Planning from Observation and Interaction

Tyler Han, Siyang Shen, Rohan Baijal, Harine Ravichandiran, Bat Nemekhbold,
Kevin Huang, Sanghun Jung, and Byron Boots

University of Washington

***Abstract*—Observational learning requires an agent to learn to perform a task by referencing only observations of the performed task. This work investigates the equivalent setting in real-world robot learning where access to hand-designed rewards and demonstrator actions are not assumed. To address this data-constrained setting, this work presents a planning-based Inverse Reinforcement Learning (IRL) algorithm for world modeling from observation and interaction alone. Experiments conducted entirely in the real-world demonstrate that this paradigm is effective for learning image-based manipulation tasks from scratch in under an hour, without assuming prior knowledge, pre-training, or data of any kind beyond task observations. Moreover, this work demonstrates that the learned world model representation is capable of online transfer learning in the real-world from scratch. In comparison to existing approaches, including IRL, RL, and Behavior Cloning (BC), which have more restrictive assumptions, the proposed approach demonstrates significantly greater sample efficiency and success rates, enabling a practical path forward for online world modeling and planning from observation and interaction.**

I. Introduction

You are learning how to receive a baseball pitch for the first time. Before you, a player demonstrates how to play by receiving a pitch and striking the ball far into the distance. Now your turn, you place your feet at the plate and align your body to best mimic what you observed. Then, the pitch comes and you begin your swing. You feel the inertia of the bat resist your movement; you feel your feet pivot and shift as your weight re-distributes; you observe the ballistic trajectory of the ball as it approaches you from the mound. While you may miss the first time, you have learned about the subtle challenges that your demonstrator encountered that you could not see but only experience for yourself.

Even when learning from a demonstration, such as in baseball, rarely are we capable of performing a task immediately without ever having attempted it firsthand. This reality suggests an accessibly scalable problem setting for robotics via **observational learning**, where only observations of non-expert end-users can be referenced while learning and improving with experience.

While there are many approaches to learning-from-observation (LfO) and policy improvement in robot learning, effective methods can require vast amounts of prior or online data in addition to direct action supervision (e.g. tele-operation). Often, these methods produce large, unwieldy models trained using domain knowledge, embodiment-specific data, or expert operators [49, 34]. As a consequence, extensions of the approach to complex robotic embodiments and demonstrations from novice end-users remain challenging.

Experiment videos, code, and more on project website at: `https://uwrobotlearning.github.io/mpail2/`

Recently, Model Predictive Adversarial Imitation Learning (MPAIL) was proposed as a planning-based Inverse Reinforcement Learning (IRL) algorithm for observational learning without access to reward information [16]. MPAIL aims to learn behaviors from scarce amounts of demonstration and online data. By deploying its learned components (reward and value functions) online in planning, MPAIL demonstrates high real-world robustness and sample efficiency in navigation in comparison to policy-based LfO methods.

Like other IRL methods, MPAIL remains limited in its scalability to more complex observations and tasks without further modifications; for instance, MPAIL employs state-based dynamics, purely random plan sampling, on-policy methods for improvement, and massively simulated interactions. Motivated by MPAIL's demonstrated potential in real-world data efficiency and robustness for observational learning, this work presents MPAIL2, which aims to scale Inverse Reinforcement Learning from Observation (IRLfO) by introducing off-policy learning, multi-step policy optimization, and world modeling.

This work presents the following contributions to the best of our knowledge: (1) *this is the first work in IRLfO demonstrated purely in the real-world from scratch without prior modeling assumptions.* (2) By enabling real-world IRLfO for visual manipulation, we find that this paradigm unlocks significant interaction efficiency improvements over common baselines in visual manipulation. *Where baseline methods in RL with demonstrations see no success after over an hour of real-world training, MPAIL2 sees consistent success within 40 minutes or less.* Evaluations include comparisons to RLPD and Behavior Cloning (Diffusion Policy), which operate under less general scopes and are provided greater data access [3, 7]. (3) Finally, MPAIL2 significantly improves on interaction efficiency while learning representations towards world modeling. *This is the first work in RL with demonstrations exhibiting online transfer learning of new tasks purely in the real-world from scratch.*

II. Related Work

Methods in Learning from Observation (LfO) learn to perform tasks from observation alone. Along with its implications in learning from videos, LfO is perhaps the closest problem setting to robot learning with minimal to no direct supervision. Since, it is theoretically unencumbered by requirements of hand-designed rewards used in Reinforcement Learning (RL) and human teleoperation used in Imitation Learning.

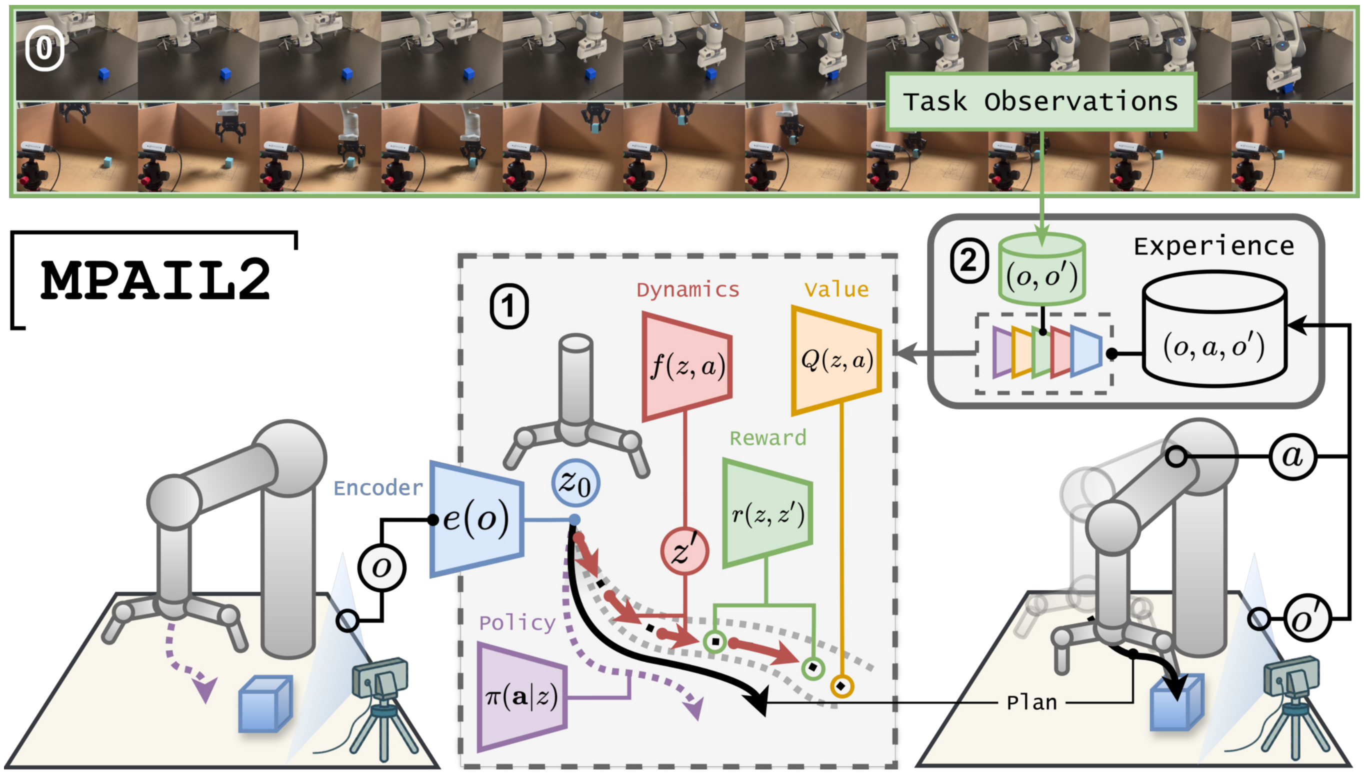


Fig. 1. Overview of MPAIL2. **(0)** The learner observes a task demonstration and stores the observations before training. **(1)** The learner observes the world, and encodes the observation into its current latent state, $z_0 = e(o)$. The learner's policy π suggests reactive, possibly suboptimal actions (purple dotted lines). Before executing any actions, the learner predicts and evaluates the world at future latent states to derive a plan by using the pictured component models. This process involves: (a) randomly sampling action sequences (i.e. plans) for evaluation (not pictured); (b) predicting trajectories implied by sampled plans using the dynamics model $z' = f(z, a)$ (dotted lines); (c) and finally, predicting the total return of a plan by adding up the rewards of its implied state-transitions $r(z, z')$ with terminal value bootstrapping $Q(z, a)$. **(2)** An action a is executed according to plans with higher returns; the world's response to the action is observed o'; and the interaction (o, a, o') is accumulated in the learner's experience. Learning occurs by updating all of the component models over the collective experience. Initial task observations are used only to update the reward model.

However, in practice, many LfO approaches require hours of demonstration *and* domain data for learning seconds-long tasks [26, 49, 8, 40]. Domain-specific representations and pre-trained encoders can help reduce data requirements but incidentally can reduce method generality. For instance, in [49], a visual hand detector and state estimator are required for demonstrating kinematic plans for a manipulator's low-level controller. Additionally, the generalization of these approaches to broader settings like mobile navigation or non-quasi-static domains is not straightforward.

Inverse Reinforcement Learning from Observation (IRLfO) is an approach to LfO which is known to significantly reduce the required demonstration quantity, often to just one-to-few demonstrations [16, 9, 26, 10]. However, in reinterpreting demonstrations as reward, IRL trades off demonstration efficiency for interaction efficiency, often requiring impractical amounts of online interaction for real-world learning [16, 9, 10, 45].

Of many approaches to improving sample efficiency in RL, this work investigates world modeling and Model-Based Reinforcement Learning, motivated by the discussions in Section I. In particular, *planning-based* learners in RL have demonstrated remarkable potential and efficiency in simulated experiments [17, 21]. For real-world robot learning, planning-based learners have also been shown to be substantially more robust than policy networks [16].

To the best of our knowledge, no prior works in planning-based learners have investigated the setting of real-world IRL. Notable is the model-based IRL work in [43] demonstrating visual reaching in the real-world. Further, MPAIL is the only IRLfO algorithm demonstrating sufficient robustness for real-world deployment without assumptions upon the inferred reward [16]. Notable related works assuming access to demonstration actions or reward include [24, 9, 43].

III. Model Predictive Adversarial Imitation Learning 2

A. Problem

Observations of task demonstrations are provided to the learner. Its goal is to reproduce these observations as best it can. Consider the Partially Observable Markov Decision Process (POMDP) to define the environment using the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, R)$ denoting the *state space, action space, observation space, transition model*, and *reward* respectively. **Beyond task observations, access to the true reward** R **is not assumed in any further capacity**; for instance, goal and terminal observations and states are not assumed since they imply knowledge of R and $\mathcal{T}$ [23]. Additional

Algorithm 1 MPAIL2

```
Require: $\mathcal{D} \subset \mathcal{O} \times \mathcal{O}$                          ▷ Task Observations
         $e_\omega : \mathcal{O} \to \mathcal{Z}$                                        ▷ Encoder
         $f_\psi : \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$                      ▷ Dynamics
         $r_\theta : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$                     ▷ Inferred Reward
         $Q_\zeta : \mathcal{Z} \times \mathcal{A} \to \mathbb{R}$                      ▷ Value
         $\mathbf{a}_t \sim \pi_\phi(\cdot|z_t)$                                          ▷ Policy
         $\mathbf{a}_t \sim \widehat{\Pi}(\cdot|\mathbf{a}, z_t\,;\, f_\psi, r_\theta, Q_\zeta, \pi_\phi)$   ▷ Planner
         $\mathcal{B} := \{\}$                                                            ▷ Replay Buffer
 1: while learning do
 2:     Interact using planner (Algorithm 2)
            $\mathcal{B} \leftarrow \mathcal{B} \cup \{(o_t, a_t, o_{t+1})\}_{t=1}^{T}$                    (2)
 3:     for updates per episode do
 4:         Sample trajectories and task observations
                $\{(o_t, a_t, o_{t+1})\}_{t=1}^{H} \sim \mathcal{B}\,,\ (o, o') \in \mathcal{D}$          (3)
 5:         Update Encoder and Dynamics (14)
                $\mathcal{L}_{e,f}(\omega, \psi) = \mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t'=t}^{t+H} \rho^{t'-t} \left\|\hat{z}_{t'} - \mathrm{sg}\left(z_{t'}\right)\right\|_2^2 \,\middle|\, \boldsymbol{\tau}_t\right]$   (4)
 6:         Update Inferred Reward
                $\mathcal{L}_r(\theta) = \mathbb{E}_{(z,z')\sim\boldsymbol{\tau}}[r] - \mathbb{E}_d[r] + \beta\,\mathrm{GP}\left(r, \boldsymbol{\tau}, d\right)$   (5)
 7:         Update Value
                $\mathcal{L}_Q(\zeta) = \mathbb{E}_{(z,a,z')\sim\boldsymbol{\tau}, a'\sim\pi_\phi(\hat{\mathbf{a}}'|z')}\left[\left(q_t - \bar{G}_t^\lambda(\hat{\boldsymbol{\tau}}_t^\pi)\right)^2\right]$   (6)
 8:         Update Policy (14)
                $\mathcal{L}_\pi(\phi) = -\mathbb{E}_{\hat{\tau}}\left[G_t^\lambda(\hat{\boldsymbol{\tau}}_t^\pi)\right]$   (7)
 9:     end for
10: end while
```

reward and termination knowledge is orthogonal to our setting and subsequently points to promising future work. Assumed known are task observations, $\{\mathbf{o}_{1:T_i}\} \in \mathcal{D}$ with variable task lengths T_i. The learner is to infer a reward $r \approx R$ using these examples that would be maximized for task observations relative to other potential policies,

$$\arg\max_{r\in\mathcal{R}} \mathbb{E}_{\mathcal{D}}[r] - \max_{\pi\in\Pi} \mathbb{E}_\pi\left[r\right]. \tag{1}$$

Recognizable as apprenticeship learning [1], this ambiguous objective will be better algorithmically defined in the following section using components of our proposed world model.

