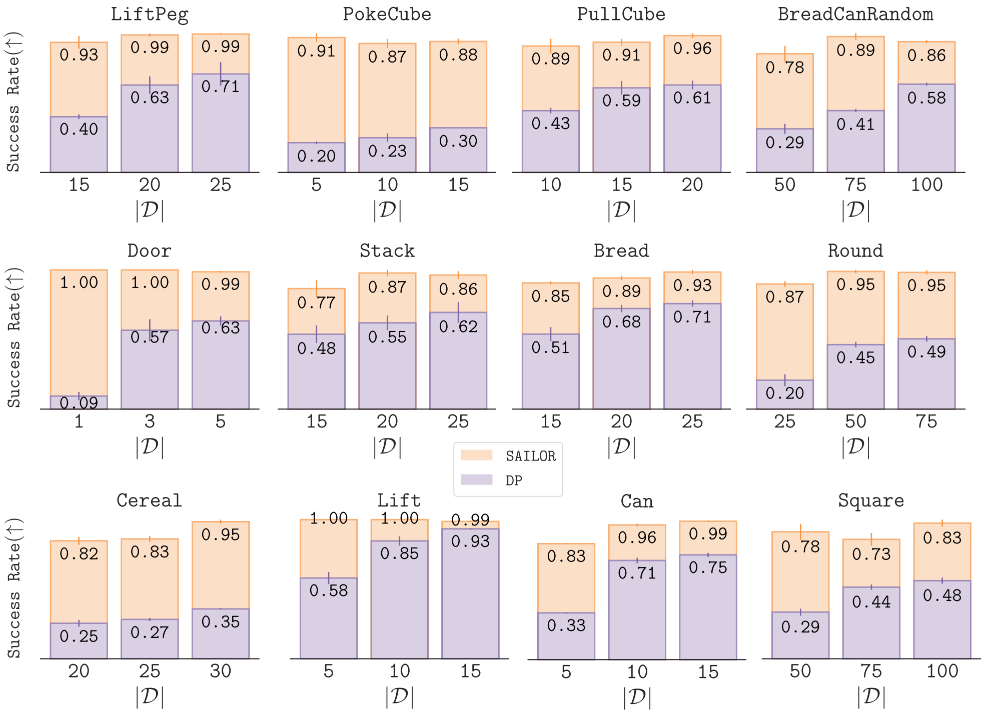


Figure 4: Across 12 visual manipulation problems from 3 benchmarks, SAILOR consistently outperforms diffusion policy (DP) trained on the same demos.

some of the most recent (o, a) pairs in buffer $\mathcal{B}$, post-hoc relabel them by calling the SAILOR stack on the observation o to generate new action labels, and fine-tune the base policy π^{base} via behavioral cloning. This can be seen as using SAILOR as an expert for a DAgger [Ross et al., 2011] update.

4 Experiments

Benchmarks. We evaluate the efficacy of SAILOR on 12 challenging visual manipulation problems. This includes 3 tasks from Robomimic [Mandlekar et al., 2021b] {Lift, Can, Square}, 6 from RoboSuite [Zhu et al., 2020] {Door, Stack, Bread, Cereal, Round, BreadCanRandom}, and 3 from ManiSkill [Tao et al., 2025] {PullCube, PokeCube, LiftPeg}. These include multi-step pick-and-place tasks (BreadCanRandom), articulated object manipulation (Door), and tool use (Square, Round, PokeCube). For Robomimic tasks, we use the provided demonstrations, while we collect our own demonstrations using a SpaceMouse controller for the other suites. To solve each task, SAILOR is provided with a set of expert demonstrations $|\mathcal{D}|$, along with a budget specifying the maximum number of environment interactions it can perform. The observation space consists of RGB images from a wrist-mounted camera and a third-person camera mounted in front of the agent, and the proprioceptive states. More details are provided in App. C and D.

Algorithms. We compare three imitation learning methods: Diffusion Policies (DP, Chi et al. [2023]) trained via BC, a model-free inverse RL method (DPPO-IRL) that directly updates DP parameters using DPPO [Ren et al., 2024a] against a learned RM with a separate encoder, and SAILOR. For evaluation, we measure the Success Rate (SR) across 50 rollouts, and report the mean and standard error obtained with 3 seeds. More details on the implementation of the methods are provided in App. A, B. All methods were trained on 1 NVIDIA 6000 Ada GPU with 48 GB of memory.

4.1 Results

Can SAILOR outperform DP trained on the same $\mathcal{D}$? Fig. 4 compares SAILOR and DP across various tasks and size of demonstration datasets. We observe that SAILOR significantly outperforms DP across

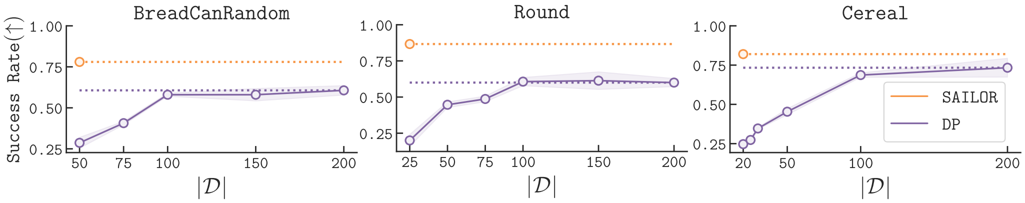


Figure 5: We see that simply scaling up the amount of data used for training DP via behavioral cloning by 5-10× often plateaus in performance and is unable to match the performance of SAILOR.

