Online World Modeling Enables Real-World Inverse Reinforcement Learning from Observation

Tyler Han, Bat Nemekhbold, Siyang Shen, Rohan Baijal, Richard Ebock, Harine Ravichandiran, Sanghun Jung, Kevin Huang, and Byron Boots
{than123, bxtbold, andyshen, rbaijal, ebockr, harine, shjung13, kehuang, bboots}@uw.edu

University of Washington

Abstract

Current methods in robot learning are fundamentally bottlenecked by one or more of: hand-designed rewards, simulation modeling, or action supervision (e.g. teleoperation) each requiring significant domain expertise, engineering effort, and robot-operator labor. Towards eliminating these bottlenecks, this work pursues observational learning via Inverse Reinforcement Learning from Observation (IRLfO) in which only access to task observations (e.g. video) is assumed. Due to the challenging setting and limitations of RL methods, IRLfO has thus far remained impractical for real-world robot learning. Here, we present the first IRL method to learn visual manipulation in the real world from scratch, and the first real-world demonstration of positive online transfer across visual manipulation tasks from scratch. In under 40 minutes, MPAIL2 learns pick-and-place from scratch to 82% success, where RL and BC with equal interaction and demonstration budgets reach only 0% and 12% despite their reward and action supervision. Videos and more on project page: https://uwrobotlearning.github.io/mpail2/

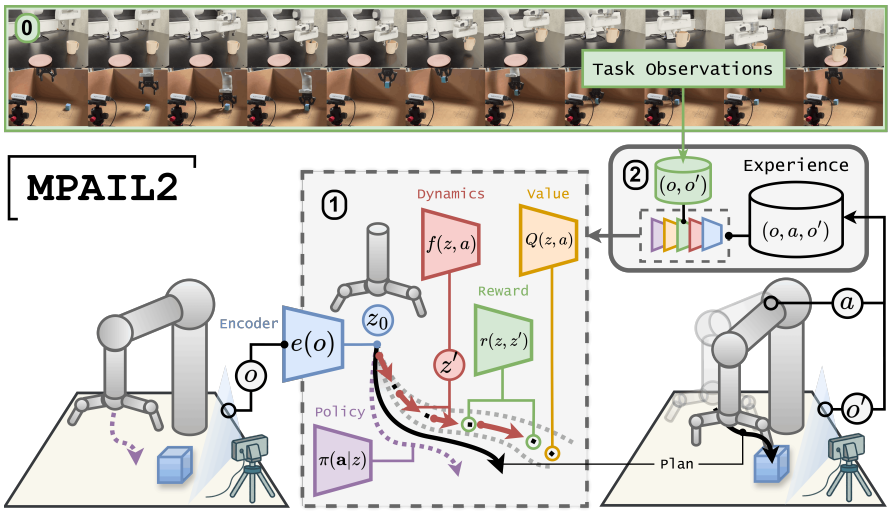


Figure 1: Overview of MPAIL2. **(0)** *Only* task observations are provided to the learner. **(1)** The learner makes and **encodes** observations into a its latent state representation. The **policy** predicts reactive actions which seed online planning. Performing online planning, the learner samples actions, predicts their **future states**, and evaluates plans using the adversarially learned **reward** and the **value**. **(2)** Plans with higher returns are more likely to contribute to the final decision. **(2)** After executing, the interaction (o, a, o') where o' is the posterior observation is accumulated in the learner's experience. All components are updated online.

1 Introduction

You are learning to receive a baseball pitch for the first time. Before you, your coach demonstrates the task by receiving then striking the ball. You step to the plate and mimic what you observed, but as the pitch arrives, you begin to experience the challenges your coach had long mastered: the sudden inertia of the bat, the shifting balance in your feet, and the ball's inbound trajectory. Even when provided a demonstration, we are often not immediately capable of completing a task without firsthand experience. However, our ability to infer the task from a few observations, evaluate our own performance, and self-improve with experience sets us apart from robots today.

One might wish that robots could similarly learn in this self-supervised manner, without bottlenecks such as reward shaping, direct action-level supervision, or hand-engineered simulations and models. This setting can be referred to as real-world Inverse Reinforcement Learning (IRL), where the reward must be learned alongside the optimal policy (Abbeel & Ng, 2004). IRL from Observation (IRLfO) further constrains demonstrations to be observation-only, without action labels, towards more scalable data sources, such as video or end-user (cross-embodiment) demonstration (Torabi et al., 2019b).

Thus far, IRL has largely been impractical for real-world robot learning primarily due to policy brittleness and poor sample efficiency (Han et al., 2025). Recently, Model Predictive Adversarial Imitation Learning (MPAIL) was proposed as a planning-based approach to IRLfO (Han et al., 2025), alleviating policy brittleness towards real-world on-robot deployment. But, like other IRL algorithms, MPAIL requires learning from thousands of policy rollouts (Sun et al., 2021), limiting its applicability to real-world settings.

This work presents MPAIL2 towards scalable, real-world IRLfO *and positive online task transfer*: MPAIL2 is the first method to learn visual manipulation tasks like pushing and pick-and-place from scratch in under 40 minutes, without privileged supervision like rewards, actions, or interventions. SOTA RL and Behavior Cloning baselines reach only 0% and 12% success, respectively, at matched interaction budgets, despite being given privileged rewards and action labels. This work is also the first to demonstrate positive online task transfer directly in the real world without offline re-training. Our key contributions are as follows:

1. An online, off-policy, latent world-modeling framework, which simultaneously improves encoder, dynamics, reward, value, and policy models using only observations and interaction. *No rewards, simulation, pre-training, or action supervision required.*
2. Simulated and real-world experiments finding that latent world modeling and planning are each critical towards enabling practical, real-world IRLfO.
3. Experiments supporting scalable robot learning directly in the real-world through positive online transfer learning and video-only demonstrations enabled by MPAIL2.

Due to MPAIL2's substantial reduction in training-from-scratch times, the following analyses also benefit from, to our knowledge, the first multi-seed evaluations in real-world IRL and the first multi-seed, multi-task evaluations in real-world RL. Experiment videos, code, and more can be found on the project website: https://uwrobotlearning.github.io/mpail2/.

The Big, Natural World.

The Big World Hypothesis (Javed & Sutton, 2024) holds that real agents inhabit environments too complex to be fully modeled, and must therefore act well under permanent resource and representational bounds rather than assuming a small enough world that can be completely known in advance. To be faithfully executed, this view makes several concrete demands on a learning method. State and dynamics must be constructed from the agent's own experience rather than presupposed. Externally engineered supervision, such as rewards, action labels, and simulation, becomes a potential liability rather than a convenience, since it encodes a smaller world than the one the agent actually inhabits. Competence must accumulate continually across tasks rather than reset with each one. And

all of this must happen under the real-time, real-compute constraints of an embodied agent, not the unbounded offline budget of a simulator.

MPAIL2 is, to our knowledge, the first method to satisfy all four conditions simultaneously in the real world. Its encoder and dynamics are learned entirely from self-supervised interaction with no pre-training (Q1). It matches or exceeds RL and BC baselines despite forgoing their reward and action supervision, suggesting that such supervision is not merely unnecessary but can actively constrain performance relative to a self-supervised alternative (Q2). Its representations transfer positively across tasks and observed demonstrations without offline re-training (Q3). All of the above is achieved from scratch, with hardware limitations, in under 40 minutes. We view these results less as an application of the Big World Hypothesis than as among the first direct empirical evidence for it outside of simulation.

With this work, we feel strongly that researchers today can currently engage with the Big World Hypothesis directly in the real-world by feasibly training "newborn" robots, which live in the same Big World as us. For newborns, the world can only be as big as it has thus far perceived it to be: its representations fragmenting and splintering under the pressure of complexities in its own experience, not due to complexities forced upon it via unnaturally privileged supervision. We also aim to provide LeRobot and low-cost robot MPAIL2 integrations to further lower barrier-to-entry.

2 Model Predictive Adversarial Imitation Learning 2

Problem. Consider a Partially Observable Markov Decision Process (POMDP) represented as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, \Omega, \gamma_H)$. The learner makes observations $o \in \mathcal{O}$ according to $\Omega(o \mid s)$ while the state $s \in \mathcal{S}$ is hidden. The state-transition reward function $R : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$, transition dynamics T, and terminal states are unknown. The learner receives a set of observation trajectories $\mathcal{D} = \{(o_1, \ldots, o_{T_i})_i\}_{i=1}^N$ generated by an optimal expert policy π_E. No action or reward supervision is provided. Following the apprenticeship-learning view of IRL (Abbeel & Ng, 2004), the learner seeks to infer candidate rewards $r \in \mathcal{R}$ and policies $\pi \in \Pi$ by assuming the expert performs optimally in expectation:

$$\arg\max_{r \in \mathcal{R}} \; \mathbb{E}_{\pi_E}[\, r \,] - \max_{\pi \in \Pi} \; \mathbb{E}_{\pi}[\, r \,] \tag{1}$$

While MPAIL demonstrates that online planning significantly improves learner robustness in IRL, it is computationally and fundamentally limited by its observation-based dynamics, and on-policy reward and value optimization (Ho & Ermon, 2016; Kostrikov et al., 2018), leading to reliance on prior modeling as well as massively simulated interactions (Han et al., 2025).

Approach. To enable real-world IRLfO, we propose MPAIL2, which instead performs latent world modeling and off-policy reward and value optimization to dramatically improve sample efficiency. To accomplish this, MPAIL2 introduces an encoder, latent dynamics model, multi-step policy, and corresponding off-policy objectives and architectures.

The *task-agnostic* latent representation is self-supervised through the encoder $z_t = e_\omega(o_t)$ and latent dynamics $\hat{z}_{t+1} = f_\psi(z_t, a_t)$ (Grill et al., 2020). The inferred reward $r_t = r_\theta(z_t, z_{t+1})$ and value $q_t = Q_\zeta(z_t, a_t)$ are used for offline, off-policy model-based policy optimization as well as for evaluating sampled plans during online planning. The multi-step policy $\hat{\mathbf{a}}_{t:t+H} \sim \pi_\phi(\cdot \mid z_t)$ produces H-step plans that warm-start planning and support off-policy value estimation. Learner interactions (o_t, a_t, o_{t+1}) are accumulated in a replay buffer $\mathcal{B}$. Figure 1 visualizes one interaction loop. As outlined in Algorithm 1 (Appendix A.1), each of the above components are optimized independently in each learner update. The remainder of Section 2 provides details on these components and their optimization objectives.