In [16], MPAIL is realized with state-based dynamics as well as on-policy reward and value optimization which renders the algorithm too sample inefficient for real-world learning. Here, we re-formulate and design MPAIL2 for off-policy training and latent planning using the following components,

$$\begin{aligned} &\text{Encoder} && z_t = e_\omega(o_t) && (8)\\ &\text{Dynamics} && \hat{z}_{t+1} = f_\psi(z_t, a_t) && (9)\\ &\text{Inferred Reward} && r_t = r_\theta(z_t, z_{t+1}) && (10)\\ &\text{Value} && q_t = Q_\zeta(z_t, a_t) && (11)\\ &\text{Policy} && \hat{\mathbf{a}}_{t:t+H} \sim \pi_\phi(\cdot|z_t) && (12)\end{aligned}$$

As outlined in Algorithm 1, each component is updated separately from other components in each learner update. The remainder of this section provides details on the training losses of these components.

B. *Encoder & Dynamics*

The encoder and dynamics model are trained jointly using only sequences of transitions (i.e. trajectories),

$$\boldsymbol{\tau}_t := \{z_{t'}, a_{t'}, z_{t'+1}\}_{t'=t}^{t+H}. \tag{13}$$

The time subscript is dropped in some cases, such as in Equation (5), when distinguishing the beginning index is inconsequential to the loss. Transition sequences with length H are the building blocks to our learning objectives and are henceforth assigned a shorthand notation for convenience; bolded symbols denote temporal sequences of length H beginning at time t. For instance, $\mathbf{a}_t := \mathbf{a}_{t:t+H-1}$ denotes an action sequence from time t to $t + H - 1$.

We also distinguish *predicted* from *experienced* objects using $\hat{\ }$ (hat) to indicate a result of a *predicted* latent. For instance, $\hat{r}_t = r_\theta(\hat{z}_t, \hat{z}_{t+1})$ and $\hat{q}_t = Q_\zeta(\hat{z}_t, \hat{a}_t)$ are predicted whereas $r_t = r_\theta(z_t, z_{t+1})$ and $q_t = Q_\zeta(z_t, a_t)$ are obtained from experienced and encoded data from the replay buffer.

Terminology. Given an initial observation o_t and plan $\mathbf{a}_t$, let the *predicted trajectory* be a sequence of transitions,

$$\hat{\boldsymbol{\tau}}_t(\mathbf{a}_t) := \{\hat{z}_{t'}, a_{t'}, \hat{z}_{t'+1}\}_{t'=t}^{t+H} \tag{14}$$

where

$$\hat{z}_{t+1} = f_\psi(\hat{z}_t, a_t), \quad \hat{z}_t := z_t = e_\omega(o_t) \tag{15}$$

Let $\hat{\boldsymbol{\tau}}_t^\pi := \hat{\boldsymbol{\tau}}_t(\hat{\mathbf{a}}_t)$ further denote a policy-planned trajectory in which actions are sampled from the policy π.

As the dynamics model is deterministic, *plan* and *trajectory* are used interchangeably. For instance, *policy plan* is equivocal to *policy-planned trajectory*. Likewise, the encoder allows direct sampling of latent *experienced* trajectories from the buffer $\boldsymbol{\tau}_t \sim e_\omega(\mathcal{B})$ and expert data $\{z, z'\} \sim e_\omega(\mathcal{D})$ where each $z_t = e_\omega(o_t)$, used in Algorithm 1, eq. (3).

More complex, probabilistic models are not evaluated in this work but are orthogonal and deserve discussion. For instance, consider the Recurrent State Space Model (RSSM) [15], as used in [21]. The RSSM conditions rewards on the learner's recurrent hidden state. Thus, imitation learning using the RSSM, along with other popular RL methods like RLPD [3] are enabled by assuming access to expert actions but remain valuable future work in settings with access to action labels.

Objective. The optimization objective of the dynamics model is a self-supervised latent prediction loss as in Equation (4). The encoder and dynamics model depends only upon observed trajectories. Downstream task-related losses are not used to update the latent representation or dynamics. This has valuable implications on the ability for only the encoder and dynamics model to be pre-trained or inherited and continually transferred across tasks, since its predictions are entirely independent of any downstream task-related heads.

C. Inferred Reward

Adversarial Imitation Learning (AIL) algorithms compute Equation (1) by alternating updates to the reward r and policy π in a game-theoretic manner. As a result, AIL has been shown to be highly sample efficient among IRL algorithms [20]. Though, in practice, it is often unstable to train. On-policy updates, where updates for both reward and policy depend upon only the previous policy's interactions, exacerbate this instability and worsen interaction efficiency [23].

Instead, the reward can be trained using all experienced interactions in an off-policy manner, which serves two purposes for MPAIL2. In addition to improved sample efficiency, an off-policy reward is required to maintain its accuracy in all states, whereas an on-policy reward converges meaninglessly to a constant [11]. Reward accuracy in a greater coverage of states is necessary for planning, which seeks to re-optimize the policy at each step by randomly sampling actions online in the environment. However, we remark that random online sampling is a consequence of the chosen planner, MPPI [50]. We hypothesize that reward attention can be better focused under a more suitable planner and is promising future work.

The objective of the inferred reward is given by

$$\arg\max_{r\in\mathcal{R}} \mathbb{E}_{\mathcal{D}}[r(z,z')] - \mathbb{E}_{\mathcal{B}}\left[r(z,z')\right] \ . \tag{16}$$

Supervision of the inferred reward r_θ arrives from the principle that the expert performs optimally in expectation when compared with "other policies". Observe that "other policies" are represented by the off-policy replay buffer $\mathcal{B}$. In practice, this reward objective requires additional regularization for stability. We find the Gradient Penalty as used in the Wasserstein GAN formulation [2] to be the most effective,

$$\mathrm{GP}\left(r,\boldsymbol{\tau},d\right) = \mathbb{E}_{(\tilde{z},\tilde{z}')}\left[\left(\left\|\nabla_{(\tilde{z},\tilde{z}')} r(\tilde{z},\tilde{z}')\right\|_2 - 1\right)^2\right], \tag{17}$$

where $\tilde{z}$ are uniformly sampled linear interpolations between learner $(z,z') \in \boldsymbol{\tau}$ and expert $(z,z') \in \mathcal{D}$ latent state-transitions [13].

The inferred reward r_θ forms the foundation of the value and policy in the same manner in which it does for RL methods.

D. Value

The value Q is trained in an entropy-regularized, off-policy manner [14] using a λ-return target $\bar{G}_\lambda$, defined in the next section. Additional learning stabilization methods such as ensembling [6] and polyak updates [14] are employed as previously contributed for latent-planning in the RL setting under known reward [17]. The overbar in the return $\bar{G}_\lambda$ indicates that it employs the slower, offline target value $\bar{Q}$.

In prior work, direct value function optimization using demonstration data eliminates the need for unstable, adversarial training of the reward [12]. However, this approach invariably requires access to action labels but points to promising future work in this setting.

E. Policy

Planners can be computationally slow due to the sequential, auto-regressive nature of dynamics rollouts. By requiring additional inferences from single-step policies $\pi(a|z)$ in tandem, existing work in planning-based learners often double the computational demand during online planning [18].

Towards faster, real-time control, we choose to solve for a multi-step policy $\pi(\mathbf{a}_{t:t+H-1}|z_t)$. This is motivated by two characteristics of planning-based learners: (1) online improvement of a policy via planning presumes unreliable policy output. (2) the planner often does not choose plans generated by the policy, especially early in training. Thus, we hypothesize that the primary role of a policy network in planning-based learners is to support off-policy value optimization and to seed online planning [17]. Given this intuition, the multi-step policy aims to leverage the high in-distribution performance and memorization capacity of policy networks to save on online computation.

The multi-step policy $\pi_\phi(\mathbf{a}_t|z_t)$ therefore yields full plans rather than immediate single-step actions. It is optimized over a TD(λ)-*like* return along H-length predicted rollouts [44]. At a given time t, the policy objective is

$$\max_{\pi\in\Pi} \mathbb{E}_{\hat{\mathbf{a}}_t\sim\pi(\cdot|z_t)}\left[G_t^\lambda(\hat{\boldsymbol{\tau}}_t^\pi)\right] \tag{18}$$

where the return G_t^λ is computed by functionally recursing through the policy plan $\hat{\boldsymbol{\tau}}_t^\pi := \hat{\boldsymbol{\tau}}_t(\hat{\mathbf{a}}_t)$,

$$\begin{aligned} G_t^\lambda(\hat{\boldsymbol{\tau}}^\pi_{t':t+H}) := \lambda \hat{q}_{t'} + (1-\lambda)\left[\hat{r}_{t'} + \gamma G_t^\lambda(\hat{\boldsymbol{\tau}}^\pi_{t'+1:t+H})\right] \\ -\alpha \log \pi_\phi(a_{t'}|z_t) \end{aligned} \tag{19}$$

beginning with $t' := t$ and terminating with $t' = t+H$ such that $G_t^\lambda(\hat{\boldsymbol{\tau}}^\pi_{t+H:t+H}) := \hat{q}_{t+H}$. α is the temperature coefficient as in [14] and is automatically tuned such that the policy remains at a target entropy at each action step. We can interpret $\lambda \in [0,1]$ as how much the current return estimate should depend on the current Q-value rather than the predicted model-based return thereafter. As a result, this objective supervises every predicted action along the multi-step policy with a spectrum of data-driven model-free and model-based return [4]. Finally, as in [14], the policy's actions are entropy-regularized at each step along the plan. *All terms in this objective are fully differentiable with respect to each action step.*

F. Planner

As briefly discussed in Section III-E, we hypothesize that while policies (i.e. modules which predict actions) operate remarkably when in-distribution, they generalize poorly without

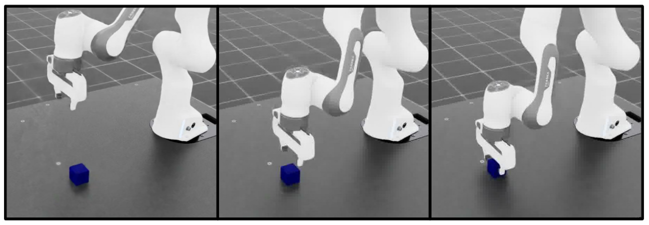
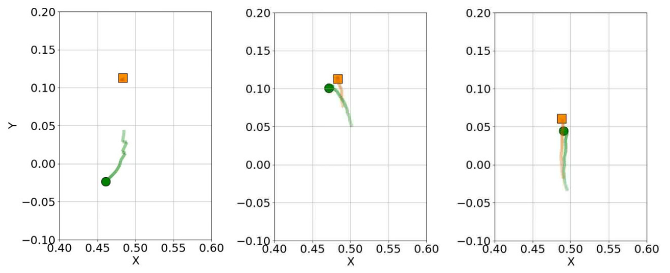



Fig. 2. *Top.* Simulation of MPAIL2 on the Block Push (state) task at three time steps along an episode. *Bottom.* Predicted plans in the XY plane (top-down) for the next one second at each time step. End-effector plans are drawn in green. Block plans are drawn in orange. Block trajectories, static in the leftmost frame, become dynamic as the robot approaches and makes contact. These predictions show how the agent gradually learns to plan over physically causal relationships, like contact. *Note: planning occurs in latent space. This visualization is made possible by a separately trained decoder.*

the ability to refine, or optimize, their decisions iteratively online. Much prior work has demonstrated that iterative methods, like MPAIL [16], Diffusion Policy [7], and more [35], are powerfully efficient, implicit, action representations in robot learning. Under the world modeling perspective, this online refinement arrives in the form of a planner.