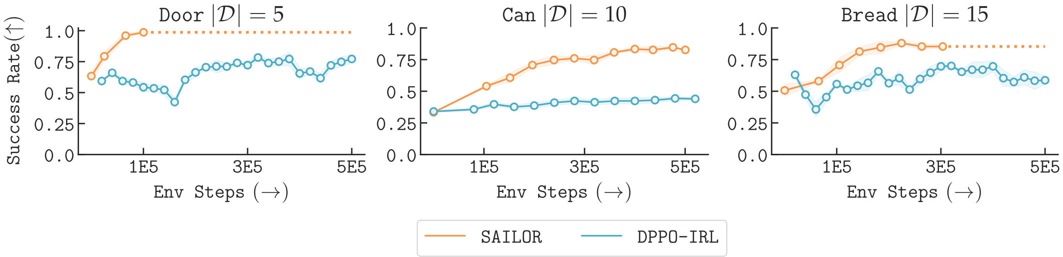


Figure 6: We find that SAILOR is significantly more interaction-efficient than a model-free inverse RL baseline that directly updates DP policy parameters via DPPO [Ren et al., 2024a].

all environments and dataset scales considered, indicating that the learning to search paradigm allows us to squeeze out significantly more from the same expert data. We find particularly large gaps in the low-data regime, reflecting how L2S inherits sample-efficiency benefits of inverse RL approaches that learn *verifiers* from human data rather than just learning policies / *generators* [Swamy et al., 2025].

Does DP catch up with more data? A natural question after viewing the preceding results might be as to whether more demonstrations could close the gap between DP and SAILOR, reflecting robot learning's current emphasis on large-scale data collection [ONeill et al., 2024, Khazatsky et al., 2024]. In Fig. 5, we observe that while DP improves with more expert data, the performance plateaus after 100 demonstrations. This means that when we scale up the number of provided demonstrations to 200, $\approx 10\times$ the amount provided to SAILOR, DP is still unable to match our method's performance.

Note further that these are expert demonstrations collected directly for the target environment. One can imagine that as practitioners choose to scale up offline data with sources such as human-video demonstrations and Internet-scale pretraining, data which is by design even less in-distribution to the task at hand, we expect a pure BC model to exhibit diminishing performance gains. SAILOR, by contrast, might be able to absorb that same large, noisy dataset into its WM and RM. Pre-training these components on broad "notions of success" may yield robust dynamics priors and value estimates that drive on-policy action distillation and keep improving the policy long after BC has plateaued.

How much real-world interaction does the WM save us? Model-based approaches are well-known to be more interaction-efficient than their model-free analogs, a benefit SAILOR inherits. In particular, each observation isn't treated as a single data point. Instead, SAILOR uses it as a seed to spawn an entire tree of counter-factual trajectories, not only slashing the number of costly real trials needed to reach a given performance, but also pruning potentially high-variance or dangerous paths before executing them on the physical hardware. To quantify the size of this benefit, we compare SAILOR to a model-free IRL algorithm DPPO-IRL that directly updates policy parameters after performing rollouts in the real world. In Fig. 6, we consistently see that even with 5× the interaction budget, DPPO-IRL is unable to match the performance of SAILOR, reflecting the importance of the WM.

Can the RM detect nuanced failures? We now explore qualitatively what our learned RM is detecting. We do this by sampling trajectories that fail to ultimately complete the task from the base DP, truncate at a prefix where failure is not yet guaranteed, and then roll out SAILOR counterfactually from these states to recover from mistakes. In Fig. 7, we see that our RM is able to detect nuanced failures that occur at various stages of a complex task (e.g. a narrowly missed tool hang after a successful grasp).

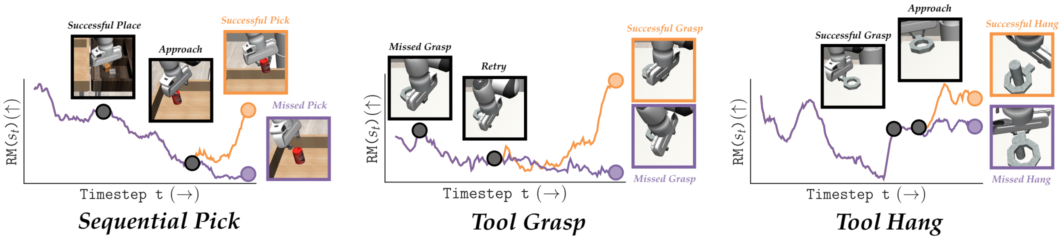


Figure 7: At different stages of multi-step task execution, we see our learned reward model RM able to identify a variety of nuanced failures and SAILOR able to counterfactually avoid them.

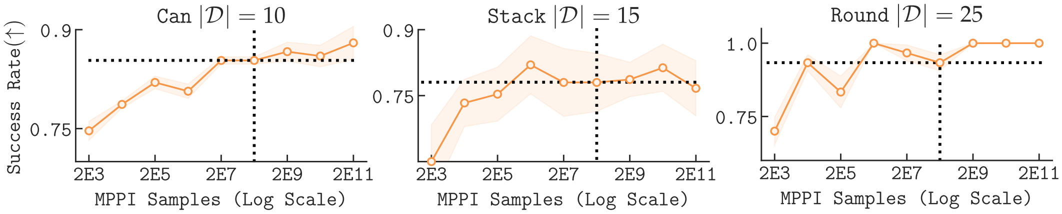


Figure 8: We use 256 samples (analogous to N in BoN) for the MPPI planning process at train-time (black cross). We see that more compute leads to higher performance up to 256 and do not see a degradation in performance even with 8× as many samples, indicating robustness to reward hacking.