Notation. Hats $\hat{_}$ denote predicted quantities to distinguish imagination from interaction, bars $\bar{_}$ slow target networks, and **bold** symbols for temporal sequences of length H. Given o_t and plan $\mathbf{a}_t$, the *predicted trajectory* is defined as $\hat{\boldsymbol{\tau}}_t(\mathbf{a}_t) := \{\hat{z}_{t'}, a_{t'}, \hat{z}_{t'+1}\}_{t'=t}^{t+H}$ where $\hat{z}_{t'+1} = f_\psi(\hat{z}_{t'}, a_{t'})$,

$\hat{z}_t \coloneqq z_t = e_\omega(o_t)$. Let $\hat{\boldsymbol{\tau}}_t^\pi \coloneqq \hat{\boldsymbol{\tau}}_t(\hat{\mathbf{a}}_t)$ denote a policy-planned trajectory. Let $\tau_{t_1:t_2} = \{z_{t'}, a_{t'}, z_{t'+1}\}_{t'=t_1}^{t_2}$. The encoder allows direct sampling $\boldsymbol{\tau}_t \sim e_\omega(\mathcal{B})$ and $\{z, z'\} \sim e_\omega(\mathcal{D})$.

Encoder & Dynamics The goal of the latent dynamics are two-fold: (1) to enable online planning without requiring predictions directly in high-dimensional observation space and (2) to anchor task (reward) learning by restricting discrimination to lie within representations that are supported by the learner's direct interactions, i.e. its world model. Thus, the encoder and dynamics are trained jointly, task-agnostic, and self-supervised, by minimizing the discounted multi-step latent prediction loss

$$\mathcal{L}_{e,f}(\omega, \psi) = \mathbb{E}_{\boldsymbol{\tau}\sim\mathcal{B}}\left[\sum_{t'=t}^{t+H} \rho^{\,t'-t} \left\|\hat{z}_{t'} - \mathrm{sg}(z_{t'})\right\|_2^2\right], \tag{2}$$

where $\rho \in (0, 1)$ is a temporal discount and $\mathrm{sg}(\cdot)$ stops gradients on the target latent to prevent representational collapse (Chen & He, 2021; Tang et al., 2023). The loss depends only on observed transitions, independent of any reward or value head, yielding a task-independent representation that supports transfer (section Q3).

Inferred Reward Besides improved sample efficiency, training the reward off-policy maintains its coverage over all previous interactions, helping to stabilize the learned reward signal while generalizing it to more states (Kostrikov et al., 2018; Zhang et al., 2025b). Otherwise, on-policy reward learning converges meaninglessly to a constant (Fu et al., 2018). Greater reward coverage is also necessary for reliable sample-based planning, in which the reward must evaluate a number of sampled states online that may not necessarily be in-distribution. The reward's training objective realizes the top-level apprenticeship-learning problem in Equation (1) by latent world model surrogacy,

$$\arg\max_{r\in\mathcal{R}} \; \mathbb{E}_{(z,z')\sim e_\omega(\mathcal{D})}[r(z, z')] - \mathbb{E}_{(z,z')\sim e_\omega(\mathcal{B})}[r(z, z')] - \beta \cdot \mathrm{GP}(r, \mathcal{B}, \mathcal{D}), \tag{3}$$

where non-expert policies of Equation (1) are represented by samples from $\mathcal{B}$ and reward candidacy $r \in \mathcal{R}$ is regularized by the gradient penalty (Gulrajani et al., 2017) (defined in Appendix A.1.2).

Value The value supports recovery behavior learning, demonstration generalization, enables reasoning beyond the planning horizon, and provides a stable critic for the policy update. A Q-ensemble is used for variance reduction. The training objective is the entropy-regularizsed off-policy λ-return target (Haarnoja et al., 2018)

$$\mathcal{L}_Q(\zeta) = \mathbb{E}_{(z,a,z')\sim\boldsymbol{\tau},\; \hat{\mathbf{a}}'\sim\pi_\phi(\cdot|z')}\left[\left(Q_\zeta(z, a) - \bar{G}_t^\lambda(\hat{\boldsymbol{\tau}}_t^\pi)\right)^2\right], \tag{4}$$

with the λ-return $\bar{G}_t^\lambda$ computed from the slow target $\bar{Q}_\zeta$ via the recursion defined in Equation (6). Polyak updates (Haarnoja et al., 2018) and ensembling (Chen et al., 2020) extend standard latent-planning practice from the known-reward setting (Hansen et al., 2023) to our inferred-reward case.

Multi-step Policy We train π_ϕ to seed the planner with H-step warm-start plans. Single-step policies in current planning-based learners (Hansen et al., 2023) can otherwise induce computationally expensive rollouts during online planning, requiring several inferences from policy-then-dynamics in sequence. Also unlike multi-step policies supervised directly by expert action labels (Zhao et al., 2023; Chi et al., 2025), the IRLfO setting has no action labels available. Supervision arrives entirely through the inferred reward and value. The policy network is optimized against a TD(λ)-style return over H-step imagined rollouts with entropy regularization,

$$\max_{\pi\in\Pi} \; \mathbb{E}_{\hat{\mathbf{a}}_t\sim\pi(\cdot|z_t)}\left[G_t^\lambda(\hat{\boldsymbol{\tau}}_t^\pi) - \alpha \, \log \pi(\hat{\mathbf{a}}_t \mid z_t)\right], \tag{5}$$

where the λ-return G_t^λ is a TD(λ) return rolled out through the learned dynamics, mixing the bootstrap $\hat{q}$ with the model-based return $\hat{r}$ at rate $\lambda \in [0, 1]$,

$$G_t^\lambda(\hat{\boldsymbol{\tau}}_{t':t+H}^\pi) \coloneqq \lambda \hat{q}_{t'} + (1 - \lambda)\left[\hat{r}_{t'} + \gamma G_t^\lambda(\hat{\boldsymbol{\tau}}_{t'+1:t+H}^\pi)\right] \tag{6}$$

beginning with $t' \coloneqq t$ and terminating with $t' = t + H$ such that $G_t^\lambda(\hat{\boldsymbol{\tau}}_{t+H:t+H}^\pi) \coloneqq \hat{q}_{t+H}$. The temperature α entropy coefficient (Haarnoja et al., 2018), tuned to a target entropy per action step.

Task	Observations	Object dim. (reset region)	Demos
Simulation — IsaacLab, Franka arm; 5 seeds			
Block Push (state)	state	4×4×4 cm (10×6 cm)	27
Block Push	images + proprioception	4×4×4 cm (10×6 cm)	27
Pick-and-Place	images + proprioception	4×4×4 cm (10×6 cm)	30
Real — 64×64 RGB (external + wrist); Franka: Push, Kinova Gen3: PnP; 3 seeds			
Block Push	images + proprioception	5×5×5 cm (18×18 cm)	10 (1,451 trans.)
Pick-and-Place	images + proprioception	4×4×3 cm (8×16 cm)	10 (1,025 trans.)
Mug-on-Plate	images + proprioception	mug ∅8×9 cm (18×18 cm), plate ∅14 cm (20×30 cm)	15 (1,802 trans.)
Block Push (Video)	external image only	5×5×5 cm (18×18 cm)	10 (914 trans.)

Table 1: **Experiment Setup Summary.** *Block Push*: push cube past a target line. *Pick-and-Place*: grasp, lift 15 cm, place past a target line. Full details in Appendices B.2 and B.3.

Planner As previously found, planning-based actors exhibit significantly more robust and generalizable behavior compared to policy-based actors (Han et al., 2025; Hansen et al., 2023; Jain et al., 2025). Here, we choose Model Predictive Path Integral (MPPI) (Williams et al., 2018) due to its proven theoretical connections, simplicity, extensibility, popularity, and familiarity in other planning-based learners (Han et al., 2025; Hansen et al., 2023). A fraction of the sampled plans is drawn from $\pi_\phi(\cdot \mid z_t)$ to warm-start MPPI, and the remainder is sampled from the current proposal.

Algorithm and Architectures. Algorithmic and architectural details of MPAIL2's models, training, and planning procedures can be found in Appendix A.1 and Appendix A.2, respectively.

3 Experiments

Our experiments are motivated by the following research questions:

Q1 **Enabling Real-world IRLfO.** How does the design of MPAIL2 enable real-world IRLfO?

Q2 **IRLfO vs. RL/BC.** Does less supervision mean less performance?

Q3 **Scaling real-world learning.** How does MPAIL2 scale real-world robot learning beyond single-task imitation?

Experiment Setup. All experiments use human-operated demonstrations (Orsini et al., 2021). Reward and success definitions are in Appendix B. Results report two checkpoints: *Best* (best of saved checkpoints) and *Last* (final checkpoint), the latter quantifying performance when stopping criteria are unavailable. Random resets are managed programmatically. Real-world tasks are evaluated on two independent robot platforms (Franka: Block Push and Mug-on-Plate; Kinova Gen3: Pick-and-Place). Tasks details (e.g. object sizes and reset distributions) are described in Table 1. *In transfer learning*, methods are first trained on an initial task, e.g. Block Push, then on a new but related task, like pushing in the opposite ($+y$) direction, with model weights initialized from the first task. Only [−P] and BC are evaluated as they are the only other methods to achieve consistent success on the initial task. *In video-only demonstration tasks*, only the external, table-mounted camera is provided as observations. Action-supervised methods are still provided actions. Architectural changes made to MPAIL2 for video-only experiments are detailed in Appendix B.5.

Baselines. We compare with the following baselines, organized by their problem scope (Table 2).

IRL Methods. **AIRL** (Fu et al., 2018), **DAC** (Kostrikov et al., 2018), **MAIRL** (Sun et al., 2021), and **MPAIL** (Han et al., 2025) operate in the same IRLfO scope as MPAIL2 and are trivially extended to the observation-only setting by learning rewards over latent state-transitions (Torabi et al., 2019b). Towards answering **Q1** and fair comparison, we improve upon each of these baselines by applying MPAIL2's design choices where possible. These designs are not part of the original proposals but have been empirically validated in real-world robot learning (Ball et al., 2023; Luo et al., 2025). As a result, these IRL baselines can also be analyzed as ablations on MPAIL2's integration of planning [P], dynamics modeling [M], and off-policy optimization of reward and value [O]. Each baseline's components are summarized alongside their evaluation results in Table 2.

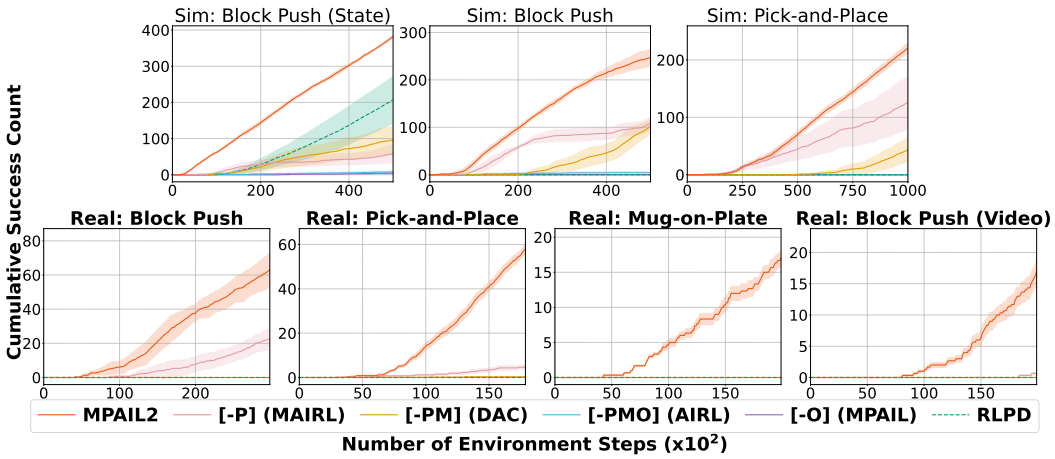


Figure 2: Cumulative successes over training across simulated and real world experiments. The onset and slope of each curve correspond to the learning time and approximate success rate of the method. Evaluations by checkpoint reported in Table 3. [−PMO] (AIRL) and [−O] (MPAIL) are not evaluated in real due to their negligible success rates across simulation tasks.