As online model-based policy optimization, planning invites endless approaches which are orthogonal to this work. Here, we have chosen Model Predictive Path Integral (MPPI) [50] due to its simplicity, extensibility, and familiarity in other planning-based learners [16, 21, 17]. MPPI aims to solve for the next multivariate Gaussian distribution over plans,

$$\arg\max_{\boldsymbol{\mu}_t,\boldsymbol{\sigma}_t} \mathbb{E}_{\mathbf{a}_t\sim\mathcal{N}(\boldsymbol{\mu}_t,\boldsymbol{\sigma}_t)}\left[\gamma^H \hat{q}_{t+H} + \sum_{t'=t}^{H-1}\gamma^{t'-t}\hat{r}_{t'} \;\middle|\; \hat{\boldsymbol{\tau}}(\mathbf{a}_t)\right] \tag{20}$$

where the predicted rewards $\hat{r}$ and values $\hat{q}$ are used to estimate the predicted return of each plan $\hat{\boldsymbol{\tau}}(\mathbf{a})$.

Complete algorithmic detail of MPAIL2's planning procedure during training, as well as any modifications from vanilla MPPI, is given in Algorithm 2. Figure 1 illustrates the algorithmic procedure during planning.

IV. Experiments

Self-prediction is a capability demonstrated previously by MPAIL through navigation experiments [16]. However, to robustly plan in the natural world, an agent must also be capable of predicting behaviors external to itself, such as objects or other agents (Fig. 2). Experiments in this work turn towards manipulation to validate *world* modeling capabilities of MPAIL2 as the learner must understand how its actions affect external, partially observable, and partially controllable states.

Scope	Known Reward	Task Actions	Task Observations	Online
IRL(fO)			✓	✓
RLPD	✓	✓	✓	✓
BC		✓	✓	

TABLE I. Summary of Scope Assumptions. The IRL from Observation setting does not make any assumptions on task reward, and does not require actions in the demonstration data, unlike other common approaches for robotic manipulation, like RL and BC.

The following experiments aim to validate MPAIL2's design choices towards real-world sample efficiency by comparing the approach with other IRL methods. We also compare to state-of-the-art baselines in Reinforcement Learning (RL) and Behavior Cloning (BC) that are frequently chosen for manipulation tasks similar or equivalent to those investigated here. In Section IV-C, MPAIL2's capabilities as a world model is investigated through a knowledge transfer experiment where two real-world tasks are trained in sequence.

Disclaimer: while easily interpretable, evaluations which use demonstration data generated from synthetic policies, converged under known reward, correlate poorly with evaluations using imperfect human demonstrations [33]. The experiments in this work use only human-operated demonstrations, and evaluation metrics are designed to best capture human intent. Explicit reward, metric, and success definitions where applicable can be found in Section B. Evaluation of a method in a task also include performance metrics of two model checkpoints: *Best* indicates the best performing model checkpoint given evaluation statistics over a training run; *Last* indicates the last model checkpoint in training. In applications, it may be impossible to extract the "best" model when clear performance measures are not readily available. Thus, *last* model evaluations help better quantify performance in the event that training should be run indefinitely.

Baselines

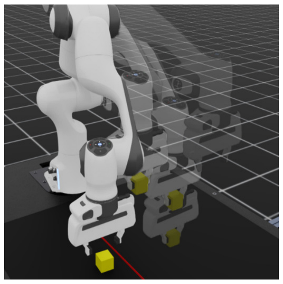
We compare and evaluate the following baselines, organized by their problem scope (summarized in Table I):

1) IRL Methods as Ablations: Methods like **AIRL** [11], **MAIRL** [43], and **DAC** [23] fall within the same scope as MPAIL2 by adhering to the IRL setting, where rewards are not known. Though these methods are not originally proposed for the observation-only setting, we follow [46] to trivially extend them by learning rewards over (latent) state-transitions without actions. More complex IRL extensions, such as [38, 48], have proposed modifications to model classes or losses to improve sample efficiency or stabilization. These extensions are orthogonally applicable to all AIL methods, including MPAIL2, and are thus excluded to reduce experimental complexity but remain promising future work.

MPAIL2 makes several well-established but indispensable modifications (in Section III and in modern work [17, 3, 27]) towards real-world sample-efficiency. To the best of our knowledge, these modifications have not previously been demonstrated in prior work for IRL in simulated nor real-world settings. Nonetheless, other IRL methods need be evaluated

(a) Pick and Place (Real)

(b) Push (Real)


(c) Pick and Place (Sim)

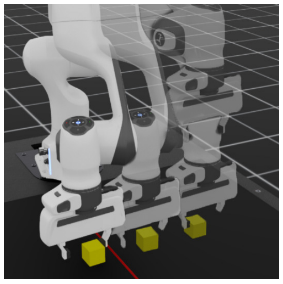

(d) Push (Sim)

Fig. 3. Overview of evaluation tasks. We evaluate our method and baselines on 4 tasks - 2 in sim and 2 in real. The Pick and Place tasks (a,c) involve reaching the block, grasping it, lifting and placing it beyond a target line. The Push tasks (b,d) involve reaching the block and pushing it beyond a target line.

Implementation	Planning	Model	Off-Policy	Close to
MPAIL2	✓	✓	✓	
MPAIL2 [−P]		✓	✓	MAIRL [43]
MPAIL2 [−PM]			✓	DAC [23]
MPAIL2 [−PMO]				AIRL [11]

TABLE II. Summary of IRL Ablations. As discussed in Section IV, IRL baselines in this work benefit immensely from techniques and designs used in MPAIL2 and real-world RL. For precision, baselines like MAIRL, DAC, and AIRL are alternatively viewed as ablations on MPAIL2 without specific components.

under the same modifications to draw meaningful methodological conclusions. To avoid any unintended miscrediting of prior work and for clarity, Table II reports each baseline under its closest taxonomic category. We emphasize that the baseline evaluated corresponds to the particular implementation adopted in this work.

MPAIL2 [−P] (also [−P]) denotes MPAIL2 without online planning. [−PM] denotes MPAIL2 without online planning nor model-based methods. [−PMO] denotes MPAIL2 without online planning, model-based methods, nor off-policy replay. Table II summarizes IRL methods evaluated in this work and their differences. All implementation details, as well as precise divergences of baselines from previous work, is clarified in Section C.

2) Reinforcement Learning with Prior Data (RLPD): When reward and actions are provided, RLPD has previously demonstrated remarkable sample-efficieny in real-world training from scratch. We compare MPAIL2's sample efficiency (without reward and action access) to state-of-the-art RL implementation, **RLPD** [3], using the known *dense* reward where applicable as well as action-labeled demonstrations. RLPD employs several well-established design choices for sample-efficient real-world RL with demonstrations, some of which MPAIL2 has drawn upon. Its implementation is provided by [27]. Further details and discrepancies are shown in Section B.

3) Behavior Cloning (Diffusion): Diffusion Policy (DP) is a highly successful implicit, and similarly iterative, policy approach to **Behavior Cloning (BC)** when action labels are provided, only offline training is allowed, and long inference times are permissible [7]. DP evaluations contextualize ideal performance when In-Distribution (ID) to the demonstrations. BC is provided unlimited offline updates until convergence.

A. Simulated Sample-Efficiency Experiments

Simulated experiments conducted in IsaacLab [32] are image-based unless otherwise stated. As in popular manipulation setups [27, 24, 34], image-based task observations include arm proprioception and two camera images: one camera mounted on the wrist of the robot and the other camera affixed to the table and raised to some height. In simulated experiments, the initial configuration ranges are consistent: a 4×4 cm block is placed uniformly at random within a 10×6 cm region.

Hyperparameters are kept constant for MPAIL2 and its IRL ablations across all experiments. Metrics are reported over five seeds for statistical significance. Detailed observation space, action space, and other settings are given in Section B. Fig. 4 shows the performance of all methods on the three tasks.

1) **Block Push.** The arm navigates to the cube until it is within the gripper. While ensuring that the cube is stably within the gripper, the arm moves in the $-y$ direction, pushing the cube along. The arm stops once the cube has reached $y = -0.1$. 27 demonstrations are provided. Failure often results from moving the cube without accounting for its unstable pushing dynamics. The agent must constantly re-adjust whenever the cube is perturbed off-center to the gripper.
2) **Block Push (state).** In the state-based variant, only the observation space changes such that the cube's 3D position and orientation is provided in place of the camera images. 27 demonstrations are provided.
3) **Pick-and-Place.** The arm navigates to the cube until it is within the gripper, where it closes to grasp the block. With the block grasped, the arm first raises directly upwards 15 cm, then moves in the $-y$ direction beyond the target threshold, and finally lowers the block to the table. The gripper releases the block. 30 demonstrations are provided.

B. Real-World Sample-Efficiency Experiments

As [−PMO] (AIRL) does not show any improvement within the simulation experiments, we omit the baseline from real-world experimentation as purely model-free method performance is better represented by [−PM].

Random resets are experimentally controlled for via programmatic management. Resets are not selectively chosen by

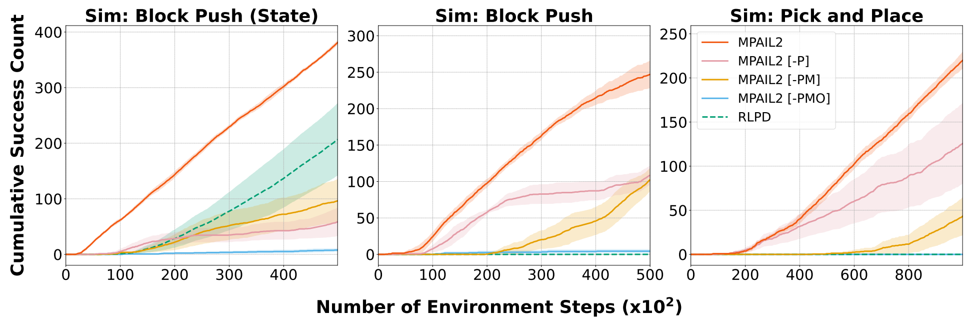


Fig. 4. Cumulative successes in simulated experiments. Offline results of BC are shown in Table III.

Models	Last Success (%)	Best Checkpoint #	Best Checkpoint Success (%)
Sim: Block Push (State)			
MPAIL2	**88.0** ± **6.0**	500	**88.0** ± **6.0**
MPAIL2[-P] (MAIRL)	32.0 ± 19.4	500	32.0 ± 19.4
MPAIL2[-PM] (DAC)	25.0 ± 10.7	200	43.0 ± 21.0
MPAIL2[-PMO] (AIRL)	8.0 ± 8.0	300	8.0 ± 8.0
RLPD	60.4 ± 12.5	500	60.4 ± 12.5
BC	60.2 ± 1.6	—	60.2 ± 1.6
Sim: Block Push			
MPAIL2	**73.0** ± **13.3**	300	**82.0** ± **5.1**
MPAIL2[-P] (MAIRL)	35.0 ± 15.2	200	49.0 ± 21.8
MPAIL2[-PM] (DAC)	59.0 ± 20.8	500	59.0 ± 20.8
MPAIL2[-PMO] (AIRL)	0.0	300	16.0 ± 16.0
RLPD	0.0	500	0.0
BC	31.6 ± 2.0	—	31.6 ± 2.0
Sim: Pick and Place			
MPAIL2	**50.0** ± **8.4**	300	**58.0** ± **14.5**
MPAIL2[-P] (MAIRL)	41.0 ± 21.4	500	41.0 ± 21.4
MPAIL2[-PM] (DAC)	33.0 ± 19.6	500	33.0 ± 19.6
MPAIL2[-PMO] (AIRL)	0.0	100	0.0
RLPD	0.0	500	0.0
BC	46.6 ± 2.3	—	46.6 ± 2.3

TABLE III. Comparison of the performance of our method against baselines on simulation tasks. *Best* indicates the best performing model checkpoint given evaluation statistics over a training run; *Last* indicates the last model checkpoint in training. Evaluation metrics are computed over 100 trials per checkpoint across 5 randomly generated seeds.

the human standby operator. Each method is trained on three seeds per task for statistical significance. Fig. 5 shows the performance of all methods on the three tasks.