How robust is SAILOR to reward hacking? A common concern with test-time scaling approaches is that with enough compute, they may "hack" a learned proxy reward model and perform worse on the ground-truth metric [Gao et al., 2023]. We explore the test-time scaling properties of SAILOR by scaling up the number of samples generated in the MPPI planning procedure, which is analogous to the N in BoN. We see in Fig. 8 that up to the number of samples used for training (256), more compute consistently leads to better performance. Furthermore, we do not see a degradation in performance even with 8× as many samples, reflecting a high degree of robustness to reward hacking.

4.2 What Matters in Learning To Search?

We now ablate the importance of several parts of the overall SAILOR pipeline.

Warm-Start. We found that allocating $\approx 20\%$ of the given interaction budget to a "warm-start phase" significantly boosts our final performance, as shown in the leftmost part of Fig. 9. We attribute this to the fact that having the WM, RM, and V accurate on π^{base}'s distribution reduces exploration in Phase II.

Hybrid WM Updates. Another component that has a significant effect on model performance is using a mix of expert and learner data to update the WM. We observe in Fig. 9, center, that this "hybrid" fitting of the WM not only has the potential to improve initial model performance (as for PokeCube), but can also improve final performance (as for Cereal), reflecting the insights of Ren et al. [2024b].

Expert Iteration. Finally, we find that adding an expert iteration subroutine can further boost model performance, as shown in the rightmost part of Fig. 9. This can be attributed to the base policy knowing how to recover from mistakes its prior iterations would have made.

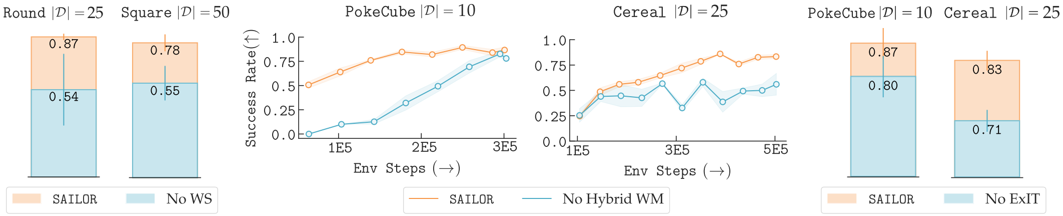


Figure 9: We ablate the impact of three components of our overall training pipeline: *warm starting*, *hybrid world-model training*, and *expert iteration*, and find that all three matter for performance.

References

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement–residual rl for precise visual assembly. *arXiv preprint arXiv:2407.16677*, 2024.

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. *Advances in neural information processing systems*, 30, 2017.

James Bagnell, Sham M Kakade, Jeff Schneider, and Andrew Ng. Policy search by dynamic programming. *Advances in neural information processing systems*, 16, 2003.

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pages 1577–1594. PMLR, 2023.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456): 885–890, 2019.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

Jonathan D. Chang, Kiante Brantley, Rajkumar Ramamurthy, Dipendra Misra, and Wen Sun. Learning to generate better than your llm, 2023. URL https://arxiv.org/abs/2306.11816.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. Learning to search better than your teacher. In *International Conference on Machine Learning*, pages 2058–2066. PMLR, 2015.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning: Learning fast without a model. *arXiv preprint arXiv:2101.05982*, 2021.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024.

Sudeep Dasari, Oier Mees, Sebastian Zhao, Mohan Kumar Srirama, and Sergey Levine. The ingredients for robotic diffusion transformers. *arXiv preprint arXiv:2410.10088*, 2024.

Nicolas Espinosa-Dice, Sanjiban Choudhury, Wen Sun, and Gokul Swamy. Efficient imitation under misspecification. *arXiv preprint arXiv:2503.13162*, 2025.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D Goodman. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*, 2024.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR, 2023.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2555–2565. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0oabwyZbOu.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2024. URL https://arxiv.org/abs/2301.04104.

Siddhant Haldar, Vaibhav Mathur, Denis Yarats, and Lerrel Pinto. Watch and match: Supercharging imitation with regularized optimal transport. In *Conference on Robot Learning*, pages 32–43. PMLR, 2023a.

Siddhant Haldar, Jyothish Pari, Anant Rai, and Lerrel Pinto. Teach a robot to fish: Versatile imitation from one minute of demonstrations. *arXiv preprint arXiv:2303.01497*, 2023b.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *ICML*, 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*, 2024.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Arnav Kumar Jain, Shiva Kanth Sujit, Shruti Joshi, Vincent Michalski, Danijar Hafner, and Samira Ebrahimi Kahou. Learning robust dynamics through variational sparse gating. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=49TS-pwQWBa.

Arnav Kumar Jain, Harley Wiltzer, Jesse Farebrother, Irina Rish, Glen Berseth, and Sanjiban Choudhury. Non-adversarial inverse reinforcement learning via successor feature matching. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=LvRQgsvd5V.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Michael Kearns, Yishay Mansour, and Andrew Y Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine learning*, 49:193–208, 2002.

Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8077–8083. IEEE, 2019.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Sergey Levine and Vladlen Koltun. Guided policy search. In *International conference on machine learning*, pages 1–9. PMLR, 2013.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021a.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2021b.

Brendan McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 525–533. JMLR Workshop and Conference Proceedings, 2011.

Russell Mendonca, Oleh Rybkin, Kostas Daniilidis, Danijar Hafner, and Deepak Pathak. Discovering and achieving goals via world models, 2021.

Diganta Misra. Mish: A self regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681*, 2019.

Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

Abby ONeill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

J Parker-Holder, P Ball, J Bruce, V Dasagi, K Holsheimer, C Kaplanis, A Moufarek, G Scully, J Shar, J Shi, et al. Genie 2: A large-scale foundation world model. *URL: https://deepmind.google/discover/blog/genie-2-a-large-scale-foundation-world-model*, 2024.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

Alexander Popov, Alperen Degirmenci, David Wehr, Shashank Hegde, Ryan Oldja, Alexey Kamenev, Bertrand Douillard, David Nistér, Urs Muller, Ruchi Bhargava, et al. Mitigating covariate shift in imitation learning for autonomous vehicles using latent space generative world models. *arXiv preprint arXiv:2409.16663*, 2024.

Nathan D Ratliff, David Silver, and J Andrew Bagnell. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, 27:25–53, 2009.

Allen Z Ren, Justin Lidard, Lars L Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. *arXiv preprint arXiv:2409.00588*, 2024a.

Juntao Ren, Gokul Swamy, Zhiwei Steven Wu, J Andrew Bagnell, and Sanjiban Choudhury. Hybrid inverse reinforcement learning. *arXiv preprint arXiv:2402.08848*, 2024b.

Stephane Ross and J Andrew Bagnell. Agnostic system identification for model-based reinforcement learning. *arXiv preprint arXiv:1203.1007*, 2012.

Stephane Ross and J Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Jonathan Spencer, Sanjiban Choudhury, Matthew Barnes, Matthew Schmittle, Mung Chiang, Peter Ramadge, and Siddhartha Srinivasa. Learning from interventions: Human-robot interaction as both explicit and implicit feedback. In *16th robotics: science and systems, RSS 2020*. MIT Press Journals, 2020.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

Wen Sun, Geoffrey J Gordon, Byron Boots, and J Bagnell. Dual policy iteration. *Advances in Neural Information Processing Systems*, 31, 2018.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Gokul Swamy, Sanjiban Choudhury, J Andrew Bagnell, and Steven Wu. Of moments and matching: A game-theoretic framework for closing the imitation gap. In *International Conference on Machine Learning*, pages 10022–10032. PMLR, 2021.

Gokul Swamy, Sanjiban Choudhury, J Bagnell, and Steven Z Wu. Sequence model imitation learning with unobserved contexts. *Advances in Neural Information Processing Systems*, 35:17665–17676, 2022a.

Gokul Swamy, Nived Rajaraman, Matt Peng, Sanjiban Choudhury, J Bagnell, Steven Z Wu, Jiantao Jiao, and Kannan Ramchandran. Minimax optimal online imitation learning via replay estimation. *Advances in Neural Information Processing Systems*, 35:7077–7088, 2022b.

Gokul Swamy, David Wu, Sanjiban Choudhury, Drew Bagnell, and Steven Wu. Inverse reinforcement learning without reinforcement learning. In *International Conference on Machine Learning*, pages 33299–33318. PMLR, 2023.

Gokul Swamy, Sanjiban Choudhury, Wen Sun, Zhiwei Steven Wu, and J Andrew Bagnell. All roads lead to likelihood: The value of reinforcement learning in fine-tuning. *arXiv preprint arXiv:2503.01067*, 2025.

Umar Syed and Robert E Schapire. A game-theoretic approach to apprenticeship learning. *Advances in neural information processing systems*, 20, 2007.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Viswesh Nagaswamy Rajesh, Yong Woo Choi, Yen-Ru Chen, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *Robotics: Science and Systems*, 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Sai Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. Chatgpt for robotics: Design principles and model abilities, 2023. URL https://arxiv.org/abs/2306.17582.

Anirudh Vemula, Yuda Song, Aarti Singh, Drew Bagnell, and Sanjiban Choudhury. The virtues of laziness in model-based rl: A unified objective and algorithms. In *International Conference on Machine Learning*, pages 34978–35005. PMLR, 2023.

Yuhang Wang, Hanwei Guo, Sizhe Wang, Long Qian, and Xuguang Lan. Bootstrapped model predictive control. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=i7jAYFYDcM.

Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2): 344–357, 2017.

Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on robot learning*, pages 2226–2240. PMLR, 2023.

Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12156–12163. IEEE, 2024.

Runzhe Wu, Yiding Chen, Gokul Swamy, Kianté Brantley, and Wen Sun. Diffusing states and matching scores: A new framework for imitation learning. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=kWRKNDU6uN.

Yilin Wu, Ran Tian, Gokul Swamy, and Andrea Bajcsy. From foresight to forethought: Vlm-in-the-loop policy steering via latent alignment. *arXiv preprint arXiv:2502.01828*, 2025b.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Mengke Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. In *The Thirteenth International Conference on Learning Representations*, 2025.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, Yifeng Zhu, and Kevin Lin. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.

Brian D Ziebart, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. 2008.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1âĂŞ2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading “NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope?

   Answer: [Yes]

   Justification: We show that our method outperforms DP in Figure 4, present DP cannot catch-up with 5-10× data in Figure 5, and show robustness of learned RM in Figure 8 and Figure 7.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We describe the architecture details of our method in Appendix A. The baselines are described in B. For the tasks where we collected demonstrations, we have added details in Appendix D.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We plan to release the data and code with the final version of the paper. We have provided details in main paper and appendix for reproducibility.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so âĂIJNoâĂİ is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have added the training and test details in Sec. 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report all our results with mean score and standard error obtained with 3 seeds.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We add details on compute details under Implementation Details in Sec 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification:

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have added citations for code and benchmarks in the main paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The code and dataset used in this work will be documented and released.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.