Reinforcement Learning with Prior Data (RLPD). **RLPD** is a state-of-the-art method in real-world RL-with-demonstrations. It operates at the scope where hand-designed reward and action supervision are available (Ball et al., 2023). We compare MPAIL2 (without reward or action access) to RLPD implemented via (Luo et al., 2025) with dense reward where known and action-labeled demonstrations.

Behavior Cloning (Diffusion). **Diffusion Policy** (Chi et al., 2025) demonstrates ideal, state-of-the-art performance when the environmental reset is "in-distribution" to the provided set of demonstrations.

Q1 World modeling *and* planning together enable real-world IRLfO.

We find that all of MPAIL2's design choices contribute positively towards alleviating the sample efficiency and robustness challenges that have made IRLfO and its approaches historically impractical. We make two significant observations:

World modeling is critical to efficiency in IRLfO. Of all evaluated IRL methods, those utilizing a latent dynamics model (MPAIL2 and [−P]) are the only approaches which demonstrate any successes when training in the real-world. Even in simulation, MPAIL2 and [−P] demonstrate successes significantly (2–3×) earlier in training than other methods. We further observe that off-policy training is itself critical. [−O], which preserves MPAIL2's world model, planner, and adversarial objective but trains the reward, value, and policy on-policy, fails to match MPAIL2's sample efficiency or stability under a matched interaction budget (Figure 2 and Table 3).

Planning critically mitigates IRL instability and improves actor robustness, enabling real-world IRLfO. Simulation results reveal and corroborate expected adversarial instability (Orsini et al., 2021), as reflected by the decaying or inconsistent slopes in Figure 2 corresponding to approximate drops in success rate. However, we observe that the instability is substantially mitigated via planning. While [−P]'s success rate tends to drop without resurgence, MPAIL2 exhibits little to no signs of instability with the exception of the late decay in Sim: Block Push. The impact of the planner's stability is clear in the real-world experimentation, where in many cases, MPAIL2 is the only IRL method to demonstrate consistent improvement over the course of training. In all cases, it is the *only* IRLfO method to demonstrate improvement over BC, indicating that it is actively expanding the robot's capability beyond the demonstration coverage strictly through online interaction *with neither rewards nor action supervision*.

	Supervision		Components						Transfer		Scratch		Video
Method	Reward	Actions	Plan.	Model	Off-Policy	BP	PnP	MoP	BP (rt.)	PnP (rt.)	BP (rt.)	PnP (rt.)	BP
MPAIL2			✔	✔	✔	**100±0**	**82±12**	**55 ± 15**	**90±5**	**94±10**	**88±6**	**53±37**	**63±10**
[−P] (MAIRL)				✔	✔	64±18	16±15	3.9 ± 2.8	14±13	16±18	53±12	12±12	29±5
[−PM] (DAC)					✔	0±0	0±0	0±0	0±0	0±0	0±0	0±0	0±0
BC (Diffusion)		✔				**94±6**	12±6	34±12	16±8	18±6	34±10	12±6	26±6
RLPD	✔*	✔			✔	0±0	0±0	0 ± 0	0±0	0±0	0±0	0±0	0±0

Table 2: **Real-world task evaluations summarizing over 96 total training runs from scratch.** (3 seeds/task; 50 trials/task): Success rate (%, mean ± std across seeds) at best checkpoint (full checkpoint evaluations in Appendix B.1). [−PMO] (AIRL) and [−O] (MPAIL) are not evaluated in real experiments due to impractically poor sample efficiency. *Transfer*: new push/place direction, initialized from last checkpoint of initial task. *Scratch*: new task trained from scratch. BP: Block Push, PnP: Pick-and-Place, MoP: Mug-on-Plate, rt.: right (opposite direction). *RLPD given known dense reward where available, otherwise classifier-based (Luo et al., 2025) reward.

Q2 Comparing IRLfO, RL, and BC: less supervision does not mean less performance.

MPAIL2 and its IRLfO ablations significantly outperform or compete with the performance of RLPD and BC, although they receive strictly less supervision. We reiterate that: a) RLPD is provided action supervision and dense reward (or hand-labeling for classifier-based reward (Luo et al., 2025)); and b) BC is provided action supervision; none of this information is provided to the IRL methods. Interestingly, the minimal off-policy IRL ablation, [−PM] (DAC), performs qualitatively better than RLPD by learning to reach within the allotted interactions. [−PM] (DAC) differs from RLPD merely through reward type and action supervision. As a result, we see a distinct benefit of IRL through **online, data-driven reward modeling** (Arjovsky et al., 2017). These results support the notion that a learned reward, even one without negatively-labeled examples, can be more informative than a hand-designed one. Despite RLPD's overall poor performance in almost all evaluations, its performance in Block Push (state) is second only to MPAIL2 and competitive in Gymnasium tasks (Figure 10), validating its implementation. RLPD appears to scale poorly with observation dimensionality and less direct supervision compared to even [−PM] (DAC).

MPAIL2 shows consistent improvement compared to BC by learning to generalize beyond the demonstrations through online interaction alone. Unlike supervised learning, which is unable to learn from failure or during deployment, new behavior can emerge autonomously under the (I)RL paradigm. For example, MPAIL2, like BC, tends to execute demonstrated motions even after missing a grasp in the first few minutes of training. After a few episodes (about 10 minutes), MPAIL2 learns to re-attempt grasps when missed, before continuing the remainder of the demonstration. Unlike RL or RLPD, these behaviors are not directly supervised via rewards or negative demonstrations. Poor behaviors are implicitly de-incentivized in IRL merely through the principle that they partially resemble, or do not resemble at all, the observed demonstrations; and recovery behaviors necessarily emerge self-supervised through learned reward and value generalization.

Q3 Scaling beyond single-task imitation: transfer learning and video-only demonstration.

Real-world robot learning is bottlenecked not only by per-task sample efficiency, but by how training, supervision effort, and learned structure scale across tasks. The ability to accumulate, understand, and transfer experience throughout robot deployment autonomously is a long-standing promise of model-based methods and RL (Sutton & Barto, 2018). Towards scalable real-world robot learning, we show that: MPAIL2 (1) exhibits consistent positive transfer to new but related tasks, and (2) can learn from video-only demonstration.

MPAIL2 learns significantly faster with weights transferred than trained-from-scratch, indicating positive representation transfer. When training Block Push (rt.) from scratch, MPAIL2 achieves the highest performing model (88% success rate) after 140 episodes (50 minutes). When using a transferred model, only 60 episodes (20 minutes) are needed for the highest performing model

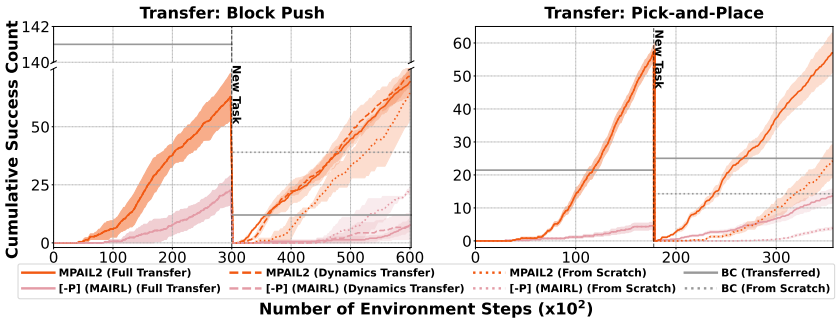


Figure 3: Cumulative successes in real-world transfer experiments compared to training from scratch. Methods are trained on the initial task (e.g. Block Push), then weights are transferred to training on a new task (e.g. Block Push right). *Dynamics Transfer* reflects transfer of only encoder and dynamics model weights. Numerical results are in Table 4.

(90%). For pick-and-place, this transfer reduction is about 20 minutes *while achieving roughly double the success rate of the best from-scratch model (53% vs. 94%)*. Object segmentation, identification, and rigid-body dynamics are the primary dynamics of our evaluation tasks and plausibly the primary features undergoing transfer, evidenced by how MPAIL2 spends much of the beginning learning to reach the object. Transfer is also supported by the following observation:

Planning mitigates the effects of plasticity loss, substantially improving transfer learning capability. BC and [−P] exhibit negative transfer, consistent with plasticity loss (Lyle et al., 2022). Due to the identical weight transfer procedure as [−P], MPAIL2's policy must likewise suffer from plasticity loss. However, the planner still exhibits positive transfer, suggesting that it may be enabled by disregarding policy output should it not align with the reward and value during interaction.

MPAIL2 can learn from scalable demonstration modalities (e.g. video). An advantage of IRLfO via MPAIL2 is that the reward can be conditioned on only sensing modalities accessibly shared by both demonstrator and learner (Zakka et al., 2022). For example, conditioning the reward model on only the external camera view may enable leveraging more accessible demonstration data, rather than restrictive combinations of wrist cameras, proprioception, and end-effector actions required by supervised learning (Collaboration, 2024). As a first step toward scalable third-person demonstration modalities, we evaluate learning from the fixed external camera only. With MPAIL2, the robot learns to perform the Block Push task from scratch to 63% success rate using only demonstration observations from the table camera view, which is more than double BC's success rate (26%) *even with action supervision*. Experiment details in Appendix B.5.

4 Conclusion

Due to its historically impractical learning efficiency and robustness, prior work in real-world IRL for robotics is scarce, especially so in the observation-only setting. In this work, we introduce MPAIL2 to enable real-world IRLfO, demonstrated through the learning of visual manipulation tasks from scratch in under 40 minutes from observation *without reward, action supervision, or simulation*. In enabling real-world IRLfO, we find that the performance of MPAIL2 and even its ablated IRL baselines do not suffer from less supervision and prior information compared to RL and BC. In fact, MPAIL2 strongly outperforms its RL and BC baselines by learning to generalize through self-guided online interaction. Beyond learning efficiency, enabling robots to learn and scale to tasks like baseball from observation will ultimately also depend upon supervision effort and transferability. To this end, we show that MPAIL2 also enables positive transfer learning and learning from more scalable demonstration modalities. Through demonstrating and validating these long-promised benefits of world modeling and IRL(fO), we hope that this work encourages researchers and practitioners alike to study the paradigm towards eliminating the numerous bottlenecks through which current robot learning methods may be limited.