1) **Block Push**. The real world setup is similar to simulation. The experiments are conducted on a Franka robotics arm. The observation space includes 64 × 64 RGB images from a fixed table-top RGB camera (Intel RealSense D435i), a wrist-mounted RGB camera (Intel RealSense D435i) rigidly attached to the arm's wrist and proprioception. Actions are defined in end-effector space and include Cartesian velocity commands in x, y, and z. At the beginning of each episode a 5 cm × 5 cm cube is placed in a 18 cm × 18 cm reset region. The target line is 21 cm from the center of the reset region. Demonstration data is collected in the real world via space-mouse, consisting of 10 demonstrations totaling 1,451 transitions.
2) **Pick-and-Place**. The real world experiments are conducted on a Kinova Gen3 6-DoF robotics arm equipped with a Robotiq 2F-85 gripper. The observations consist of 64 × 64 RG images from a table-top camera, a wrist-mounted camera and proprioceptive (joint) measurements. Actions are defined in end-effector space and include Cartesian velocity commands in x, y, and z, together with a gripper command. At the beginning of each episode, a light blue cuboid (4 cm × 4 cm × 3 cm) is randomly placed within a (8 cm × 16 cm) region. The minimum distance between the reset region and the target placement line in 18 cm. Demonstration data is collected in the real world via keyboard-based teleoperation, consisting of 10 demonstration episodes with a total of 1,025 transitions.

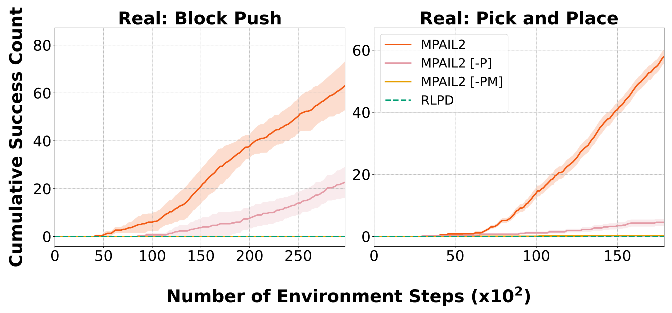


Fig. 5. Cumulative successes in real-world experiments. Offline results of BC are shown in Table IV. Complete training time for one training run (including resets, computation, etc.) is approximately 90 minutes for Block Push and 70 minutes for Pick and Place.

For more experimental details refer Section B.

C. Transfer Learning Experiment

Beyond single-task sample efficiency, methods in RL desirably enable continually transferrable learning across tasks. To evaluate transfer learning capabilities for each method, we train the method **first** on the (real-world) Block Push task until consistent successes are observed. Then, a **second** set of task observations are provided where the block is pushed in the opposite ($+y$) direction. For all methods, the weights of the models from the first task are used to initialize the weights for training the second task. As we are primarily interested in the transferability of the learned representations, we do not augment the setting to goal-conditioning or beyond. Aside from [$-$P] and BC, other online methods do not demonstrate successes in the initial task, which is a prerequisite to transfer learning. Thus, they are not evaluated in this experiment. As with previous experiments, BC is provided unlimited offline updates until convergence.

Models	Last Success (%)	Best Checkpoint #	 Success (%)
Real: Block Push			
MPAIL2	62	100	**100**
MPAIL2[-P] (MAIRL)	34	130	64
MPAIL2[-PM] (DAC)	0	—	0
RLPD	0	—	0
BC	**94**	—	94
Real: Pick-and-Place			
MPAIL2	**68**	140	**82**
MPAIL2[-P] (MAIRL)	16	150	16
MPAIL2[-PM] (DAC)	0	—	0
RLPD	0	—	0
BC	12	—	12
Real: Transfer Push (Transferred)			
MPAIL2	**62**	60	**90**
MPAIL2[-P] (MAIRL)	8	130	14
MPAIL2[-PM] (DAC)	0	NA	0
RLPD	0	NA	0
BC	8	—	8
Real: Transfer Push (From Scratch)			
MPAIL2	**80**	140	**90**
MPAIL2[-P] (MAIRL)	34	140	52
MPAIL2[-PM] (DAC)	0	NA	0
RLPD	0	NA	0
BC	26	—	26
Real: Transfer Pick and Place (Transferred)			
MPAIL2	**47**	93	**94**
MPAIL2[-P] (MAIRL)	0	123	16
MPAIL2[-PM] (DAC)	0	NA	0
RLPD	0	NA	0
BC	14	—	18
Real: Transfer Pick and Place (From Scratch)			
MPAIL2	**43**	143	**53**
MPAIL2[-P] (MAIRL)	6	135	12
MPAIL2[-PM] (DAC)	0	NA	0
RLPD	0	NA	0
BC	8	—	12

TABLE IV. Comparison of the performance of our method against baselines on real robot tasks. *Best* indicates the best performing model checkpoint given evaluation statistics over a training run; *Last* indicates the last model checkpoint in training. Evaluation metrics are computed over 50 trials per checkpoint over 3 randomly generated seeds.

D. Discussion of Results

Latent dynamics and model-based methods improve interaction efficiency. Allowing the model-free approach to be represented by [−PM], comparing MPAIL2, [−P], and [−PM] across all tasks reveal that latent dynamics representations and model-based training substantially improve interaction efficiency on average.

Planning improves learning stability and interaction efficiency. Across all tasks, [−P] suffers substantially from instability. Despite exhibiting interaction efficiency comparable at times to MPAIL2, [−P] begins to decrease its performance (or fail entirely) at some point during training. MPAIL2 exhibits this behavior on one seed in Sim: Block Push. Improvements to MPAIL2 relative to [−P] in sample efficiency are believed to arise as a consequence of improved learning stability via online planning.

Planning mitigates effects of plasticity loss, substantially improving transfer learning capability. Both BC and [−P] exhibit decreasing performance when transferring in Block Push (Fig. 6). This is likely attributed to plasticity loss of the models [28]. However, as MPAIL2 is subject to the same treatment, we surmise that planning plays an important role in mitigating its effects. More importantly, MPAIL2 begins to show success twice as fast on the new task when compared to training entirely from scratch, suggesting some degree of representation transfer to the new task.

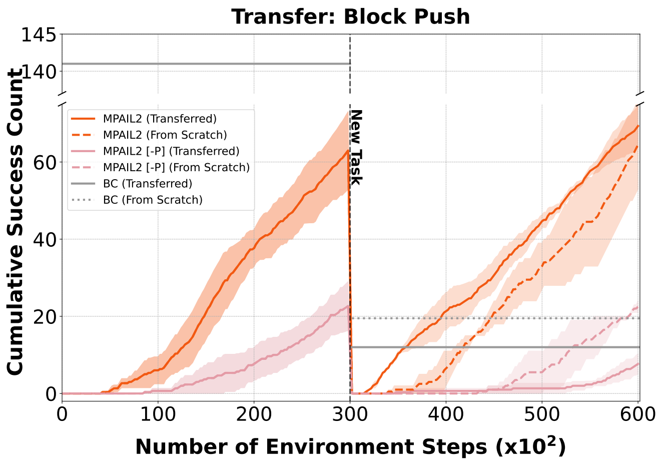


Fig. 6. Cumulative successes in real-world transfer experiment. Methods are first trained on the Block Push task, then transferred to a new task where demonstrations push the block in the opposite $(+y)$ direction. For experimental control, methods are trained on the second task from scratch. BC performance is shown as a fraction of the total interactions.

V. Limitations and Future Work

As existing work in real-world IRL from observation is scarce, efforts were made to minimize the number of confounding or superfluous design choices that would not obfuscate methodological conclusions. However, in doing so, we acknowledge a great deal of potential extensions, optimizations, regularization, architectures, etc. that may substantially improve performance for downstream applications. This section aims to consolidate future work mentioned throughout the manuscript as well as other promising goals and ideas.

Pre-training, prior data, world models, and cross-embodiment. Perhaps the most productive direction of future work may release constraints upon access to prior data. Indeed, our transfer experiments reveal that all components of MPAIL2, even those that are task-dependent, benefit from pre-training. Considering the generality of large pre-trained world [31], policy [5], and reward [52] models, the integration of prior data within the MPAIL2 framework yields high potential for general, pre-trained learning and planning from observation. Existing work in IRL has further demonstrated generalization to cross-embodiment learning provided greater amounts of data [51].

Stabilization methods. The adversarial manner in which the reward is trained can present as learning instability or inconsistency. In applications, it may be challenging or impossible to depend upon laboratory-setting metrics like reward or success rate which make early-stopping and deployment possible. The ability to continually train and improve monotonically with experience is necessary for real-world performance.

State-based observation and recurrent dynamics. Briefly mentioned in section III-B, latent states in real-world robotics may not necessarily be best represented as markovian phenomena. For example, hardware delay, PID tracking errors, simultaneous localization and mapping (SLAM) involve information dynamics which persist through time. Recurrent dynamics and state-based observers are potential approaches to resolve these discrepancies.

Uncertainty, exploration, and safety. World models like MPAIL2 help compartmentalize sources of uncertainty; encoder, dynamics, value, reward, policy, and planner are each responsible for different aspects of reasoning and acting. In this framework, it is more straightforward to rationalize what a learner might improve upon when the task is not the only objective. *What might an agent do beyond the task?*

Action and reward assumptions. While this work investigates more restrictive settings without access to demonstration action nor hand-designed rewards, MPAIL2 can be extended to leverage additional prior knowledge as in [37] or in [3].

VI. Conclusion

To fully utilize advancements in diverse mechanics and sensing, robots must not be limited by our ability to teleoperate them nor limited by sensing necessarily interpretable to us [19]. As humans do, they may learn by observing and predicting the impact of their actions in the world, constrained yet enabled by their own embodiments.

Previously, Inverse Reinforcement Learning from Observation has been impractically sample inefficient for real-world training. This work presents off-policy Model Predictive Adversarial Imitation Learning 2 (MPAIL2) towards real-world observational learning. By removing assumptions upon task demonstration actions and hand-designed reward, we investigate a problem setting in which world models and planning demonstrate remarkable learning efficiency and knowledge transfer directly in the real-world.

References

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Twenty-first international conference on Machine learning - ICML '04*, page 1, Banff, Alberta, Canada, 2004. ACM Press. doi: 10.1145/1015330.1015430. URL http://portal.acm.org/citation.cfm?doid=1015330.1015430.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 214–223. PMLR, July 2017. URL https://proceedings.mlr.press/v70/arjovsky17a.html. ISSN: 2640-3498.

[3] Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient Online Reinforcement Learning with Offline Data, May 2023. URL http://arxiv.org/abs/2302.02948. arXiv:2302.02948 [cs].

[4] Mohak Bhardwaj, Sanjiban Choudhury, and Byron Boots. Blending MPC & Value Function Approximation for Efficient Reinforcement Learning. October 2020. URL https://openreview.net/forum?id=RqCC_00Bg7V.

[5] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert 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. In *Proceedings of The 9th Conference on Robot Learning*, pages 17–40. PMLR, October 2025. URL https://proceedings.mlr.press/v305/black25a.html.

[6] Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized Ensembled Double Q-Learning: Learning Fast Without a Model. October 2020. URL https://openreview.net/forum?id=AY8zfZm0tDd.

[7] 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*, 44(10-11):1684–1704, September 2025. ISSN 0278-3649. doi: 10.1177/02783649241273668. URL https://doi.org/10.1177/02783649241273668. Publisher: SAGE Publications Ltd STM.

[8] Zichen Jeff Cui, Yibin Wang, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. From Play to Policy: Conditional Behavior Generation from Uncurated Robot Data, December 2022. URL http://arxiv.org/abs/2210.10047. arXiv:2210.10047 [cs].

[9] Neha Das, Sarah Bechtle, Todor Davchev, Dinesh Jayaraman, Akshara Rai, and Franziska Meier. Model-Based Inverse Reinforcement Learning from Visual Demonstrations. In *Proceedings of the 2020 Conference on Robot Learning*, pages 1930–1942. PMLR, October 2021. URL https://proceedings.mlr.press/v155/das21a.html.

[10] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 49–58. PMLR, June 2016. URL https://proceedings.mlr.press/v48/finn16.html.

[11] Justin Fu, Katie Luo, and Sergey Levine. Learning Robust Rewards with Adverserial Inverse Reinforcement Learning. February 2018. URL https://openreview.net/forum?id=rkHywl-A-.

[12] Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, and Stefano Ermon. IQ-Learn: Inverse soft-Q Learning for Imitation. In *Advances in Neural Information Processing Systems*, volume 34, pages 4028–4039. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/210f760a89db30aa72ca258a3483cc7f-Abstract.html.

[13] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/892c3b1c6dccd52936e27cbd0ff683d6-Abstract.html.

[14] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, August 2018. URL http://arxiv.org/abs/1801.01290. arXiv:1801.01290 [cs].

[15] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, April 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-08744-2. URL https://www.nature.com/articles/s41586-025-08744-2.