A Implementation Details

In this section we first describe the network architecture for each component of SAILOR in Sec. A.1 following details of training pipeline and hyperparameters in Sec. A.2

A.1 Network Architecture

Base Policy. SAILOR uses a Diffusion Policy (DP, Chi et al. [2023]) as the base policy. The base policy takes the stack of current and previous observations ($[o_{t-1}, o_t]$) as input and predicts the k-step action plan $a^{\text{base}}_{t:t+k}$. The observation o_t has proprioceptive states and RGB images from a wrist camera and a front camera. To encode RGB images, DP uses a ResNet-18 encoder initialized with the ImageNet1K_V1 weights, and the intermediate activations are passed through a Spatial Softmax layer [Mandlekar et al., 2021a] to get the final image embedding. Here, each input image uses a different copy of the encoder, and all encoded image inputs are concatenated together with the proprioceptive state and finally passed to the noise network ϵ.

The noise network is a conditional 1D U-Net, where conditioning is incorporated via FiLM modulation, following the approach of [Chi et al., 2023]. The network is conditioned on the diffusion timestep (represented as a 16 dimentional embedding generated by a sinosuidal encoding layer followed by a small MLP), and the encoded observations. The UNet consists of multiple convolution and transposed convolution layers with channels [64, 128, 256], kernel size 3, GroupNorm Wu and He [2018] for stable training and Mish [Misra, 2019] activation function. The noise network is trained to reverse the forward noising process of adding Gaussian noise at each step (DDPM, Ho et al. [2020], Nichol and Dhariwal [2021]). Lastly, DP is trained using a behavior cloning objective, formulated as a denoising task where the model learns to predict the noise added to a k-step action chunk, conditioned on the given observation and diffusion timestep.

During inference, DP uses the DDIM sampling [Song et al., 2020] with the noise network to generate the action chunk for a given observation. In this work, we use the first action generated a^{base}_t to step in the environment. The agent uses a action blending mechanism that smoothens the action at current step using the predictions from earlier observations [Dasari et al., 2024]. We found this to significantly boost the performance of DP in our experiments.

World Model. The world model used in SAILOR uses the architecture of DreamerV3 [Hafner et al., 2024]. The encoder (enc) encodes the pixel-based inputs with stride 2 convolutions to a resolution of 4×4 and state inputs are embedded with a 5-layer MLP. Note that the WM is using a separate encoder for observations as we found it to work well in our experiments. The decoder (dec) uses a transposed convolutions with stride 2 to reconstruct image observations and a 5-layer MLP to reconstruct the proprioceptive states. Note that the RGB images are downscaled to size 64×64 for training. The latent representation z_t is a combination of a deterministic recurrent state h_t and a stochastic state s_t. The deterministic state is has a GRU network with 512 dimensional hidden state. The latent state z_{t-1} and action a_{t-1} from previous time step are used to estimate the deterministic component h_t at current step. The deterministic state is fed through the dynamics model f to sample the prior stochastic state $\hat{z}_t$. Moreover, the deterministic state combined with the observation is passed through the encoder enc to get the posterior state z_t. The stochastic representation of 1024 dimensions is sampled through a vector of multinomial distributions where the gradients are backpropagated through straight-through estimators [Bengio et al., 2013] while learning.

The world model is trained by hybrid learning where the half of the batch of sequences is obtained from the demonstration $\mathcal{D}$ and other half comes from the replay buffer $\mathcal{B}$. More formally, consider a sampled batch of observation subsequences $o_{t:t+u}$, actions $a_{t:t+u}$, continuation flag $c_{t:t+u}$ (to predict the end of episode), where $t \sim \mathcal{U}\{1, \ldots, T\}$, u denotes the length of the sequence, and T denotes the length of a trajectory. We minimize a combination of a prediction loss ℓ_{pred}, a dynamics loss ℓ_{dyn} and a representation loss ℓ_{rep} over samples drawn from $\mathcal{D}$ and $\mathcal{B}$:

$$\begin{aligned}\ell(\mathcal{D}, \mathcal{B}, \texttt{WM}) = &\frac{1}{2}\mathbb{E}_{o_{t:t+u}, a_{t:t+u}, c_{t:t+u} \sim \mathcal{D}}[\beta_{\text{pred}}\ell_{\text{pred}}(\cdot) + \beta_{\text{dyn}}\ell_{\text{dyn}}(\cdot) + \beta_{\text{rep}}\ell_{\text{rep}}(\cdot)] \\ &+ \frac{1}{2}\mathbb{E}_{o_{t:t+h}, a_{t:t+h}, c_{t:t+h} \sim \mathcal{B}}[\beta_{\text{pred}}\ell_{\text{pred}}(\cdot) + \beta_{\text{dyn}}\ell_{\text{dyn}}(\cdot) + \beta_{\text{rep}}\ell_{\text{rep}}(\cdot)],\end{aligned} \quad (4)$$

where $\beta_{\text{pred}} = 1.0$, $\beta_{\text{dyn}} = .1$ and $\beta_{\text{rep}} = .5$ represent the loss weights. The loss functions are:

$$\ell_{\text{pred}}(o_{t:t+u}, a_{t:t+u}, c_{t:t+u}, \texttt{WM}) = -\sum_{h=t}^{t+u} \ln \texttt{dec}(o_h|z_h) - \ln \texttt{dec}(c_h|z_h),$$

$$\ell_{\text{dyn}}(o_{t:t+u}, a_{t:t+u}, c_{t:t+u}, \texttt{WM}) = \sum_{h=t}^{t+u} \max(1, \mathbb{D}_{\text{KL}}[\texttt{sg}(\texttt{enc}(z_h|z_{h-1}, a_{h-1}, o_h)) \| \texttt{dec}(\hat{z}_h|z_{h-1}, a_{h-1})]),$$

$$\ell_{\text{rep}}(o_{t:t+u}, a_{t:t+u}, c_{t:t+u}, \texttt{WM}) = \sum_{h=t}^{t+u} \max(1, \mathbb{D}_{\text{KL}}[\texttt{enc}(z_h|z_{h-1}, a_{h-1}, o_h) \| \texttt{sg}(\texttt{dec}(\hat{z}_h|z_{h-1}, a_{h-1}))]).$$