References

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

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

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].

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*, pp. 17–40. PMLR, October 2025. URL https://proceedings.mlr.press/v305/black25a.html.

Xinlei Chen and Kaiming He. Exploring Simple Siamese Representation Learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15745–15753, Nashville, TN, USA, June 2021. IEEE. ISBN 978-1-6654-4509-2. DOI: 10.1109/CVPR46437.2021.01549. URL https://ieeexplore.ieee.org/document/9578004/.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized Ensembled Double Q-Learning: Learning Fast Without a Model. In *International Conference on Learning Representations*, October 2020. URL https://openreview.net/forum?id=AY8zfZm0tDd.

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.

Open X-Embodiment Collaboration. Open X-Embodiment: Robotic Learning Datasets and RT-X Models : Open X-Embodiment Collaboration0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903, May 2024. DOI: 10.1109/ICRA57147.2024.10611477. URL https://ieeexplore.ieee.org/document/10611477/.

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].

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*, pp. 1930–1942. PMLR, October 2021. URL https://proceedings.mlr.press/v155/das21a.html.

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*, pp. 49–58. PMLR, June 2016. URL https://proceedings.mlr.press/v48/finn16.html.

Justin Fu, Katie Luo, and Sergey Levine. Learning Robust Rewards with Adversarial Inverse Reinforcement Learning. In *International Conference on Learning Representations*, February 2018. URL https://openreview.net/forum?id=rkHywl-A-.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised Learning, September 2020. URL http://arxiv.org/abs/2006.07733. arXiv:2006.07733 [cs].

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.

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].

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.

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].

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, Robust World Models for Continuous Control. In *International Conference on Learning Representations*, October 2023. URL https://openreview.net/forum?id=Oxh5CstDJU.

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].

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.

Minyoung Hwang, Alexandra Forsey-Smerek, Nathaniel Dennler, and Andreea Bobu. Masked IRL: LLM-Guided Reward Disambiguation from Demonstrations and Language, March 2026. URL http://arxiv.org/abs/2511.14565. arXiv:2511.14565 [cs].

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. In *Advances in Neural Information Processing Systems*, October 2025. URL https://openreview.net/forum?id=qN5hmLkBtC.

Khurram Javed and Richard S. Sutton. The Big World Hypothesis and its Ramifications for Artificial Intelligence. July 2024. URL https://openreview.net/forum?id=Sv7DazuCn8#discussion.

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*, pp. 2120–2136. PMLR, October 2021. URL https://proceedings.mlr.press/v155/julian21a.html.

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. In *International Conference on Learning Representations*, September 2018. URL https://openreview.net/forum?id=Hk4fpoA5Km.

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.

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)*, pp. 1118–1125, May 2018. DOI: 10.1109/ICRA.2018.8462901. URL https://ieeexplore.ieee.org/document/8462901/.

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].

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].

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 2019a. URL http://arxiv.org/abs/1803.11347. arXiv:1803.11347 [cs].

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

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].

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, pp. 14656–14668. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/7b647a7d88f4d6319bf0d600d168dbeb-Abstract.html.

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].

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].

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.

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.

Nathan D. Ratliff, David Silver, and J. Andrew Bagnell. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, 27(1):25–53, July 2009. ISSN 1573-7527. DOI: 10.1007/s10514-009-9121-3. URL https://doi.org/10.1007/s10514-009-9121-3.

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*, pp. 1838–1849. PMLR, March 2023. URL https://proceedings.mlr.press/v205/rosete-beas23a.html.

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].

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/.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. MIT Press, 2018.

Yunhao Tang, Zhaohan Daniel Guo, Pierre Harvey Richemond, Bernardo Avila Pires, Yash Chandak, Remi Munos, Mark Rowland, Mohammad Gheshlaghi Azar, Charline Le Lan, Clare Lyle, András György, Shantanu Thakoor, Will Dabney, Bilal Piot, Daniele Calandriello, and Michal Valko. Understanding Self-Predictive Learning for Reinforcement Learning. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 33632–33656. PMLR, July 2023. URL https://proceedings.mlr.press/v202/tang23d.html.

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

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

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].

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].

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/.

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*, pp. 537–546. PMLR, January 2022. URL https://proceedings.mlr.press/v164/zakka22a.html.

Jesse Zhang, Marius Memmel, Kevin Kim, Dieter Fox, Jesse Thomason, Fabio Ramos, Erdem Bıyık, Abhishek Gupta, and Anqi Li. PEEK: Guiding and Minimal Image Representations for Zero-Shot Generalization of Robot Manipulation Policies, September 2025a. URL http://arxiv.org/abs/2509.18282. arXiv:2509.18282 [cs].

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*, pp. 460–488. PMLR, October 2025b. URL https://proceedings.mlr.press/v305/zhang25a.html.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware, April 2023. URL http://arxiv.org/abs/2304.13705. arXiv:2304.13705 [cs.RO].

Appendix Contents

A Algorithm & Architecture **15**

A.1 MPAIL2 Algorithms 15

A.2 MPAIL2 Architecture 16

B Experiment Setup **19**

B.1 Checkpoint Evaluation Results 19

B.2 Real-World Block Push Setup 19

B.3 Real-World Pick-and-Place Setup 20

B.4 Real-World Mug-on-Plate Setup 21

B.5 Video-only Demonstration Setup 21

C Implementation **22**

C.1 Hyperparameters 22

C.2 Baseline Implementations 22

D Additional Results **24**

D.1 Task Overview 24

D.2 Learning Curves and Ablations 24

D.3 World Model Analysis 24

E Additional Discussion **28**

E.1 Discussion by Task 28

E.2 Problem Learning Intuition 28

E.3 Why Is MPAIL2 Significantly More Sample Efficient? 29

E.4 Why Does MPAIL2 Exhibit Transfer? 29

F Related Work **31**

G Limitations and Future Work **32**

Website

Interactive project page: https://uwrobotlearning.github.io/mpail2/

A Algorithm & Architecture

The following section contains details describing the MPAIL2 algorithm, split into training and planning (acting). Architectural diagrams are included in Appendix A.2 to provide visual aid.

A.1 MPAIL2 Algorithms

A.1.1 Training

Algorithm 1 MPAIL2

Require: $\mathcal{D} \subset \mathcal{O} \times \mathcal{O}$ ▷ Task Observations
$e_\omega : \mathcal{O} \to \mathcal{Z}; \quad f_\psi : \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$ ▷ Encoder, Dynamics
$r_\theta : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}; \quad Q_\zeta : \mathcal{Z} \times \mathcal{A} \to \mathbb{R}$ ▷ Inferred Reward, 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

while learning **do**
  Interact using planner (Algorithm 2)

$$\mathcal{B} \leftarrow \mathcal{B} \cup \{(o_t, a_t, o_{t+1})\}_{t=1}^{T} \tag{7}$$

  for updates per episode **do**
    Sample trajectories and task observations

$$\{(o_t, a_t, o_{t+1})\}_{t=1}^{H} \sim \mathcal{B}\,,\ (o, o') \in \mathcal{D} \tag{8}$$

    Update Encoder and Dynamics (2)

$$\mathcal{L}_{e,f}(\omega, \psi) = \mathbb{E}_{\boldsymbol{\tau}}\left[\sum_{t'=t}^{t+H} \rho^{t'-t}\, \|\hat{z}_{t'} - \mathrm{sg}\,(z_{t'})\|_2^2 \,\middle|\, \boldsymbol{\tau}_t\right] \tag{9}$$

    Update Inferred Reward

$$\mathcal{L}_r(\theta) = \mathbb{E}_{(z,z')\sim\boldsymbol{\tau}}[r] - \mathbb{E}_d[r] + \beta\,\mathrm{GP}\,(r, \boldsymbol{\tau}, d) \tag{10}$$

    Update Value

$$\mathcal{L}_Q(\zeta) = \mathbb{E}_{(z,a,z')\sim\boldsymbol{\tau}, a'\sim\pi_\phi(\hat{\mathbf{a}}'|z')}\left[\left(q_t - \bar{G}_t^\lambda(\hat{\boldsymbol{\tau}}_t^\pi)\right)^2\right] \tag{11}$$

    Update Policy (5)

$$\mathcal{L}_\pi(\phi) = -\mathbb{E}_{\hat{\tau}}\left[G_t^\lambda(\hat{\boldsymbol{\tau}}_t^\pi)\right] \tag{12}$$

  end for
end while

A.1.2 Gradient Penalty

$$\mathrm{GP}(r, \mathcal{B}, \mathcal{D}) = \mathbb{E}_{(\tilde{z},\tilde{z}')}\left[\left(\left\|\nabla_{(\tilde{z},\tilde{z}')} r(\tilde{z}, \tilde{z}')\right\|_2 - 1\right)^2\right], \tag{13}$$

with $(\tilde{z}, \tilde{z}')$ uniform linear interpolations between expert latents $e_\omega(\mathcal{D})$ and buffer latents $e_\omega(\mathcal{B})$ (Gulrajani et al., 2017).

A.1.3 Planning

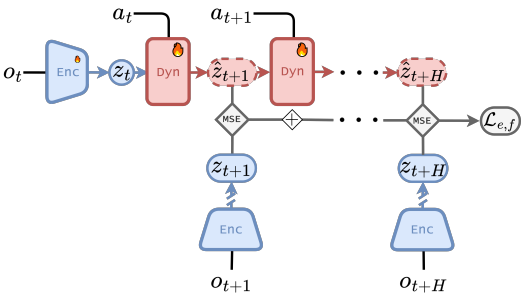
The planning algorithm is shown in Algorithm 2. It largely matches the implementation of MPPI (Williams et al., 2018; Hansen et al., 2023) with the learned dynamics, reward, and value models dropped in (highlighted in blue in Algorithm 2). Its objective can be written,

$$\arg\max_{\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t} \mathbb{E}_{\mathbf{a}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t)} \left[\gamma^H \hat{q}_{t+H} + \sum_{t'=t}^{H-1} \gamma^{t'-t} \hat{r}_{t'} \;\middle|\; \hat{\boldsymbol{\tau}}(\mathbf{a}_t) \right] \tag{14}$$

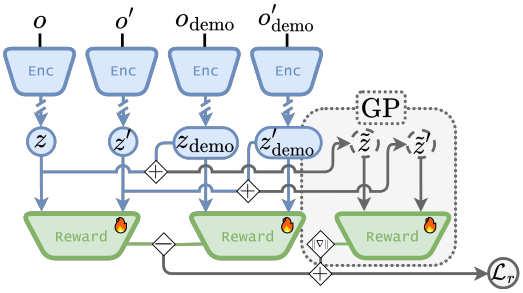
where the predicted rewards $\hat{r}$ and values $\hat{q}$ are used to estimate the return of each plan $\hat{\boldsymbol{\tau}}(\mathbf{a})$.