[16] Tyler Han, Yanda Bao, Bhaumik Mehta, Gabriel Guo, Anubhav Vishwakarma, Emily Kang, Sanghun Jung, Rosario Scalise, Jason Zhou, Bryan Xu, and Byron Boots. Model Predictive Adversarial Imitation Learning for Planning from Observation, July 2025. URL http://arxiv.org/abs/2507.21533. arXiv:2507.21533 [cs].

[17] Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, Robust World Models for Continuous Control. October 2023. URL https://openreview.net/forum?id=Oxh5CstDJU.

[18] Nicklas Hansen, Hao Su, and Xiaolong Wang. Learning Massively Multitask World Models for Continuous Control, December 2025. URL http://arxiv.org/abs/2511.19584. arXiv:2511.19584 [cs].

[19] Carolina Higuera, Akash Sharma, Taosha Fan, Chaithanya Krishna Bodduluri, Byron Boots, Michael Kaess, Mike Lambeta, Tingfan Wu,

Zixi Liu, Francois Robert Hogan, and Mustafa Mukadam. Tactile Beyond Pixels: Multisensory Touch Representations for Robot Manipulation, June 2025. URL http://arxiv.org/abs/2506.14754. arXiv:2506.14754 [cs].

[20] Jonathan Ho and Stefano Ermon. Generative Adversarial Imitation Learning. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/hash/cc7e2b878868cbae992d1fb743995d8f-Abstract.html.

[21] Arnav Kumar Jain, Vibhakar Mohta, Subin Kim, Atiksh Bhardwaj, Juntao Ren, Yunhai Feng, Sanjiban Choudhury, and Gokul Swamy. A Smooth Sea Never Made a Skilled SAILOR: Robust Imitation via Learning to Search. October 2025. URL https://openreview.net/forum?id=qN5hmLkBtC.

[22] Ryan Julian, Benjamin Swanson, Gaurav Sukhatme, Sergey Levine, Chelsea Finn, and Karol Hausman. Never Stop Learning: The Effectiveness of Fine-Tuning in Robotic Reinforcement Learning. In *Proceedings of the 2020 Conference on Robot Learning*, pages 2120–2136. PMLR, October 2021. URL https://proceedings.mlr.press/v155/julian21a.html.

[23] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-Actor-Critic: Addressing Sample Inefficiency and Reward Bias in Adversarial Imitation Learning. September 2018. URL https://openreview.net/forum?id=Hk4fpoA5Km.

[24] Patrick Lancaster, Nicklas Hansen, Aravind Rajeswaran, and Vikash Kumar. MoDem-V2: Visuo-Motor World Models for Real-World Robot Manipulation, May 2024. URL http://arxiv.org/abs/2309.14236. arXiv:2309.14236 [cs].

[25] Yankai Li and Mo Chen. Unifying Model Predictive Path Integral Control, Reinforcement Learning, and Diffusion Models for Optimal Control and Planning, February 2025. URL http://arxiv.org/abs/2502.20476. arXiv:2502.20476 [cs] version: 1.

[26] YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from Observation: Learning to Imitate Behaviors from Raw Video via Context Translation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1118–1125, May 2018. doi: 10.1109/ICRA.2018.8462901. URL https://ieeexplore.ieee.org/document/8462901/.

[27] Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. SERL: A Software Suite for Sample-Efficient Robotic Reinforcement Learning, March 2025. URL http://arxiv.org/abs/2401.16013. arXiv:2401.16013 [cs].

[28] Clare Lyle, Mark Rowland, and Will Dabney. Understanding and Preventing Capacity Loss in Reinforcement Learning, May 2022. URL http://arxiv.org/abs/2204.09560. arXiv:2204.09560 [cs].

[29] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to Adapt in Dynamic, Real-World Environments Through Meta-Reinforcement Learning, February 2019. URL http://arxiv.org/abs/1803.11347. arXiv:1803.11347 [cs].

[30] Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep Online Learning via Meta-Learning: Continual Adaptation for Model-Based RL, January 2019. URL http://arxiv.org/abs/1812.07671. arXiv:1812.07671 [cs].

[31] NVIDIA, Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, Daniel Dworakowski, Jiaojiao Fan, Michele Fenzi, Francesco Ferroni, Sanja Fidler, Dieter Fox, Songwei Ge, Yunhao Ge, Jinwei Gu, Siddharth Gururani, Ethan He, Jiahui Huang, Jacob Huffman, Pooya Jannaty, Jingyi Jin, Seung Wook Kim, Gergely Klár, Grace Lam, Shiyi Lan, Laura Leal-Taixe, Anqi Li, Zhaoshuo Li, Chen-Hsuan Lin, Tsung-Yi Lin, Huan Ling, Ming-Yu Liu, Xian Liu, Alice Luo, Qianli Ma, Hanzi Mao, Kaichun Mo, Arsalan Mousavian, Seungjun Nah, Sriharsha Niverty, David Page, Despoina Paschalidou, Zeeshan Patel, Lindsey Pavao, Morteza Ramezanali, Fitsum Reda, Xiaowei Ren, Vasanth Rao Naik Sabavat, Ed Schmerling, Stella Shi, Bartosz Stefaniak, Shitao Tang, Lyne Tchapmi, Przemek Tredak, Wei-Cheng Tseng, Jibin Varghese, Hao Wang, Haoxiang Wang, Heng Wang, Ting-Chun Wang, Fangyin Wei, Xinyue Wei, Jay Zhangjie Wu, Jiashu Xu, Wei Yang, Lin Yen-Chen, Xiaohui Zeng, Yu Zeng, Jing Zhang, Qinsheng Zhang, Yuxuan Zhang, Qingqing Zhao, and Artur Zolkowski. Cosmos World Foundation Model Platform for Physical AI, July 2025. URL http://arxiv.org/abs/2501.03575. arXiv:2501.03575 [cs].

[32] NVIDIA, Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, Lukasz Wawrzyniak, Milad Rakhsha, Alain Denzler, Eric Heiden, Ales Borovicka, Ossama Ahmed, Iretiayo Akinola, Abrar Anwar, Mark T. Carlson, Ji Yuan Feng, Animesh Garg, Renato Gasoto, Lionel Gulich, Yijie Guo, M. Gussert, Alex Hansen, Mihir Kulkarni, Chenran Li, Wei Liu, Viktor Makoviychuk, Grzegorz Malczyk, Hammad Mazhar, Masoud Moghani, Adithyavairavan Murali, Michael Noseworthy, Alexander Poddubny, Nathan Ratliff, Welf Rehberg, Clemens Schwarke, Ritvik Singh, James Latham Smith, Bingjie Tang, Ruchik Thaker, Matthew Trepte, Karl Van Wyk, Fangzhou Yu, Alex Millane, Vikram Ramasamy, Remo Steiner, Sangeeta Subramanian, Clemens Volk, C. Y. Chen, Neel Jawale, Ashwin Varghese Kuruttukulam, Michael A. Lin, Ajay Mandlekar, Karsten Patzwaldt, John Welsh, Huihua Zhao, Fatima Anes, Jean-Francois Lafleche, Nicolas Moënne-Loccoz, Soowan Park, Rob Stepinski, Dirk Van Gelder, Chris Amevor, Jan Car-

ius, Jumyung Chang, Anka He Chen, Pablo de Heras Ciechomski, Gilles Daviet, Mohammad Mohajerani, Julia von Muralt, Viktor Reutskyy, Michael Sauter, Simon Schirm, Eric L. Shi, Pierre Terdiman, Kenny Vilella, Tobias Widmer, Gordon Yeoman, Tiffany Chen, Sergey Grizan, Cathy Li, Lotus Li, Connor Smith, Rafael Wiltz, Kostas Alexis, Yan Chang, David Chu, Linxi "Jim" Fan, Farbod Farshidian, Ankur Handa, Spencer Huang, Marco Hutter, Yashraj Narang, Soha Pouya, Shiwei Sheng, Yuke Zhu, Miles Macklin, Adam Moravanszky, Philipp Reist, Yunrong Guo, David Hoeller, and Gavriel State. Isaac Lab: A GPU-Accelerated Simulation Framework for Multi-Modal Robot Learning, November 2025. URL http://arxiv.org/abs/2511.04831. arXiv:2511.04831 [cs].

[33] Manu Orsini, Anton Raichuk, Leonard Hussenot, Damien Vincent, Robert Dadashi, Sertan Girgin, Matthieu Geist, Olivier Bachem, Olivier Pietquin, and Marcin Andrychowicz. What Matters for Adversarial Imitation Learning? In *Advances in Neural Information Processing Systems*, volume 34, pages 14656–14668. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/7b647a7d88f4d6319bf0d600d168dbeb-Abstract.html.

[34] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi Jim Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick Tree Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, and Zipeng Lin. Open X-Embodiment: Robotic Learning Datasets and RT-X Models : Open X-Embodiment Collaboration0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903, May 2024. doi: 10.1109/ICRA57147.2024.10611477. URL https://ieeexplore.ieee.org/document/10611477/.

[35] Chaoyi Pan, Giri Anantharaman, Nai-Chieh Huang, Claire Jin, Daniel Pfrommer, Chenyang Yuan, Frank Permenter, Guannan Qu, Nicholas Boffi, Guanya Shi, and Max Simchowitz. Much Ado About Noising: Dispelling the Myths of Generative Robotic Control, December 2025. URL http://arxiv.org/abs/2512.01809.

arXiv:2512.01809 [cs].

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library, December 2019. URL http://arxiv.org/abs/1912.01703. arXiv:1912.01703 [cs].

[37] Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. AMP: adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics*, 40(4):1–20, August 2021. ISSN 0730-0301, 1557-7368. doi: 10.1145/3450626.3459670. URL https://dl.acm.org/doi/10.1145/3450626.3459670.

[38] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Visual Adversarial Imitation Learning using Variational Models. In *Advances in Neural Information Processing Systems*, volume 34, pages 3016–3028. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/1796a48fa1968edd5c5d10d42c7b1813-Abstract.html.

[39] Ali Rahimi and Benjamin Recht. Random Features for Large-Scale Kernel Machines. In *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL https://papers.nips.cc/paper_files/paper/2007/hash/013a006f03dbc5392effeb8f18fda755-Abstract.html.

[40] Erick Rosete-Beas, Oier Mees, Gabriel Kalweit, Joschka Boedecker, and Wolfram Burgard. Latent Plans for Task-Agnostic Offline Reinforcement Learning. In *Proceedings of The 6th Conference on Robot Learning*, pages 1838–1849. PMLR, March 2023. URL https://proceedings.mlr.press/v205/rosete-beas23a.html.

[41] Nikita Rudin, Junzhe He, Joshua Aurand, and Marco Hutter. Parkour in the Wild: Learning a General and Extensible Agile Locomotion Policy Using Multi-expert Distillation and RL Fine-tuning, May 2025. URL http://arxiv.org/abs/2505.11164. arXiv:2505.11164 [cs].

[42] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017. URL http://arxiv.org/abs/1707.06347. arXiv:1707.06347 [cs].

[43] Jiankai Sun, Lantao Yu, Pinqian Dong, Bo Lu, and Bolei Zhou. Adversarial Inverse Reinforcement Learning With Self-Attention Dynamics Model. *IEEE Robotics and Automation Letters*, 6(2):1880–1886, April 2021. ISSN 2377-3766. doi: 10.1109/LRA.2021.3061397. URL https://ieeexplore.ieee.org/document/9361118/.

[44] Richard S Sutton and Andrew G Barto. Reinforcement Learning: An Introduction.

[45] Faraz Torabi, Sean Geiger, Garrett Warnell, and Peter Stone. Sample-efficient Adversarial Imitation Learning from Observation, June 2019. URL http://arxiv.org/abs/1906.07374. arXiv:1906.07374 [cs].

[46] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative Adversarial Imitation from Observation, June 2019. URL http://arxiv.org/abs/1807.06158. arXiv:1807.06158 [cs].

[47] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A Standard Interface for Reinforcement Learning Environments, November 2025. URL http://arxiv.org/abs/2407.17032. arXiv:2407.17032 [cs].

[48] Bingzheng Wang, Guoqiang Wu, Teng Pang, Yan Zhang, and Yilong Yin. DiffAIL: Diffusion Adversarial Imitation Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(14):15447–15455, March 2024. ISSN 2374-3468. doi: 10.1609/aaai.v38i14.29470. URL https://ojs.aaai.org/index.php/AAAI/article/view/29470.

[49] Chen Wang, Linxi Fan, Jiankai Sun, Ruohan Zhang, Li Fei-Fei, Danfei Xu, Yuke Zhu, and Anima Anandkumar. MimicPlay: Long-Horizon Imitation Learning by Watching Human Play, October 2023. URL http://arxiv.org/abs/2302.12422. arXiv:2302.12422 [cs].