The prediction loss optimizes for reconstructing the observations via a Mean Squared Loss (MSE) criterion and the continuation predictor via logistic regression. The dynamics and representation losses optimize the same objective is optimized with different set of parameters: the former updates the dynamics model to predict the posterior state while the latter term ensures that the stochastic term is more predictable. Note that the dynamics and representation loss only differ in the stop-gradient term sg and the loss weight terms.

Reward Model. The reward model RM takes the latent representation z_t as input and uses a 2-layer MLP to predict a scalar value expressing the desirability of being in a state. The RM is optimized with the loss function defined in Eq. 2. The RM is updated with a gradient penalty term [Gulrajani et al., 2017] with a coefficient of 10. To stabilize learning, the RM is updated less frequently than the world model. Moreover, we do not update the parameters of the world model with the loss of the RM.

Critic Network. Similar to RM, the critic V network uses a 2-layer MLP and predicts the discounted average rewards of future states with the latent representation z_t as input. The critic is updated with the Mean Squared Loss (MSE) with target $\mathbf{v}_t^\lambda$ defined in Eq. 3:

$$\ell(z_{t:t+u}, \texttt{V}) = \sum_{h=t}^{t+u} (\texttt{V}(z_h) - \mathbf{v}_h^\lambda)^2. \tag{5}$$

Unlike RM, the critic is updated with the WM. Training the critic more frequently than RM is important to ensure it accurately estimates the average rewards and remains synchronized with the changing reward function. Similar to Hansen et al. [2024], SAILOR maintains and uses an ensemble of 5 value networks. Like Dreamer, SAILOR also maintains a slow value network to compute a slow target for the critic network and uses this as a regularizer for critic loss. The slow networks are updated with EMA over the critic parameters.

MPPI Planner. SAILOR uses MPPI for planning withing the world model in this work. The planner maintains a gaussian distribution with mean μ and diagonal covariance σ to predict the residual action $\Delta^*_{t:t+k}$. The planning procedure is described in Alg. 1 where the parameters are initialized with 0 mean and a fixed standard deviation. At each iteration, 256 action chunks are sampled and scored with the RM and V by imagining future latent with WM. The top 64 sequences with highest n-step returns is used to update the parameters μ and σ. Unrolling in the latent space allows evaluating large batches in parallel on a single GPU, and thereby makes planning efficient. After 6 iterations, an action plan $\Delta^\star_{t:t+k}$ is sampled using final parameters $\mu^\star$ and $\sigma^\star$.

A.2 Training details

With the demonstrations $\mathcal{D}$, SAILOR first pretrains the base policy– DP. The training procedure of SAILOR was outlined in Alg. 2. The pretrained DP is used to collect rollouts in the environment and uses around 20% of total environment steps. For instance, when the agent is tasked with 100K environment steps, the pretrained DP is deployed to collect for 20K transitions. SAILOR adds a small noise sampled from $\mathcal{N}(0, .1)$ to the action to promote exploration while collecting data from the environment. The agent maintains a uniform replay buffer $\mathcal{B}$ with an online queue of size 1×10^5. During the *warmstart phase*, the WM, RM and critic V are updated with a hybrid batch sampled from demonstration $\mathcal{D}$ and replay buffer $\mathcal{B}$. Note that, the gradient steps of WM and V was set to $1.5\times$ the number of transitions collected. For training stability, the RM is updated slowly and once every 100 updates to the WM. After warmstart, SAILOR uses online finetuned with batch data collection and updates over multiple rounds. At each round, SAILOR deploys the planner to collect on-policy trajectories for 3500 environment steps. The WM and V are then updated for 5000 gradient steps where the RM is updated once every 100 gradient steps of WM. After every 10 rounds, the last 64 collect trajectories are relabeled with the planner and the base policy is updated for 1000 iterations using a hybrid batch composed of relabeled data and expert demonstrations . In Table. 1, we provide the details of hyperparameters used in this work. Our agents where trained on a single NVIDIA 6000Ada GPU with 48 GB memory and takes 36 hours for training end-to-end with 500K environment steps.

B Baselines

For baselines, we compare SAILOR with the pretrained DP policy and a diffusion based IRL method (DPPO-IRL). Specifically, we apply DPPO [Ren et al., 2024a] – a model-free RL algorithm that directly updates DP parameters – with feedback from a reward model learned in the same as for SAILOR (Eq. 2). In contrast to learning a WM and learning to search in SAILOR, DPPO-IRL optimizes the policy directly using the outcomes of the learned reward model. To encode the observation, the reward model used the same encoding as the base DP of DPPO and a 2-layer MLP of width 256. We tuned the hyperparameters of the reward model (update epochs and batch size), and observed that the best version used 2 as update epochs and a batch size of 100. As for SAILOR, we used a gradient penalty term with a coefficient of 10 to stabilize learning of the reward model.