A.2 MPAIL2 Architecture

Figures 4a, 4b and 5 illustrate the encoder/dynamics training objective, the reward training objective, and the full MPAIL2 architecture respectively. The PyTorch module summary can be found in Figure 6.



(a) **Encoder and Dynamics Training.** The encoder e_ω maps observation o_t to latent state z_t. Dynamics f_ψ auto-regressively predicts future latent states $\hat{z}_{t+1}, \ldots, \hat{z}_{t+H}$ given action inputs. Training minimizes the multi-step MSE loss $\mathcal{L}_{e,f}$ against stop-gradient targets obtained by fresh encoder passes on subsequent observations (Eq. 9).



(b) **Reward Training.** The reward model r_θ is trained contrastively: agent transition scores (z, z') are minimized while demonstration transition scores $(z_{\text{demo}}, z'_{\text{demo}})$ are maximized. A gradient penalty (GP) regularizes the reward function over interpolated latent pairs $(\tilde{z}, \tilde{z}')$, yielding loss $\mathcal{L}_r$ (Eq. 10).

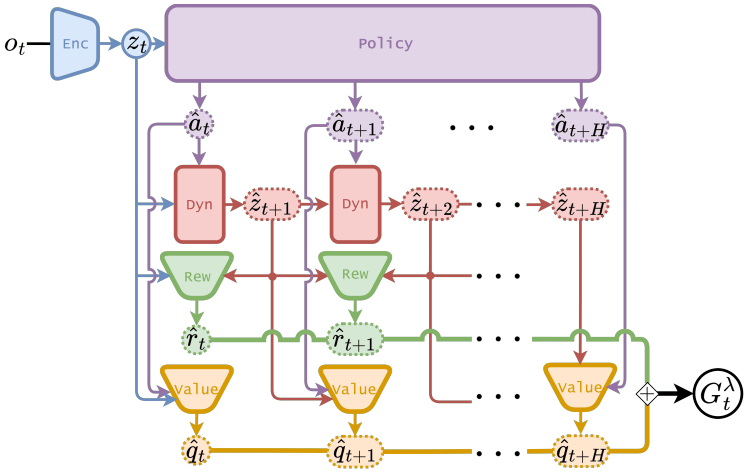


Figure 5: **MPAIL2 λ-Return.** Components are unrolled over horizon H. The encoder e_ω produces z_t; the policy produces action sequences $\hat{a}_t, \ldots, \hat{a}_{t+H}$; dynamics f_ψ predicts next latent states; reward r_θ scores transitions; value Q estimates returns. With the correct coefficients, these predictions compute the λ-return G_t^λ for actor-critic updates (Equation (6))

```
[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)
            )
        )
    )
)
```

Figure 6: **PyTorch (Paszke et al., 2019) 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 $(\sigma_{\text{min}}, \sigma_{\text{max}})$; Softmax temperature η
$e_\omega : \mathcal{O} \to \mathcal{Z}$ ▷ 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

Procedure PLAN$(o_t, \mathbf{a}_{t-1})$
$z_t \leftarrow e_\omega(o_t)$
$(\boldsymbol{\mu}_t^0)_i \leftarrow (\mathbf{a}_{t-1})_{i+1}$ ▷ Roll previous plan one timestep backward for next sampling mean
$(\boldsymbol{\mu}_t^0)_H \leftarrow 0$ ▷ Set sampling mean to 0 for last timestep
$\Sigma_{ii}^0 \leftarrow \sigma_{\text{max}}$ ▷ Set sampling std to max for first iteration
$\Sigma_{ab,a\neq b} \leftarrow 0$ ▷ Isotropic sampling
for $j \leftarrow 0$ **to** $J-1$ **do**
  for $k \leftarrow 0$ **to** $N-1$ **do** ▷ Model rollouts and return estimation (parallelized)
    $\hat{z}_t^k \leftarrow z_t$
    if $k < N - M$ **then**
      $\mathbf{a}_t^k \sim \mathcal{N}(\boldsymbol{\mu}_t^j, \Sigma^j)$ ▷ Sample random plan
    else
      $\mathbf{a}_t^k \sim \pi_\phi(\mathbf{a}|z_t)$ ▷ Sample plan from policy
    end if
    for $t' \leftarrow t$ **to** $t+H-1$ **do**
      $\hat{z}_{t'+1}^k \leftarrow f_\psi(\hat{z}_{t'}^k, a_{t'}^k)$ ▷ Predict next latent state
      $\hat{r}_{t'}^k \leftarrow r_\theta(\hat{z}_{t'}^k, \hat{z}_{t'+1}^k)$ ▷ Compute latent state-transition rewards
    end for
    $\mathcal{R}(\tau_k) \leftarrow \gamma^H Q_\zeta(\hat{z}_{t+H}^k, a_{t+H}^k) + \sum_{t'=0}^{H-1} \gamma^{t'} \hat{r}_{t'}^k$ ▷ Total trajectory return
  end for
  $\mathcal{R}_{\text{min}} \leftarrow \mathcal{R}(\tau_k)_{(K)}$ ▷ Elite cutoff given by Kth largest return
  for $k \leftarrow 0$ **to** $N-1$ **do**
    if $\mathcal{R}(\tau_k) < \mathcal{R}_{\text{min}}$ **then**
      $\mathcal{R}(\tau_k) \leftarrow 0$ ▷ Non-elite samples do not contribute to next distribution
    end if
  end for
  $\beta \leftarrow \max_k \mathcal{R}(\tau_k)$
  $\mathcal{Z} \leftarrow \sum_{k=0}^{N-1} \exp\left(\frac{1}{\eta}\left(\mathcal{R}(\tau_k) - \beta\right)\right)$ ▷ Total score
  for $k \leftarrow 0$ **to** $N-1$ **do**
    $w(\tau_k) \leftarrow \frac{1}{\mathcal{Z}} \exp\left(\frac{1}{\eta}\left(\mathcal{R}(\tau_k) - \beta\right)\right)$ ▷ Weights are a trajectory's score over the total
  end for
  $\boldsymbol{\mu}_t^j \leftarrow \sum_{k=0}^{N-1} w(\tau_k)\, \mathbf{a}_t^k$
  $\boldsymbol{\sigma}_t^j \leftarrow \sqrt{\sum_{k=0}^{N-1} w(\tau_k) \left(\mathbf{a}_t^k - \boldsymbol{\mu}_t^j\right)^2}$
  $\Sigma_{ii}^j \leftarrow \text{clip}(\boldsymbol{\sigma}^j, \sigma_{\text{min}}, \sigma_{\text{max}})$
end for
$\mathbf{a}_t \sim \mathcal{N}(\boldsymbol{\mu}_t^J, \Sigma^J)$ ▷ Sample new plan from optimized distribution
return $\mathbf{a}_t$
End Procedure

B Experiment Setup

We perform experiments in visual manipulation for two main reasons: (1) Planning-based IRL has been demonstrated to perform well in navigation (Han et al., 2025) but it is not yet clear whether the planning benefits go beyond self-prediction towards real-world dynamics, like object interactions. (2) Simulation, rewards, and action supervision are all practical bottlenecks in visual manipulation, making it a challenging yet realistic setting for IRLfO. Indeed, our baselines in RL and BC have been widely validated in visual manipulation (Ball et al., 2023; Chi et al., 2025). Best checkpoints are selected by evaluating all automatically saved checkpoints (every 10 episodes) beyond where the learner generally begins to exhibit successes (e.g. after 120 episodes in Mug-on-Plate).

B.1 Checkpoint Evaluation Results

Full evaluation results are reported in Table 3 (simulation) and in Table 4 (real-world).

Ckpt (# steps)	MPAIL2	MPAIL2 [-P] (MAIRL)	MPAIL2 [-PM] (DAC)	MPAIL2 [-PMO] (AIRL)	MPAIL2 [-O] (MPAIL)	RLPD	BC (Diffusion)
Sim: Block Push (State)							
100	64.0±13.0	21.0±14.7	14.0±14.0	0.0	0.0	4.0± 5.9	60.2± 1.6
200	76.0± 8.1	6.0± 6.0	43.0±21.0	0.0	0.0	35.4± 4.1	60.2± 1.6
300	71.0±13.7	4.0± 4.0	36.0±22.1	8.0± 8.0	4.0± 8.9	54.8±11.5	60.2± 1.6
400	87.0± 6.4	0.0	24.0±19.4	0.0	22.4±32.9	58.2± 9.0	60.2± 1.6
500	88.0± 6.0	32.0±19.4	25.0±10.7	8.0± 8.0	35.8±39.0	60.4±12.5	60.2± 1.6
Sim: Block Push (Image)							
100	78.0± 7.0	40.0±24.5	0.0	0.0	0.0	0.0	31.6± 2.0
200	80.0± 7.6	49.0±21.8	15.0±15.0	0.0	0.0	0.0	31.6± 2.0
300	82.0± 5.1	0.0	19.0±16.6	16.0±16.0	3.8± 8.5	0.0	31.6± 2.0
400	76.0± 7.8	9.0± 9.0	32.0±19.8	0.0	0.0	0.0	31.6± 2.0
500	73.0±13.3	35.0±15.2	59.0±20.8	0.0	0.0	0.0	31.6± 2.0
Sim: Pick and Place (Image)							
100	9.0± 9.0	13.0± 7.0	0.0	0.0	0.0	0.0	46.6± 2.3
200	33.0±15.0	17.0±17.0	0.0	0.0	0.0	0.0	46.6± 2.3
300	58.0±14.5	32.0±20.2	2.0± 2.0	0.0	0.0	0.0	46.6± 2.3
400	58.0±13.7	29.0±18.5	27.0±14.6	0.0	0.0	0.0	46.6± 2.3
500	50.0± 8.4	41.0±21.4	33.0±19.6	0.0	0.0	0.0	46.6± 2.3

Table 3: **Full Checkpoint History by Environment Steps in Simulation.** BC (Diffusion) is offline; its performance is constant across interaction budgets.

B.2 Real-World Block Push Setup

The real-world Block 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 $5\,\text{cm} \times 5\,\text{cm} \times 5$ cm cube is placed in a $18\,\text{cm} \times 18$ 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.