[50] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Information-Theoretic Model Predictive Control: Theory and Applications to Autonomous Driving. *IEEE Transactions on Robotics*, 34(6):1603–1622, December 2018. ISSN 1552-3098, 1941-0468. doi: 10.1109/TRO.2018.2865891. URL https://ieeexplore.ieee.org/document/8558663/.

[51] Kevin Zakka, Andy Zeng, Pete Florence, Jonathan Tompson, Jeannette Bohg, and Debidatta Dwibedi. XIRL: Cross-embodiment Inverse Reinforcement Learning. In *Proceedings of the 5th Conference on Robot Learning*, pages 537–546. PMLR, January 2022. URL https://proceedings.mlr.press/v164/zakka22a.html.

[52] Jiahui Zhang, Yusen Luo, Abrar Anwar, Sumedh Anand Sontakke, Joseph J. Lim, Jesse Thomason, Erdem Biyik, and Jesse Zhang. ReWiND: Language-Guided Rewards Teach Robot Policies without New Demonstrations. In *Proceedings of The 9th Conference on Robot Learning*, pages 460–488. PMLR, October 2025. URL https://proceedings.mlr.press/v305/zhang25a.html.

APPENDIX

Website

Experiment videos can be found at `https://uwrobotlearning.github.io/mpail2/`

A. Additional Results

Additional simulated experiments beyond manipulation are also conducted using **gymnasium environments** [47] for validating method generalization. These results are shown in Figure 7. Like experiments in Section IV, RLPD is still provided ground truth reward in addition to demonstration actions. *Unlike the experiments in Section IV*, demonstrations are synthetically collected by: first, running PPO [42] to recover an expert policy, then using this expert policy to generate demonstrations. We remark that terminations were enabled for the gymnasium environments, because drastically different optimal policies are observed when terminations are disabled. For example, the optimal policy in Hopper can converge to a "somersaulting" behavior, rather than hopping.

We additionally conduct a **transfer learning experiment in the Pick and Place** setting. Here, (10) new task demonstrations simply invert the directionality of the initial pick-and-place procedure by inverting the initial and final locations of the cube. This is unlike the Transfer Push setting where the demonstrations start in the same regions but move to opposite end goals. As the new task remains more closely in-distribution to the initial task, policy-based methods are able to demonstrate transfer, unlike in Transfer Push. MPAIL2 demonstrates significantly faster transfer with consistent success only after a few minutes of interaction under the new demonstrations. These results can be viewed in Figure 11.

Ablations over demonstration quantities, horizon lengths, and gradient penalty coefficients are also performed. These results are shown in Figure 8.

Finally, data transformations upon simulation results in Figure 4 are shown in Figure 9 and Figure 10 for visual comparisons of efficiency.

B. Experiment Details

1) Real World Push Setup: The real-world Push experiments are conducted on a Franka robotic arm. The observation space includes 64×64 RGB images from a fixed table-top RGB camera (Intel RealSense D435i), a wrist-mounted RGB camera (Intel RealSense D435i) rigidly attached to the arm's wrist and proprioception including joint position (7), joint velocities (7), end-effector Cartesian position (3) and a gripper state (1). Actions are defined in the end-effector space as position velocity (3). All actions are bound to a clipped workspace ranging from [0.4, -0.25, 0.06] to [0.65, 0.25, 0.25].

For the task a 5 cm$\times$5 cm cube is placed in a 18 cm$\times$18 cm reset region. We collect a dataset of 10 demonstrations using a space-mouse totaling 1,451 transitions.

Metrics. Due to the high-quality instrumentation of the push setup, we are able to compute block positions via camera calibrations and AprilTags. One of the uses of these block positions is providing RLPD with dense reward.

Ckpt (# steps)	**MPAIL2**	**MPAIL2 [-P] (MAIRL)**	**MPAIL2 [-PM] (DAC)**	**MPAIL2 [-PMO] (AIRL)**	**RLPD**
Sim: Block Push (State)					
100	64.0±13.0	21.0±14.7	14.0±14.0	0.0	4.0± 5.9
200	76.0± 8.1	6.0± 6.0	43.0±21.0	0.0	35.4± 4.1
300	71.0±13.7	4.0± 4.0	36.0±22.1	8.0± 8.0	54.8±11.5
400	87.0± 6.4	0.0	24.0±19.4	0.0	58.2± 9.0
500	88.0± 6.0	32.0±19.4	25.0±10.7	8.0± 8.0	60.4±12.5
Sim: Block Push (Visual)					
100	78.0± 7.0	40.0±24.5	0.0	0.0	0.0
200	80.0± 7.6	49.0±21.8	15.0±15.0	0.0	0.0
300	82.0± 5.1	0.0	19.0±16.6	16.0±16.0	0.0
400	76.0± 7.8	9.0± 9.0	32.0±19.8	0.0	0.0
500	73.0±13.3	35.0±15.2	59.0±20.8	0.0	0.0
Sim: Pick and Place					
100	9.0± 9.0	13.0± 7.0	0.0	0.0	0.0
200	33.0±15.0	17.0±17.0	0.0	0.0	0.0
300	58.0±14.5	32.0±20.2	2.0± 2.0	0.0	0.0
400	58.0±13.7	29.0±18.5	27.0±14.6	0.0	0.0
500	50.0± 8.4	41.0±21.4	33.0±19.6	0.0	0.0

TABLE V. **Full Checkpoint History by Environment Steps.**

	Last	**Best Checkpoint**	
Models	Success (%)	#	Success (%)
Real: Transfer Pick and Place (Transferred)			
MPAIL2	**47**	93	**94**
MPAIL2[-P] (MAIRL)	0	123	16
MPAIL2[-PM] (DAC)	0	NA	0
RLPD	0	NA	0
BC	14	—	18
Real: Transfer Pick and Place (From Scratch)			
MPAIL2	**43**	143	**53**
MPAIL2[-P] (MAIRL)	6	135	12
MPAIL2[-PM] (DAC)	0	NA	0
RLPD	0	NA	0
BC	8	—	12

TABLE VI. **Comparison of methods evaluated on Transfer Pick and Place experiment.** *Best* indicates the best performing model checkpoint given evaluation statistics over a training run; *Last* indicates the last model checkpoint in training. Evaluation metrics are computed over 50 trials per checkpoint over 3 randomly generated seeds.

Let d_{EE}^{B} be the distance between the block and the center of the end-effector (gripper). Let $y_{\text{goal}} := -0.1$ be the target y-threshold and let y_B be the y-coordinate of the block. The dense reward is given by

$$r(s) = -d_{\text{EE}}^{\text{B}} - |y_{\text{B}} - y_{\text{goal}}|. \tag{21}$$

Success of an episode or evaluation is credited if any $y_B < y_{\text{goal}}$ at some point in the trajectory.

2) Real World Pick-and-Place Setup: All real-world Pick-and-Place experiments are conducted on a Kinova Gen3 6-DoF robotic arm equipped with a Robotiq 2F-85 gripper. The observation space includes 64×64 RGB images from a fixed table-top RGB camera (Intel RealSense D435i), a wrist-mounted RGB camera (Intel RealSense D410) rigidly attached to the arm's wrist including joint position (6), joint velocities (6), end-effector Cartesian position (3) and a gripper state (2) - gripper state and gripper trigger.

At the beginning of each episode, a cuboid object (4 cm × 4 cm × 3 cm, light blue) is randomly placed within an 8 cm ×

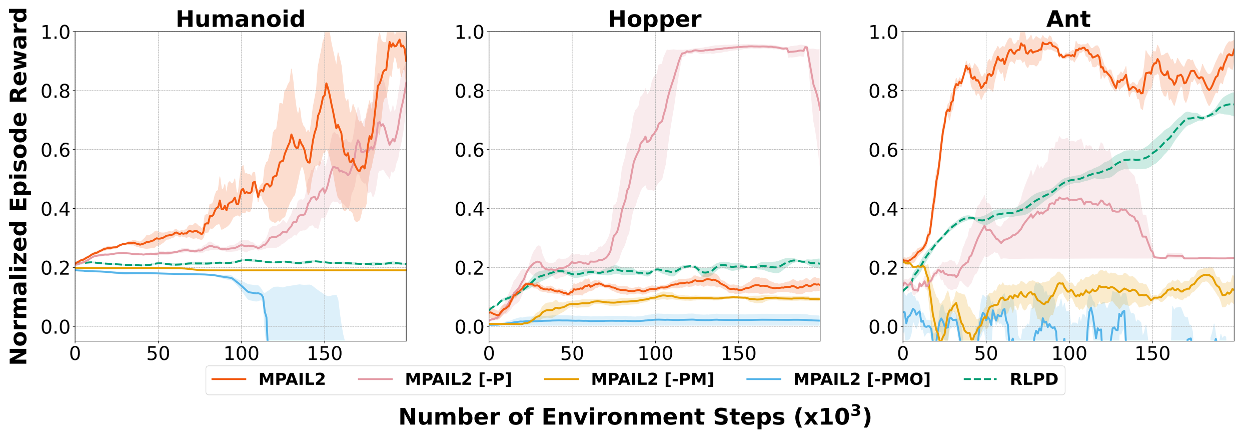


Fig. 7. **Results in gymnasium environments [47]**. All methods are provided 50 synthetic demonstration observations. RLPD is additionally provided dense reward and demonstration actions. Reward is normalized such that 1.0 reflects the average reward across the demonstrations.

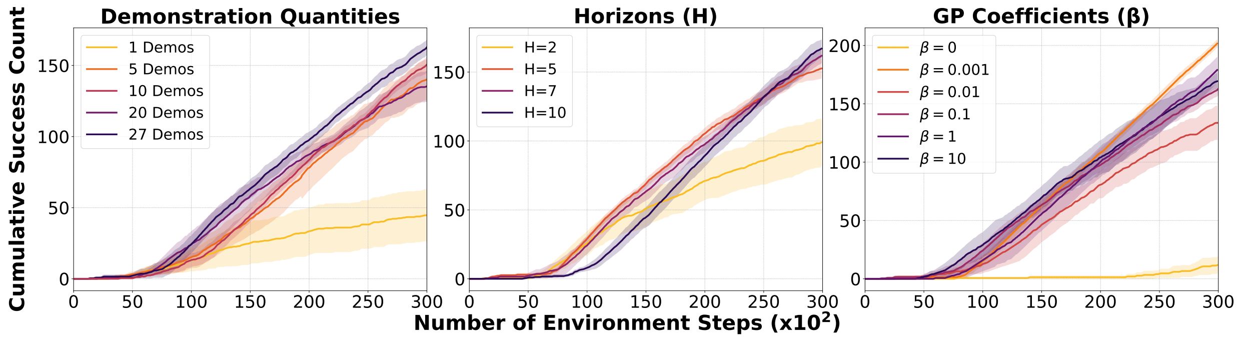


Fig. 8. **MPAIL2 ablations** over number of demonstrations (left), horizon lengths (middle), and gradient penalty coefficients (right) in Sim: Block Push.

16 cm reset region. The minimum distance between the reset region and the target placement line is 18 cm.

Actions are defined in end-effector space and include Cartesian position commands in x,y, and z, together with a gripper command. All actions are constrained to a clipped workspace with bounds $[0.30, -0.15, 0.166]$ and $[0.47, 0.15, 0.27]$.

Demonstration data are collected in the real world via keyboard-based teleoperation, consisting of 10 demonstration episodes with a total of 1,025 transitions.

Metrics. The real-world pick-and-place setup does not have the capacity to provide dense rewards to RLPD. However, as done in [27], the operator is on standby to label progressions for reward classification. Ultimately, RLPD never makes contact with the block and is not able to receive reward despite this assistance.

Success in the pick-and-place task is determined by completing four stages at any time in order: (i) initialized: $z_B < z_{\text{min}}$, (ii) lifted: $z_B \geq z_{\text{min}}$ and $y_B > y_{\text{goal}}$, (iii) transported: $z_B \geq z_{\text{min}}$ and $y_B \leq y_{\text{goal}}$, (iv) placed: $z_B \leq 0.03$ and $z_{\text{EE}} \leq 0.03$. These conditions are used in simulation to compute dense reward (using the Lagrangians of the conditions) for RLPD. In the real-world setup, stages are hand-labeled after training for evaluation metrics.