Name	**Value**
DP Pretraining	
Batch Size	256
Optimizer	AdamW
Training iterations	24,000
LR scheduler	Cosine Annealing
LR scheduler warmup steps	100
LR range	$[1 \times 10^{-4}, 1 \times 10^{-5}]$
World Model	
Replay capacity	1×10^{5}
Batch size	16
Batch length	32
Optimizer	Adam
Reconstruction Loss Scale	1.0
LR	1×10^{-4}
Demo Sampling Ratio	50%
Reward Model	
Optimizer	Adam
LR	3×10^{-5}
Gradient Penalty coefficient	10
Critic	
Discount factor γ	.997
Return lambda λ	.95
EMA regularizer	1
EMA decay	.98
Optimizer	Adam
LR	3×10^{-5}
Ensemble size	5
MPPI Planner	
Iterations	6
Samples	256
Top candidates	32
Temperature	.5
General	
Warmstart env step ratio	20%
Env steps per round	3500
Update steps per round	5000
Distillation frequency	10
Trajectories relabeled for distillation	64
Distillation steps	1000

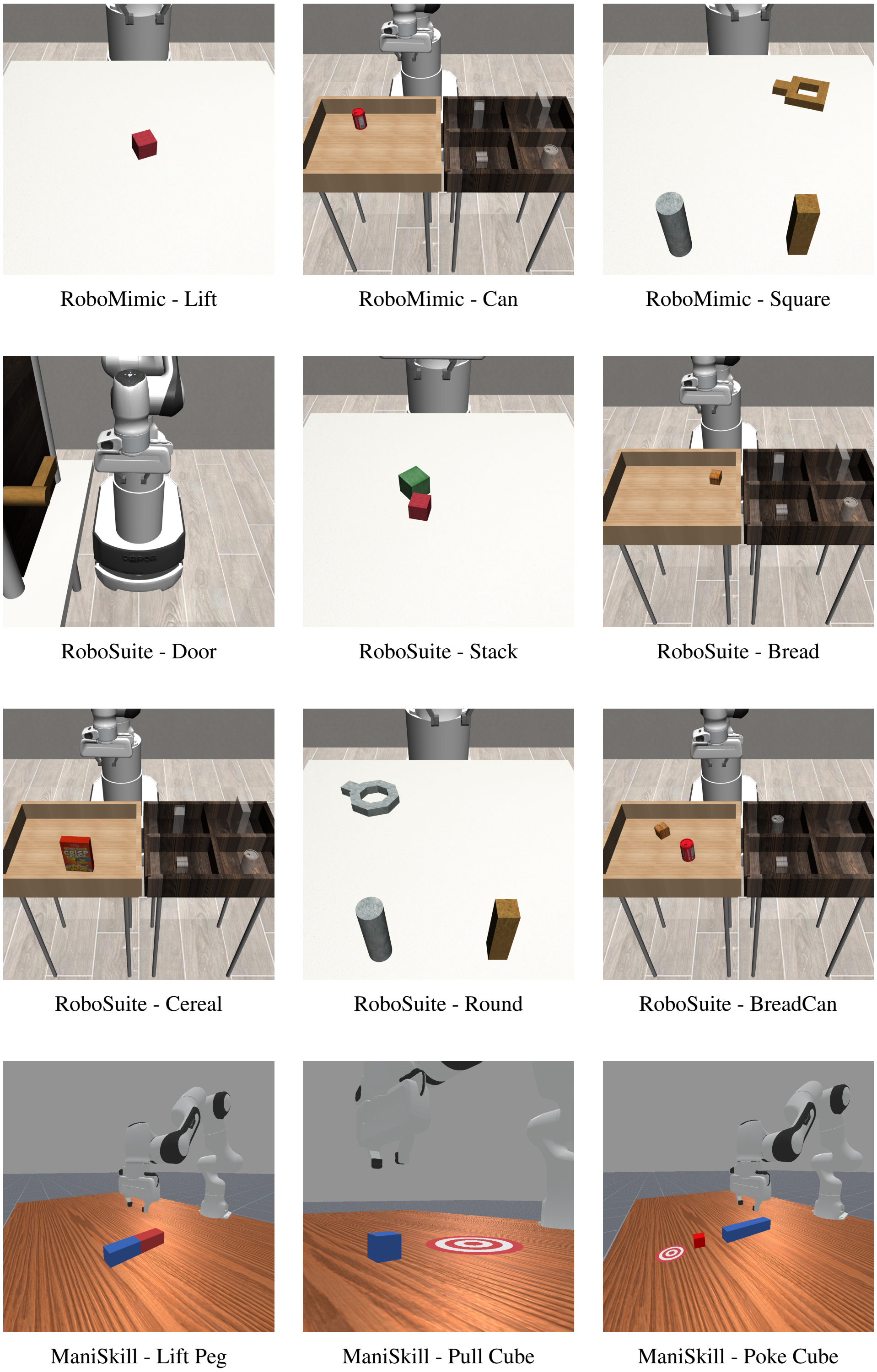
Table 1: **Hyperparameters.** For training, we recommend tuning training parameters in the General section that includes the update steps per round $\in [1000, 2000, 3500, 5000, 10000]$, distillation frequency of the base policy $\in [1, 5, 10, 50]$ and distillation steps of the base policy $\in [500, 1000, 2000]$. LR, Env and EMA denotes learning rate, environment and exponential moving average.

C Benchmarks

In this section, we describe the environments used in this work. The experiments are conducted on multiple robotic manipulation environments from RoboMimic [Mandlekar et al., 2021b], RoboSuite [Zhu et al., 2020] and ManiSkill3 [Tao et al., 2025]. Fig. 10 presents a visual image for each of the task used in this work. For each task, the agent is provided with an RGB image from the wrist camera and an RGB image from a camera in front of the agent. The agent is also given the proprioceptive states composed of position, orientation of the end effector and the position of gripper. In Table 2, we provide the episode horizon, the environment steps used for IRL training and a brief description of the task. The action space is a 7-dimensional vector with values between $[-1, 1]$. The first 6 dimensions of the action control the change in position and orientation of the end-effector and the last value opens or closes the gripper.