Let d^{B}_{EE} 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

$$r(s) = -d^{\text{B}}_{\text{EE}} - |y_{\text{B}} - y_{\text{goal}}|. \tag{15}$$

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

Ckpt	MPAIL2	[-P] (MAIRL)	[-PM] (DAC)	RLPD	BC (Diff.)
Real: Block Push					
Last	61.8 ± 13.5	34.0 ± 17.7	0.0 ± 0.0	0.0 ± 0.0	$\mathbf{94.1 \pm 5.9}$
Best (#)	$\mathbf{100 \pm 0}$ (100)	64.0 ± 17.7 (130)	0.0 ± 0.0 (—)	0.0 ± 0.0 (—)	$\mathbf{94.1 \pm 5.9}$ (—)
Real: Pick-and-Place					
Last	$\mathbf{68.0 \pm 13.6}$	16 ± 14.8	0.0 ± 0.0	0.0 ± 0.0	12.0 ± 5.8
Best (#)	$\mathbf{82.0 \pm 12.2}$ (140)	16.0 ± 14.8 (150)	0.0 ± 0.0 (—)	0.0 ± 0.0 (—)	12.0 ± 5.8 (—)
Real: Mug-on-Plate					
Last	$\mathbf{35.3 \pm 17.3}$	2.0 ± 2.8	0.0 ± 0.0	0.0 ± 0.0	$\mathbf{33.5 \pm 12.3}$
Best (#)	$\mathbf{54.9 \pm 14.7}$ (150)	3.9 ± 2.8 (190)	0.0 ± 0.0 (—)	0.0 ± 0.0 (—)	33.5 ± 12.3 (—)
Real: Transfer Push (Transferred)					
Last	$\mathbf{63.5 \pm 4.1}$	8.1 ± 5.7	0.0 ± 0.0	0.0 ± 0.0	7.8 ± 3.0
Best (#)	$\mathbf{90.2 \pm 4.7}$ (60)	13.9 ± 13.3 (130)	0.0 ± 0.0 (—)	0.0 ± 0.0 (—)	15.8 ± 7.8 (—)
Real: Transfer Push (From Scratch)					
Last	$\mathbf{80.3 \pm 11.5}$	34.0 ± 9.8	0.0 ± 0.0	0.0 ± 0.0	26 ± 9.8
Best (#)	$\mathbf{88.2 \pm 5.7}$ (140)	52.9 ± 11.8 (140)	0.0 ± 0.0 (—)	0.0 ± 0.0 (—)	26 ± 9.8 (—)
Real: Transfer Pick-and-Place (Transferred)					
Last	$\mathbf{47 \pm 29.4}$	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	14 ± 3.3
Best (#)	$\mathbf{94 \pm 10.2}$ (93)	16 ± 17.9 (123)	0.0 ± 0.0 (—)	0.0 ± 0.0 (—)	18 ± 5.8 (—)
Real: Transfer Pick-and-Place (From Scratch)					
Last	$\mathbf{43 \pm 29}$	6 ± 5.8	0.0 ± 0.0	0.0 ± 0.0	8 ± 3.3
Best (#)	$\mathbf{53 \pm 36.7}$ (143)	12 ± 11.7 (135)	0.0 ± 0.0 (—)	0.0 ± 0.0 (—)	12 ± 5.8 (—)
Real: Block Push (Video-Only Demonstration)					
Last	$\mathbf{58.8 \pm 14.4}$	25.5 ± 12.1	0.0 ± 0.0	0.0 ± 0.0	23.5 ± 8.3
Best (#)	$\mathbf{62.7 \pm 10.0}$ (173)	29.4 ± 8.3 (193)	0.0 ± 0.0 (—)	0.0 ± 0.0 (—)	25.5 ± 5.5 (—)

Table 4: **Real-World Results.** Success rate (%) $\pm$ std across seeds by task across methods. Evaluations performed over 50 trials/task over 3 seeds. *Last*: final-checkpoint success. *Best (#)*: best-checkpoint success with corresponding average environment-step count.

B.3 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 ($4\,\text{cm} \times 4\,\text{cm} \times 3$ cm, light blue) is randomly placed within an $8\,\text{cm} \times 16$ 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 (Luo et al., 2025), the operator is on standby to label progressions for reward classification. Within the limited interaction budget considered here, RLPD does not make contact with the block and so rarely receives reward, consistent with the difficulty of visual RL from scratch at this budget.

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.

B.4 Real-World Mug-on-Plate Setup

The real-world Mug-on-Plate experiments are conducted on a Franka robotic arm with the same observation space as the Block Push setup (Appendix B.2). Actions are defined in end-effector space as position velocity with a gripper command (4). All actions are bound to a clipped workspace ranging from $[0.4, -0.27, 0.085]$ to $[0.65, 0.27, 0.25]$.

At the beginning of each episode, a mug ($\varnothing 8\,\text{cm} \times 9\,\text{cm}$) is randomly placed within an $18\,\text{cm} \times 18\,\text{cm}$ reset region, and a plate ($\varnothing 14\,\text{cm}$) is randomly placed within a $20\,\text{cm} \times 30\,\text{cm}$ reset region. Demonstration data are collected via keyboard-based teleoperation, consisting of 15 episodes with a total of 1,802 transitions.

Metrics. The mug-on-plate setup does not support automatic object tracking, so no dense reward is available for RLPD; as in the pick-and-place setup (Appendix B.3), the operator labels reward during training via a keyboard interface. Success is defined as the mug resting stably on the plate at any point during the episode.

B.5 Video-only Demonstration Setup

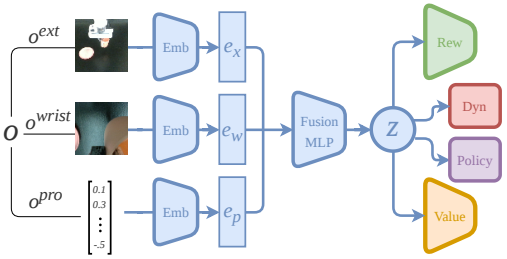


Figure 7: **Fused encoder** used in all experiments, except for video-only demonstration task. o^{ext} represents the *external* video modality.

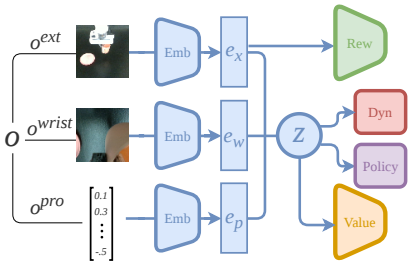


Figure 8: **Substate reward encoder** used in video-only demonstration task. o^{ext} represents the *external* video modality.

Figure 7 and Figure 8 show the architectural modifications made for Block Push task from video-only demonstration. Inclusion of fusion MLP is used in most experiments mainly due to the popularity of the design (Hansen et al., 2025; Jain et al., 2025). However, our results reveal that it is equally possible to have the dynamics "inherit" the fusion process instead, keeping the latent representation compartmentalized. This enables rewarding a subset of the latent state towards scalable demonstration modalities, like third-person video.

C Implementation

With settings evaluated in this work, MPAIL2's action rate in "Real:Push" is 10.28 $\pm$ 0.45 Hz. In simulation without time-synchronization, the action rate is 22.05 $\pm$ 1.55 Hz. Parameters can be found in Appendix C.1.

C.1 Hyperparameters

Component	**Hyperparameter**	**Value**
	Optimizer	Adam
	Learning Rate (LR)	3e-4
	Return Parameter (λ)	0.95
	Discount (γ)	0.99
	Planning 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	$-\|\mathcal{A}\|$
	Alpha LR	3e-4
	Gradient Norm Clip	1.0
Planner	Temperature (η)	2.0
	Number of Elites (K)	64
	Iterations (J)	5
	Policy Plan Fraction (M/N)	0.05
	Number of Rollouts (N)	512
	Std Range ($\sigma_{\min}, \sigma_{\max}$)	(0.05, 2.0)

Table 5: **MPAIL2 Hyperparameters**. Used across all experiments unless specified otherwise. Only the Reward model does not employ LayerNorm.

C.2 Baseline Implementations

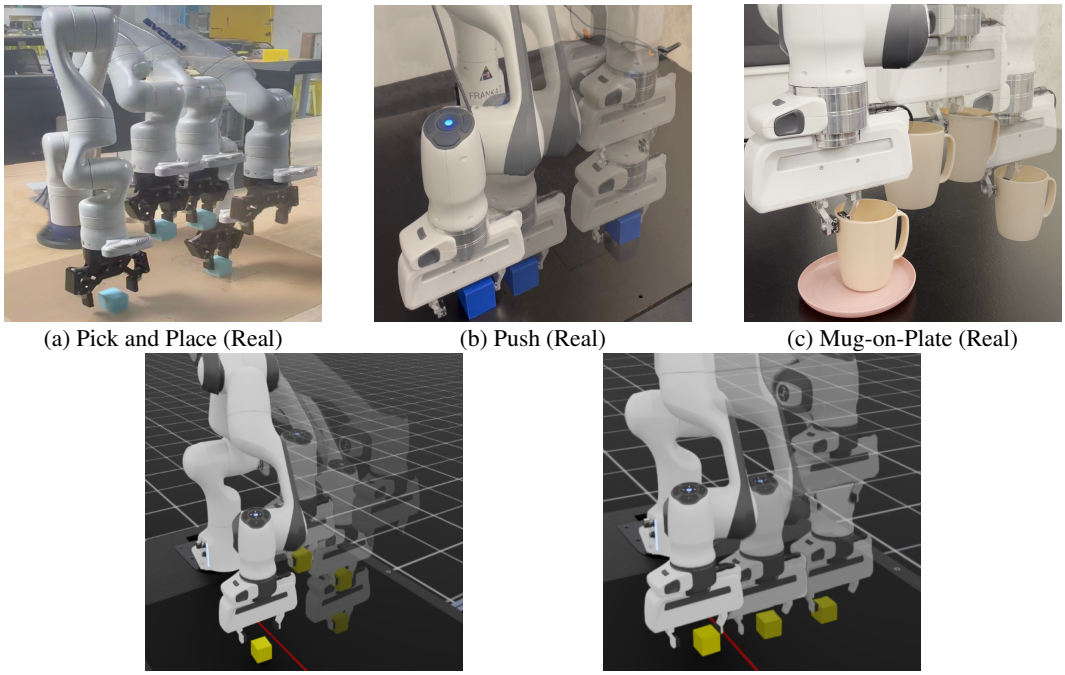
1. **[$-$P] (MAIRL (Sun et al., 2021))** simply removes the online planning component from MPAIL2. When acting in the environment, [$-$P] directly samples actions from a single-step policy $\pi_\phi(a_t|z_t)$. The policy is single-step as the multi-step policy is designed for reducing online planning 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 (Kostrikov et al., 2018))** 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) (Rahimi & Recht, 2007) and no encoding did not exhibit noticeable improvements.*

3. **[−PMO] (AIRL (Fu et al., 2018; Orsini et al., 2021))** takes the implementation of [−PM] and modifies the replay buffer such that only interactions within the previous episode are able to be sampled.
4. **[−O] (MPAIL (Han et al., 2025))** replaces the off-policy replay that MPAIL2 uses for the reward, value-ensemble, and policy updates with an on-policy rollout buffer, whereas the encoder and latent dynamics retain their off-policy JEP replay. All other components, encoder, latent dynamics, WGAN-GP latent-transition rewarddiscriminator, five-head target-network Q-ensemble trained against on-policy λ-return targets, and policy-seeded MPPI planner, are unchanged from MPAIL2. Hyperparameters are reported in Table 5. We note that this ablation subsumes a faithful MPAIL implementation, which underperformed in our preliminary experiments. To isolate the impact of off-policy training rather than confound it with component-level deficits, we retained the stronger MPAIL2 components listed above. We additionally verified that replacing the five-head Q-ensemble with MPAIL's on-policy V_ϕ value head, the value architecture that was specifically designed to maximize on-policy sample efficiency, did not improve results. Instead, we report [−O] as a strengthened MPAIL-style ablation that preserves the remaining MPAIL2 components.
5. **RLPD** is implemented as given in (Luo et al., 2025) but is also improved via an encoder in [−PM].
6. **BC** is implemented as a diffusion policy with a U-Net denoiser. Implementation and parameters are as given in (Chi et al., 2025). We train each policy for 500 epochs.