C. Implementation Details

Baselines

1) **[$-$P] (MAIRL [43])** [$-$P] simply removes the online planning component from MPAIL2. When acting in the environment, [$-$P] directly samples actions from a single-step policy $\pi_\phi(a_t|z_t)$. The policy is single-step as the multi-step policy is designed for reducing online planning computation. [$-$P] is equivalent to MPAIL2 in all other aspects: encoder, dynamics, reward, and value have equivalent losses. Where applicable, multi-step losses are computed through auto-regressive policy inference.
2) **[$-$PM] (DAC [23])** [$-$PM] removes both online planning and model-based components. Due to the large observation space and to reduce divergences from MPAIL2, [$-$PM] improves on DAC by encoding observations to a latent space. Its encoder is updated via the value target loss and a scaled learning rate. *We acknowledge that these modifications are not necessarily equivalent to MPAIL2, but we remark that various attempts were made at improving the model-free baseline and ensuring a fair ablation; for example, Random Fourier Features (RFF) [39] and no encoding did not exhibit noticeable improvements.*
3) **[$-$PMO] (AIRL [11, 33])** [$-$PMO] takes the implemen-

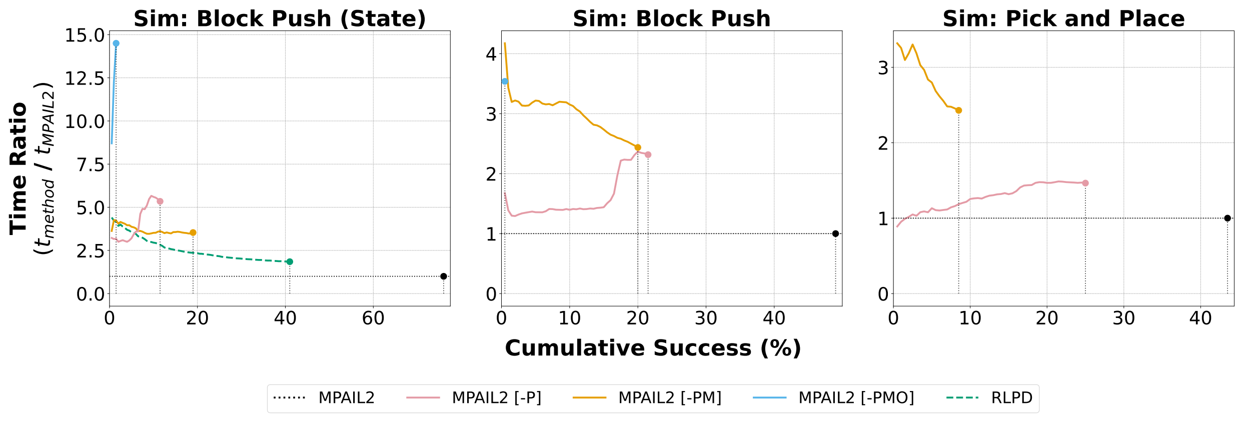


Fig. 9. **Time efficiency of MPAIL2 compared to other methods**. We plot $t_{\text{algo}}(c)/t_{\text{MPAIL2}}(c)$ where $t(c)$ is the learning iteration at which the algorithm achieves a cumulative success percentage. With respect to a given baseline, the value reflects how many times faster MPAIL2 achieves a given success percentage. An endpoint indicates where the corresponding baseline achieves its maximum success percentage.

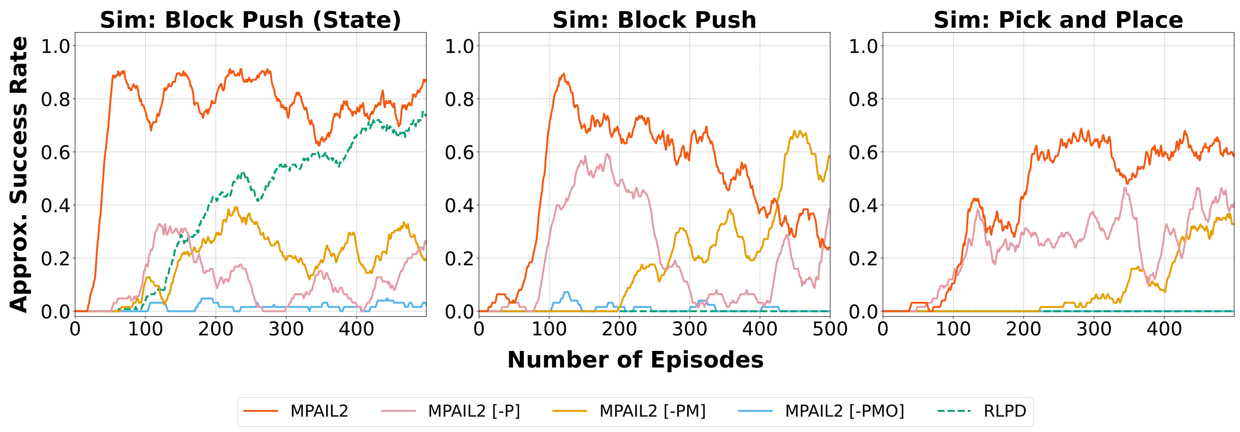


Fig. 10. **Success rate stability**. We plot the average success rate of each method over a moving window of 25 episodes. A value of 1 indicates success on all episodes across 25 updates. This graph captures a method's stability across updates. In *Sim: Block Push (State)* and *Sim: Pick and Place*, MPAIL2 is able to maintain a consistent success rate of roughly 0.8 and 0.6 respectively while the decreasing curve in *Sim: Block Push* indicates instability as witnessed by the flattening of the cumulative success curve in Figure 4.

tation of [−PM] and modifies the replay buffer such that only interactions within the previous episode are able to be sampled.

4) **RLPD** is implemented as given in [27] but is also improved via an encoder in [−PM].
5) **BC** is implemented as a diffusion policy with a U-Net denoiser. Implementation and parameters are as given in [7]. We train each policy for 500 epochs.

D. Additional Discussion of Results

The following discussions aim to provide additional evidence, details, observations, and summaries not explicitly provided in the main body.

Discussion by Task.

Block Push: In simulation, Block Push presents as a more dynamically unstable task. This characterization might explain BC's results in Table III, generally performing worse than [−P], [−PM], and MPAIL2. Often, BC appears to stay near expert trajectories when the block destabilizes completely beyond the gripper. In the real-world, these dynamics are less unstable, likely due to higher friction and slower actuation, resulting in higher real-world BC performance.

Regarding higher training instability observed in Block Push for MPAIL2 and [−P], a possible reason is that the expert never exhibits recovering from destabilization. However, as learners are always taking sampled, potentially suboptimal actions while learning, the learner almost always demonstrates destabilization and recovery. As a result, the inferred reward may become highly precise over training, making it increasingly difficult for the learner to meet the demands of the precise reward. The planner in MPAIL2 likely helps mitigate losing track of rewards entirely through long-horizon planning, resulting in significantly more stable learning than [−P]. Nevertheless, both simulated and real-world results empirically support pursuing future work which might help stabilize adversarial rewards or establish practical stopping-criteria.

Pick and Place: Compared to Block Push, Pick-and-Place is less dynamically sensitive. While it can be catastrophically unstable, we find that performance on Sim: Pick-and-Place can

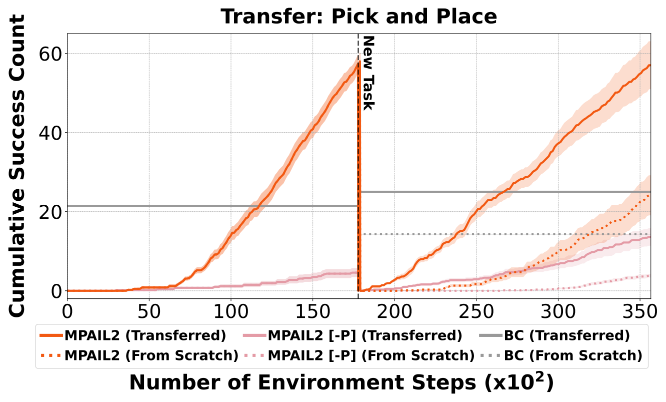


Fig. 11. **Cumulative successes in real-world transfer experiment in Pick and Place**. New task demonstrations invert the directionality of the initial demonstrations' pick and place procedure by inverting the start and end regions for the block. Model evaluations for this experiment are shown in Table VI.

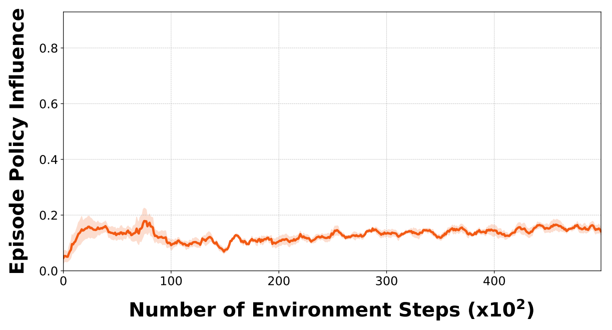


Fig. 12. **Policy influence on planning**. "Influence" of the policy on planning is calculated by taking the proportion of the total score contributed by the sampled policy plans during the final iteration of MPPI at one timestep (see Algorithm 2). Averaging over all timesteps in the episode gives the Episode Policy Influence on the y-axis of this plot.

be achieved by depending on the demonstrations. BC's high relative success suggests that the task does not require significant generalization beyond the demonstrations. However, Real: Pick-and-Place appears to require greater generalizability with policy methods ([$-$P] and BC) both dropping to below 16% in real-world from 40% in simulation. Observation noise, delays, and other real-world challenges are believed to contribute to the drop in performance. Planning appears to be robust against these challenges.

Gymnasium Environments: We evaluate MPAIL2 and all the baselines on 3 environments from gymnasium : *Ant-v5, Hopper-v5, Humanoid-v5* (Figure 7). We run each method for 200 training iterations with only 1 environment to mimic real world training settings where parallelization is not possible. Due to the high-action spaces of Humanoid (17) and Ant (8), these environments validate the role of the policy in seeding online plans. Since, prior work in MPAIL demonstrated poor performance on Ant without policy seeding.

We suspect the suboptimal performance of MPAIL2 in Hopper is primarily due to the environment's highly restrictive terminations adversely affecting planning. With terminations enabled, the learner is unable to collect experiences that would otherwise present counterfactual experience critical to planning. Future work assuming termination knowledge may benefit from providing this information to the learned dynamics model.

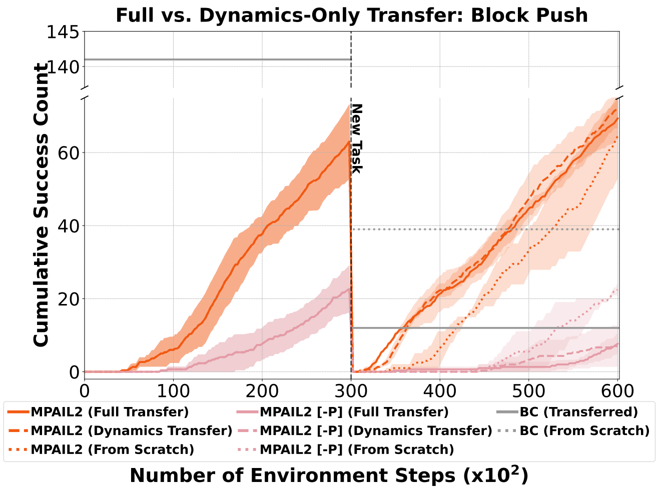


Fig. 13. **Transfer experiment** (Figure 6 with dynamics-only transfer. As in Section IV, all tasks related to transfer are performed in the real-world.

Component	**Hyperparameter**	**Value**
	Optimizer	Adam
	Learning Rate (LR)	3e-4
	Return Parameter (λ)	0.95
	Discount (γ)	0.99
	Horizon (H)	7
	Replay Size	∞
	UTD Ratio	1.0
	Batch Size	256
	Latent Dimension	512 (state: 256)
	Hidden Layers	[512, 512]
	LayerNorm	True
Encoder/Dynamics	Encoder LR	3e-5
	Temporal Discount (ρ)	0.95
Reward	GP Coefficient (β)	0.1
	LayerNorm	False
Value	Polyak Coefficient	0.01
	Gradient Norm Clip	5.0
	Ensemble Size	5
Policy	Target Entropy	$-\lvert\mathcal{A}\rvert$
	Alpha LR	3e-4
	Gradient Norm Clip	1.0
Planner	Temperature	2.0
	Number of Elites (K)	64
	Iterations	5
	Policy Plan Fraction	0.05
	Number of Rollouts (J)	512
	Std Range	(0.05, 2.0)

TABLE VII. **MPAIL Hyperparameters**. Used across all experiments unless specified otherwise. Only the Reward model does not employ LayerNorm.

Problem Learning Intuition.

As posited by prior work, MPAIL can be viewed as requiring the agent to learn a problem in addition to its solution, where the problem is given by the components: dynamics, reward, and value; and the problem's solution is given by the policy. This perspective provides intuition about why explicit policies can perform worse than planners (or iterative methods, in general [35]) despite optimizing equivalent objectives over similar data.