Domain	Task	Horizon	Env Steps	Description
RoboMimic	**Lift**	100	1×10^5	Lift Block above the desk.
	Can	200	5×10^5	Lift can and place in correct bin.
	Square	200	5×10^5	Pick square tool and insert in slot.
RoboMimic	**Door**	200	1×10^5	Pull down handle and open door.
	Stack	150	3×10^5	Lift block and place above other block.
	Bread	200	3×10^5	Lift bread and place in correct bin.
	Cereal	150	5×10^5	Lift cereal and place in correct bin.
	Round	300	5×10^5	Pick round tool and place in slot.
	BreadCan	400	5×10^5	Place both objects in respective bins.
ManiSkill	**PullCube**	50	1×10^5	Pull cube to the marked area
	PokeCube	100	3×10^5	Use tool to poke cube to marked area
	LiftPeg	150	3×10^5	Lift peg to make it stand upright

Table 2: The maximum episode length (Horizon), environment steps and description of the tasks.



RoboMimic - Lift

RoboMimic - Can

RoboMimic - Square

RoboSuite - Door

RoboSuite - Stack

RoboSuite - Bread

RoboSuite - Cereal

RoboSuite - Round

RoboSuite - BreadCan

ManiSkill - Lift Peg

ManiSkill - Pull Cube

ManiSkill - Poke Cube

Figure 10: Visual description of all tasks used from RoboMimic (top row), RoboSuite (middle rows) and ManiSkill (bottom row).

D Demonstrations

In this section, we describe the demonstrations used for learning. For RoboMimic tasks, we used the demonstrations provided in the dataset. For tasks in RoboSuite and ManiSKill, we collected upto 200 demonstrations using a 3D SpaceMouse. The camera angles were adjusted for each task to make the process intuitive to the human teleoperator. Below we provide a table of the number of demonstrations used per task for Fig. 4. For `BreadCan`, `Cereal` and `Round` we collected upto 200 demonstration for our result in Fig. 5. We plan to release the demonstrations and the codebase.

Domain	Task	Number of Demonstrations
RoboMimic	Lift	5, 10, 15
	Can	10, 15, 20
	Square	50, 75, 100
RoboMimic	Door	1, 3, 5
	Stack	15, 20, 25
	Bread	15, 20, 25
	Cereal	20, 25, 30
	Round	25, 50, 75
	BreadCan	50, 75, 100
ManiSkill	PullCube	10, 15, 20
	PokeCube	5, 10, 15
	LiftPeg	15, 20, 25

Table 3: The number of demonstrations used for each task.

E Related Work (Extended)

Reward Models. Another approach to interactive imitation learning is *inverse reinforcement learning* (IRL, [Ng et al., 2000, Syed and Schapire, 2007, Ziebart et al., 2008, Ho and Ermon, 2016, Swamy et al., 2021]), which does not require human-in-the-loop queries. In IRL, one learns a classifier that maximally differentiates learner from expert behavior and uses it as a *reward model* (RM) for RL-based policy updates. Different forms of reward models include successor features [Jain et al., 2025], score matching [Wu et al., 2025a], and optimal transport metrics Haldar et al. [2023b,a]. Unfortunately, the RL step of IRL is often rather interaction-inefficient. Recent theoretical work has argued that rather than a *global* RL procedure, a *local search* procedure[4] is sufficient for performant imitation [Swamy et al., 2023, Espinosa-Dice et al., 2025]. `SAILOR` fits within the overarching algorithmic paradigm of local search IRL but *learns to search* rather than learning a policy directly.

World Models. World models (WMs, Ha and Schmidhuber [2018]) have been an integral component of impressive RL results in a variety of domains [Hafner et al., 2020, 2021, Jain et al., 2022, Hansen et al., 2022, Hafner et al., 2024, Hansen et al., 2024, Zhou et al., 2024, Bruce et al., 2024, Parker-Holder et al., 2024, Agarwal et al., 2025], including real-world robotics [Mendonca et al., 2021, Wu et al., 2023]. While our overall algorithmic framework is agnostic to the choice of WM architecture, we use Dreamer-style world models [Hafner et al., 2020] in our experiments due to their ubiquity. Popov et al. [2024] use rollouts in a learned world model to minimize a trajectory-level divergence between the expert and the learner at train time on autonomous driving problems. In contrast, we use the world model at test-time to enable recovery from just the mistakes the learner actually makes, which might be more computationally efficient than a global search procedure [Kearns et al., 2002].

Learning to Search in Natural Language. We note briefly that the term "learning to search" is also used to describe a class of methods for *structured prediction* problems, which encompass many natural language processing tasks [Chang et al., 2015, 2023]. However, these methods usually assume access to a queryable expert policy in the vein of DAgger [Ross et al., 2011] and AggraVaTe [Ross and Bagnell, 2014]. In contrast, we make no such assumptions, and instead focus on how best to give the learner the ability to search independently at test time without additional expert guidance.

[4] By *local search*, we mean merely competing with the expert rather than the optimal policy for an adversarially chosen reward. For the former, there exist algorithms that avoid the worst-case, exponential-in-the-horizon exploration complexity of RL [Bagnell et al., 2003, Ross and Bagnell, 2014, Swamy et al., 2023].