D Additional Results

D.1 Task Overview



(a) Pick and Place (Real) (b) Push (Real) (c) Mug-on-Plate (Real)

(d) Pick and Place (Sim) (e) Push (Sim)

Figure 9: Overview of evaluation tasks. The Pick and Place tasks (a,c) involve reaching the block (or mug), grasping it, lifting and placing it beyond a target line (or on the plate). The Push tasks (b,d) involve reaching the block and pushing it beyond a target line.

D.2 Learning Curves and Ablations

Figures 10 to 13 report additional learning curves in Gymnasium environments, ablations over key hyperparameters, sample efficiency relative to baselines, and per-episode success rate stability.

D.3 World Model Analysis

Figure 14 quantifies how much the explicit policy contributes to MPPI planning scores over training, revealing that the learned policy contributes relatively little weight to the final optimized online plan.

Figure 15 visualizes the planned rollouts in the Sim: Block Push (state) task where the learner's prediction of the block trajectory can be displayed. The block's static trajectory becoming more dynamic as the robot approaches provides evidence towards MPAIL2's implicitly causal structure.

MPAIL2's latent plans can be reconstructed for inspection and verification. We find that the model correctly learns over time that block motion can only be influenced through contact (Figure 15). When performed online, these visualizations can help practitioners identify where (dynamics, reward, value) the agent may benefit from more supervision or training. These visualizations provide clear qualitative evidence that the robot's world model is improving over time, as seen in Figure 16.

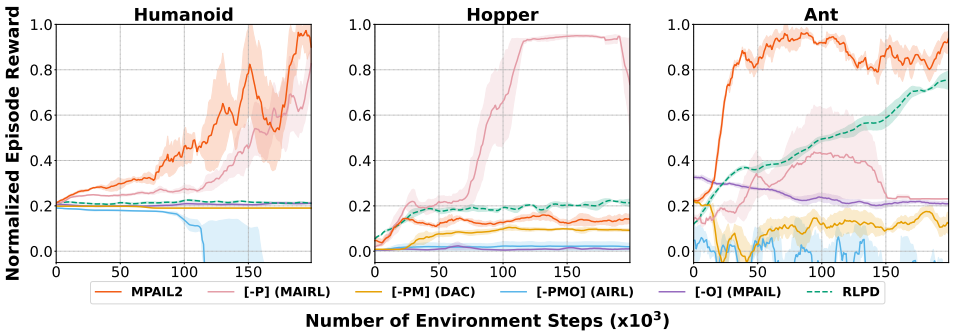


Figure 10: **Results in Gymnasium environments (Towers et al., 2025)**. All methods are provided 50 synthetic demonstration observations. RLPD is additionally provided dense reward and demonstration actions. Reward is normalized such that 1.0 reflects the average reward across the demonstrations.

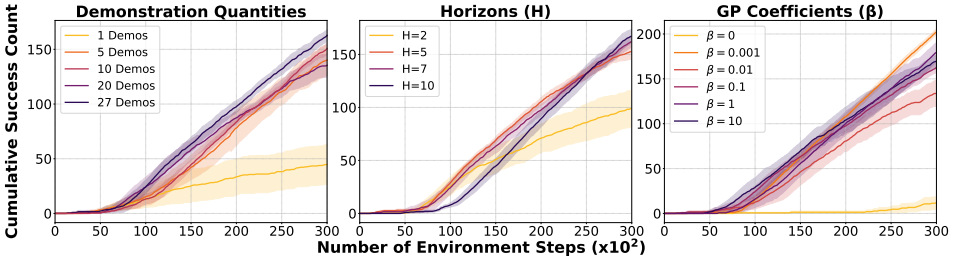


Figure 11: **MPAIL2 ablations** over number of demonstrations (left), horizon lengths (middle), and gradient penalty coefficients (right) in Sim: Block Push.

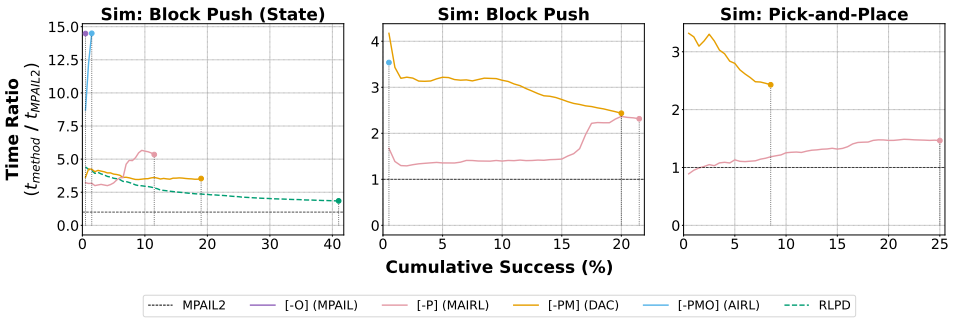


Figure 12: **Time efficiency of MPAIL2 compared to other methods**. We plot $t_{\text{algo}}(c)/t_{\text{MPAIL2}}(c)$ where $t(c)$ is the learning iteration at which the algorithm achieves a cumulative success percentage. With respect to a given baseline, the value reflects how many times faster MPAIL2 achieves a given success percentage. An endpoint indicates where the corresponding baseline achieves its maximum success percentage.

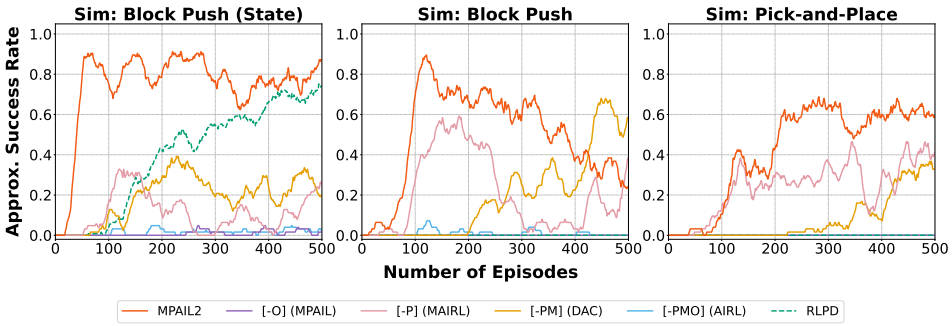


Figure 13: **Success rate stability**. We plot the average success rate of each method over a moving window of 25 episodes. A value of 1 indicates success on all episodes across 25 updates. This graph captures a method's stability across updates. In *Sim: Block Push (State)* and *Sim: Pick and Place*, 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 2.

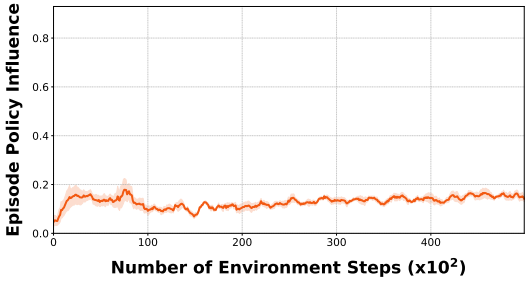


Figure 14: **Policy influence on planning.** Influence is computed as the fraction of total MPPI score attributable to policy-seeded rollouts among elite samples at the final planning iteration (see Algorithm 2), averaged over all timesteps in an episode.

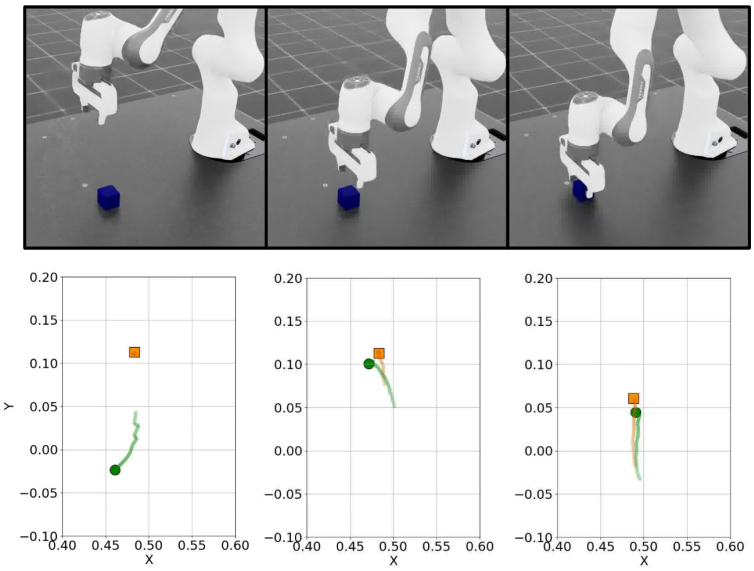


Figure 15: *Top.* MPAIL2 on Sim: Block Push (state) at three timesteps. *Bottom.* Predicted plans in the XY plane for the next one second. End-effector plans in green; block plans in orange. Block trajectories are static until contact, illustrating causal world modeling. *Planning is in latent space; a separately trained decoder is used for visualization.*