This intuition appears consistent in our visual manipulation experiments. Throughout Sim: Block Push, plans sampled from the policy make up less than 20% of the total score by the end of MPPI optimization (Figure 12). Though the policy's contributions to the score increase over training, it increases slowly relative to the rate of progress on the task. This supports our rationale from Section III-E indicating that the primary role of the explicit policy is to support off-policy value estimation and to seed online optimization. It also supports the intuition that simply enacting an explicit policy may be a primary contributor to low generalization (or large data requirements).

Our results corroborate general efforts in iterative (implicit policy) methods as a generalizable alternative to explicit policies [21, 7, 35]. Consequently, these implicit models obey an iteration objective. In Diffusion, in-distribution (energy-based) score is the iteration objective. In the case of MPAIL2, the model-based return is the iteration objective. We refer readers to [25] for an in-depth treatment of these equivalences

Why is MPAIL2 significantly more sample efficient?

First, we draw attention to the baseline, **RLPD**. Despite providing substantial assistance (dense reward and actions) to the state-of-the-art baseline, RLPD never exhibits any successes in the real-world during our experiments. Prior work reports only wall-clock time and does not include the number of interactions collected [27]. While wall-clock times for pick-and-place appear similar in magnitude (hours) as prior work, our experimental setups are not engineered to optimize for real-world training times. Thus, we hypothesize that real-world RLPD results in [27] collect more real-world interactions than our setups over the same period of wall-clock time. It is possible that RLPD may demonstrate successes provided more interactions. Additional software optimizations may also further improve MPAIL2 training times.

Now, we consider **BC** (Diffusion Policy). We observed that BC often exhibited behavior imitating exactly the demonstration data. This appears to work in its favor for Real: Block Push. While the task is dynamically sensitive, block destabilizations often remain in-distribution to the demonstration data. Thus, the BC policy is still able to draw upon demonstration coverage for task success. However, rote memorization works against the BC policy for Pick-and-Place as it results in a success rate of only $\%12$ compared to MPAIL2's $\%82$. In many failure cases, the BC policy does not grasp the cube but still continues to execute the remainder of the demonstration trajectory without the block grasped. In contrast, MPAIL2 occasionally misses the grasp but often returns to the block to reattempt a grasp. More generally, MPAIL2 often appears to exhibit recovery behaviors much earlier in training than other online baselines. While Diffusion Policy has been previously shown to exhibit recovery behaviors, it does not appear to do so in our experiments, likely due to the limited data regime.

Why does MPAIL2 exhibit transfer and improvement where standard approaches perform worse?

A long-standing challenge of continual reinforcement learning is the counterintuitive decrease in performance when policies, pre-trained on prior tasks, are fine-tuned or continually adapted for new tasks or environments [41]. Model-based representations have been believed to hold the key to unlocking continual improvement by rooting the learner's progression in the underlying dynamics of the world [44]. We believe the transfer experiment results in Figure 6 point to promising signs of these hypotheses operating in the real-world.

We reiterate that this work is the first to demonstrate online transfer learning between tasks (demonstrations) entirely in the real-world from scratch. Thus, prior work is relevant insofar as they encourage discussion as they differ in pre-training, tasks, and continual learning constraints. Prior methods have approached transfer learning through methods in data buffers, architectures, residual models, meta-learning, and more [22, 30, 29]. By contrast, MPAIL2 demonstrates transfer strictly through model-based representations and planning. Because MPAIL2 must still be subject to effects of plasticity loss, we hypothesize that transfer is enabled by the ability to *resolve new policies online*. As discussed in Section D, this capability of planning allows the learner to ignore its prior "solution" (policy) if it does not adhere to the "problem" (dynamics, reward, value) of the new task.

We conduct an additional transfer experiment where only the encoder and dynamics models are transferred to validate the transferability of the latent representation. Specifically, the weights of only e_ω and f_ψ are initialized using the pre-trained weights. The remaining components (reward r_θ, value Q_ζ, policy π_ϕ) are initialized randomly (trained from scratch). The results of this experiment can be found in Figure 13. Results of MPAIL2 indicate that dynamics-only transfer capabilities remain about the same as full transfer.

MPAIL2 deserves more attention to its transfer capabilities than can be provided in just this work's evaluations. However, by taking significant steps towards real-world transferrable IRLfO, we believe this work forms a strong foundation for further, more complex, experiments.

Areas of Improvement.

High variability in first-success times in both simulation and real-world experiments indicate that supervision of the latent space deserves further attention. Pre-trained visual encoders, initialization methods, or auxiliary losses may help in reducing variability.

Planning visualizations during development (as in Figure 2) provide insight into reasons for performance decreases or failures when they occur. As we have only visualized Sim: Block Push (state), the following conclusions may not be generally applicable. Nevertheless, these visualizations indicate that failures of MPAIL2 are often caused by collapsed plans at the beginning of the episode. Rarely does MPAIL2 fail while in the middle of the task. Rather, when they occur, failures often find the agent moving cyclically or back-and-forth.

```
[INFO] Planner initialized. Total number of params: 7114494
[INFO]  Encoder: 2327744
[INFO]  Dynamics: 793088
[INFO]  Reward: 787968
[INFO]  Value: 2649605
[INFO]  Sampling: 556088
Planner(
    (encoder): MultiCoder(
        (coders): ModuleList(
            (0): Coder(
                (coder): Sequential(
                    (0): Linear(in_features=8, out_features=256, bias=True)
                    (1): SiLU()
                    (2): Linear(in_features=256, out_features=512, bias=True)
                    (3): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
                    (4): SiLU()
                )
            )
            (1-2): 2 x CNNCoder(
                (coder): Sequential(
                    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2))
                    (1): SiLU()
                    (2): Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2))
                    (3): SiLU()
                    (4): Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2))
                    (5): SiLU()
                    (6): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1))
                    (7): SiLU()
                    (8): Flatten(start_dim=1, end_dim=-1)
                    (9): Linear(in_features=800, out_features=512, bias=True)
                    (10): SiLU()
                )
            )
        )
        (latent_coder): Sequential(
            (0): Linear(in_features=1536, out_features=512, bias=True)
            (1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
            (2): SiLU()
            (3): Linear(in_features=512, out_features=512, bias=True)
            (4): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
            (5): SiLU()
            (6): Linear(in_features=512, out_features=512, bias=True)
            (7): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
        )
    )
    (dynamics): Dynamics(
        (model): Sequential(
            (0): Linear(in_features=516, out_features=512, bias=True)
            (1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
            (2): SiLU()
            (3): Linear(in_features=512, out_features=512, bias=True)
            (4): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
            (5): SiLU()
            (6): Linear(in_features=512, out_features=512, bias=True)
            (7): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
        )
    )
    (reward): Reward(
        (model): Sequential(
            (0): Linear(in_features=1024, out_features=512, bias=True)
            (1): SiLU()
            (2): Linear(in_features=512, out_features=512, bias=True)
            (3): SiLU()
            (4): Linear(in_features=512, out_features=1, bias=False)
        )
    )
    (value): EnsembleValue(
        (Qs): ModuleList(
            (0-1): 2 x Q(
                (model): Sequential(
                    (0): Linear(in_features=516, out_features=512, bias=True)
                    (1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
                    (2): SiLU()
                    (3): Linear(in_features=512, out_features=512, bias=True)
                    (4): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
                    (5): SiLU()
                    (6): Linear(in_features=512, out_features=1, bias=True)
                )
            )
        )
    )
    (sampling): PolicySampling(
        (policy): PolicyNetwork(
            (model): Sequential(
                (0): Linear(in_features=512, out_features=512, bias=True)
                (1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
                (2): SiLU()
                (3): Linear(in_features=512, out_features=512, bias=True)
                (4): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
                (5): SiLU()
                (6): Linear(in_features=512, out_features=56, bias=True)
            )
        )
    )
)
```

Fig. 14. **PyTorch [36] 7.1M Model Architecture** for observation space with two RGB images and proprioception (i.e. Push Block and Pick-and-Place).

Algorithm 2 MPAIL2 (PLANNING)

```
Require:
    Number of trajectories to sample $N$;
    Proportion of trajectories sampled from policy $M/N$;
    Number of elite samples $K$
    Planning horizon $H$;
    Number of optimization iterations $J$
    Sampling standard deviation (std) clip range $(\sigma_{\min}, \sigma_{\max})$
    $e_\omega : \mathcal{O} \to \mathcal{Z}$                                          ▷ Encoder
    $f_\psi : \mathcal{Z} \times \mathbb{R}^m \times \mathcal{A} \to \mathcal{Z} \times \mathbb{R}^m$      ▷ Latent Dynamics
    $r_\theta : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$                                  ▷ Learned Reward
    $Q_\zeta : \mathcal{Z} \times \mathcal{A} \to \mathbb{R}$                                    ▷ Value
    $\mathbf{a} \sim \pi_\phi(\cdot|z)$                                        ▷ Multi-step Policy
1:  Procedure PLAN$(o_t, \mathbf{a}_{t-1})$
2:  $z_t \leftarrow e_\omega(o_t)$
3:  $(\boldsymbol{\mu}_t^0)_i \leftarrow (\mathbf{a}_{t-1})_{i+1}$     ▷ Roll previous plan one timestep backward for next sampling mean
4:  $(\boldsymbol{\mu}_t^0)_H \leftarrow 0$                             ▷ Set sampling mean to 0 for last timestep
5:  $\Sigma_{ii}^0 \leftarrow \sigma_{\max}$                          ▷ Set sampling std to max for first iteration
6:  $\Sigma_{ab,a\neq b} \leftarrow 0$                                            ▷ Isotropic sampling
7:  for $j \leftarrow 0$ to $J-1$ do
8:      for $k \leftarrow 0$ to $N-1$ do              ▷ Model rollouts and return estimation (parallelized)
9:          $\hat{s}_t^k \leftarrow s_t$
10:         if $k < N-M$ then
11:             $\mathbf{a}_t^k \sim \mathcal{N}(\boldsymbol{\mu}_t^j, \Sigma^j)$                          ▷ Sample random plan
12:         else
13:             $\mathbf{a}_t^k \sim \pi_\phi(\mathbf{a}|z_t)$                          ▷ Sample plan from policy
14:         end if
15:         for $t' \leftarrow t$ to $t+H-1$ do
16:             $\hat{z}_{t'+1}^k \leftarrow f_\psi(\hat{z}_{t'}^k, a_{t'}^k)$                     ▷ Predict next latent state
17:             $\hat{r}_{t'}^k \leftarrow r_\theta(z_{t'}^k, z_{t'+1}^k)$           ▷ Compute latent state-transition rewards
18:         end for
19:         $\mathcal{R}(\tau_k) \leftarrow \gamma^H Q_\zeta(\hat{z}_{t+H}^k, a_{t+H}^k) + \sum_{t'=0}^{H-1} \gamma^{t'} \hat{r}_{t'}^k$      ▷ Total trajectory return
20:     end for
21:     $\mathcal{R}_{\min} \leftarrow \mathcal{R}(\tau_k)_{(K)}$                 ▷ Elite cutoff given by $K$th largest return
22:     for $k \leftarrow 0$ to $N-1$ do
23:         if $\mathcal{R}(\tau_k) < \mathcal{R}_{\min}$ then
24:             $\mathcal{R}(\tau_k) \leftarrow 0$          ▷ Non-elite samples do not contribute to next distribution
25:         end if
26:     end for
27:     $\beta \leftarrow \max_k \mathcal{R}(\tau_k)$
28:     $\mathcal{Z} \leftarrow \sum_{k=1}^{N-1} \exp \frac{1}{\lambda} (\mathcal{R}(\tau_k) - \beta)$                             ▷ Total score
29:     for $k \leftarrow 0$ to $N-1$ do
30:         $w(\tau_k) \leftarrow \frac{1}{\mathcal{Z}} \exp \frac{1}{\lambda} (\mathcal{R}(\tau_k) - \beta)$       ▷ Weights are a trajectory's score over the total
31:     end for
32:     $\boldsymbol{\mu}_t^j \leftarrow \sum_{k=0}^{N-1} w(\tau_k)\, \mathbf{a}_t^k$
33:     $\boldsymbol{\sigma}_t^j \leftarrow \sqrt{\sum_{k=0}^{N-1} w(\tau_k) \left(\mathbf{a}_t^k - \boldsymbol{\mu}_t^j\right)^2}$
34:     $\Sigma_{ii}^j \leftarrow \text{clip}(\boldsymbol{\sigma}^j, \sigma_{\min}, \sigma_{\max})$
35: end for
36: $\mathbf{a}_t \sim \mathcal{N}(\boldsymbol{\mu}_t^J, \Sigma^J)$                   ▷ Sample new plan from optimized distribution
37: return $\mathbf{a}_t$
38: End Procedure
```