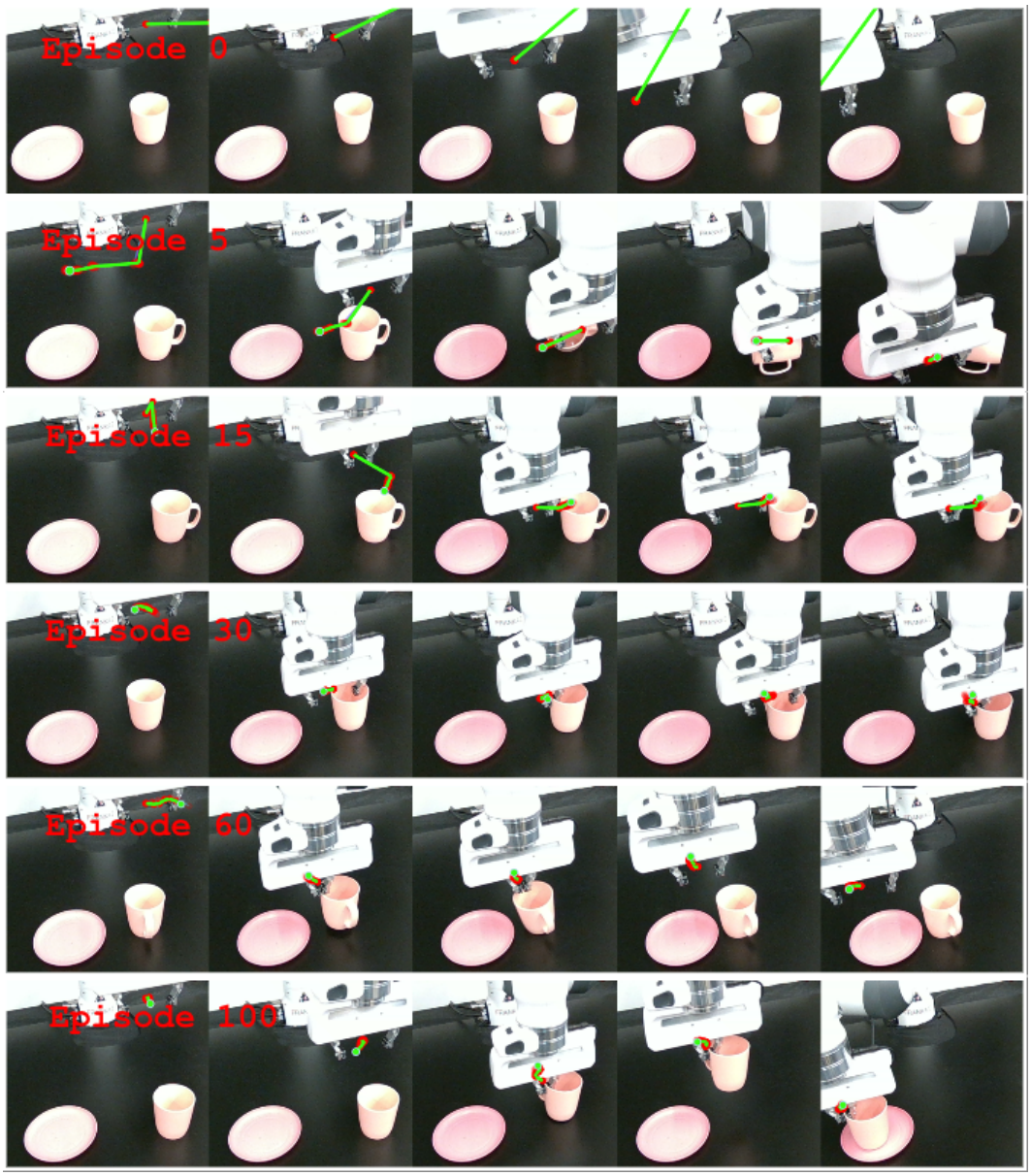


Figure 16: Real-world planning visualization on the Mug-on-Plate task. Each row shows five frames at training iterations 0, 5, 15, 30, 60, and 100. Red lines show the top-k end-effector plans; the green line is the optimal plan selected by the planner. Early episodes show unstructured, unmodeled, and exploratory behavior; by episode 100 the robot reaches, grasps, and places the mug onto the plate.

E Additional Discussion

The following discussions provide additional evidence, details, observations, and summaries beyond the main body.

E.1 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 2, 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 recovery 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 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 10). 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. 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.

E.2 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 (Pan et al., 2025)) despite optimizing equivalent objectives over similar data.

This intuition appears consistent in our visual manipulation experiments. MPAIL2 achieves high task performance with a small policy fraction (M/N = 0.05), and throughout Sim: Block Push the policy's contribution to the elite planning score remains below 20% (Figure 14), increasing slowly relative to task progress. The causal role of planning is further corroborated by the ablation: remov-

ing planning entirely ([−P], MAIRL) substantially degrades performance across all tasks (Table 2). Together these support the rationale from Section 2 that the explicit policy primarily seeds online optimization and supports off-policy value estimation, rather than serving as the primary execution mechanism.

Our results corroborate general efforts in iterative (implicit policy) methods as a generalizable alternative to explicit policies (Jain et al., 2025; Chi et al., 2025; Pan et al., 2025). 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 (Li & Chen, 2025) for an in-depth treatment of these equivalences.

E.3 Why Is MPAIL2 Significantly More Sample Efficient?

First, we consider the state-of-the-art RL baseline, **RLPD**. We emphasize that RLPD is competently implemented: given state observations, dense reward, and action-labeled demonstrations, it is the second strongest method on Sim: Block Push (State). Its difficulty is concentrated in the visual, real-world setting, where – within the small interaction budgets considered here, and despite the additional reward and action supervision it requires – it does not yet exhibit successes. This pattern is consistent with the well-known sample complexity of visual RL from scratch rather than an intrinsic failure of the method. We note that prior work reports only wall-clock time and does not include the number of interactions collected (Luo et al., 2025). While wall-clock times for pick-and-place appear similar in magnitude (hours) to prior work, our setups are not engineered to optimize real-world training time; we therefore expect that the real-world RLPD results in (Luo et al., 2025) collect substantially more interactions over the same period, and that RLPD might improve given more interaction. Accordingly, our comparison should be read as a statement about sample efficiency under matched, low interaction budgets and minimal supervision, rather than about the asymptotic capability of RLPD. 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.

The poor performance of [−O] should not be interpreted as evidence that planning or latent dynamics are insufficient. Rather, [−O] preserves both components and differs primarily in the use of on-policy reward, value, and policy updates. Under the matched low-interaction budgets considered here, the on-policy rollout buffer provides substantially narrower state-transition coverage than the accumulated replay buffer used by MPAIL2. This makes the adversarial reward more prone to rapidly fitting the current learner distribution and gives the value/policy updates fewer diverse transitions from which to learn recovery behavior. The result is that, despite retaining the planner and world model, [−O] does not match the sample efficiency or stability of MPAIL2.

E.4 Why Does MPAIL2 Exhibit Transfer?

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 (Rudin et al., 2025). 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 (Sutton & Barto, 2018). We believe the transfer experiment results in Figure 3 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 (Julian et al., 2021; Nagabandi et al., 2019b;a). 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 Appendix E.2, 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 randomly initialized (trained from scratch). The results of this experiment can be found in Figure 3. Results of MPAIL2 indicate that dynamics-only transfer capabilities remain about the same as the 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 transferable IRLfO, we believe that this work forms a strong foundation for further, more complex, experiments.

F Related Work

Learning from Observation (LfO). Methods in LfO learn to perform tasks from observation alone, theoretically free of hand-designed rewards and teleoperation. In practice, however, effective LfO approaches can require vast amounts of prior data, direct action supervision (e.g. teleoperation), or hours of demonstrations for seconds-long tasks (Liu et al., 2018; Wang et al., 2023; Cui et al., 2022; Rosete-Beas et al., 2023). These methods often produce large, unwieldy models relying on domain knowledge and embodiment-specific data (Wang et al., 2023; Collaboration, 2024), with domain-specific representations that limit generality across embodiments and settings — making extensions to complex robotic embodiments and novice end-user demonstrations particularly challenging.

Inverse Reinforcement Learning from Observation (IRLfO) is known to drastically (Han et al., 2025; Jain et al., 2025) reduce demonstration quantity by trading off direct-supervision via demonstrations for self-supervision via RL-like exploration and interaction (Abbeel & Ng, 2004; Ho & Ermon, 2016; Finn et al., 2016). However, needing to simultaneously optimize for both a reward function and a policy, IRL methods depend upon and exacerbate the already poor real-world interaction efficiency of existing RL algorithms. As a result, IRL approaches currently can require impractical amounts of online interaction on the order of thousands of policy rollouts (Sun et al., 2021; Han et al., 2025; Das et al., 2021; Finn et al., 2016; Torabi et al., 2019a). To the best of our knowledge, few works have investigated real-world IRL for robotics and none foregoing action supervision (IRLfO). Notable is the model-based IRL work in (Sun et al., 2021) demonstrating visual reaching in the real-world. Other works in the domain rely on prior modeling and pre-trained visual features such as (Finn et al., 2016; Das et al., 2021; Abbeel & Ng, 2004; Ratliff et al., 2009).

World Modeling & Online Planning. Of many approaches to improving sample efficiency in (I)RL, this work investigates latent world modeling and online planning. In particular, *planning-based* learners in RL have demonstrated remarkable potential and efficiency in simulation (Hansen et al., 2023; Jain et al., 2025). For real-world robot learning, planning-based learners have been shown to be substantially more robust than policy networks (Han et al., 2025). Planning-based methods and diffusion-based BC are theoretically unified as implicit policies that iteratively ascend an objective (Li & Chen, 2025). By learning the objective (the model-based return) using only (observed) demonstrations and interaction, MPAIL2 can be functionally viewed as a structured approach to steering diffusion-style policies with model-based (I)RL.

G Limitations and Future Work

Real-World Observational Learning. We have been careful not to claim to have achieved real-world *observational learning*, which is the ultimate goal. Returning to the initial baseball anecdote in Section 1, one may notice that our robot is not yet able to learn by observing *other* embodiments (e.g. a human moving an object). However, previous work in IRL and reward modeling has demonstrated cross-embodiment generalization when provided more data or through context translation (Zakka et al., 2022; Zhang et al., 2025a; Hwang et al., 2026; Liu et al., 2018). Language, in particular, is a highly accessible observation modality that has shown promise for intuitive task specification.

Pre-training for generality. Our transfer experiments reveal that all components of MPAIL2, even those that are task-dependent, may benefit from pre-training. Considering the generality of large pre-trained world (NVIDIA et al., 2025), policy (Black et al., 2025), and reward (Zhang et al., 2025b) models, their integration with the online world modeling framework of MPAIL2 points towards high potential for general, pre-trained learning and planning from observation. High variability in first-success times in both simulation and real-world experiments also indicate that supervision of the latent space deserves further attention. Pre-trained visual encoders, initialization methods, or auxiliary losses may help in reducing variability.

Recurrent, probabilistic modeling for planning with memory. The present encoder architecture assumes that the latent state can be instantaneously observed. However, real-world robotics involves challenges such as hardware delay, PID tracking errors, simultaneous localization and mapping (SLAM) which involve information dynamics which persists through time, which may require solutions in (Hafner et al., 2025; Jain et al., 2025).

Remaining manual supervision. While IRLfO eliminates key bottlenecks in robot learning, the designation of task observations remains manually supervised, although it is an extremely weak form of supervision. Like RL, IRLfO also still suffers from potentially unsafe real-world exploration. Similar to most manipulation systems (Collaboration, 2024), we constrain the workspace and torque limits of the robot during learning. However, towards more expressive and dynamic policies, safe exploration should be optimized at task level rather than via a lower-level blind constraint.

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.

Uncertainty, exploration, and safety. World models, as used in MPAIL2, help compartmentalize sources of uncertainty. In the case of MPAIL2, the 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 when the task is not the only objective.

Action and reward assumptions. While this work investigates more restrictive settings without access to demonstration action or hand-designed rewards, MPAIL2 can be extended to leverage additional prior knowledge as in (Peng et al., 2021) or in (Ball et al., 2023).