DREAM TO REMEMBER: CONTINUAL STEERING OF GENERALIST ROBOT POLICIES WITH WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generalist robot policies are trained with imitation learning and are bounded by the quality and coverage of their demonstrations. Reinforcement learning can improve existing skills and add new ones, either by fine-tuning the policy weights or by training a small module to steer the input noise of a frozen generative policy. In open-ended deployment, tasks arrive sequentially, and both approaches suffer from catastrophic forgetting. Fine-tuning on a task degrades performance on prior tasks, and steering only shifts forgetting to the shared steering module. We propose Dream to Remember (D2R), which continually steers a frozen generalist policy toward new tasks, one after another, while retaining its earlier skills. As the steering module learns a new task, a frozen pretrained world model generates imagined rollouts of prior tasks, anchoring the module to its earlier behavior. Prior tasks are thus rehearsed without a replay buffer or new data. When the world model is faithful enough, D2R learns the new task entirely in imagination. Otherwise, it learns the task in the environment while using the world model only to rehearse prior tasks. On CALVIN, D2R improves eight tasks in sequence entirely in imagination, raising average success from 63.7% to 74.8%, while LoRA fine-tuning on the same sequence drops it to 11.7%. With a frozen off-the-shelf generalist policy and world model, π_0 and A2World, D2R learns unseen LIBERO-90 tasks in the environment while keeping π_0's performance on the LIBERO suites it was fine-tuned on, even though the world model is not accurate enough for policy optimization. Finally, on a real Franka robot, D2R improves three tasks in sequence entirely in imagination while retaining the earlier improvements.

1 INTRODUCTION

A robot operating in a home or warehouse should continually improve existing skills, acquire new ones, and retain what it has already learned. However, even today's generalist robot policies trained on large demonstration corpora (Chi et al., 2025; Lipman et al., 2023; Black et al., 2024; NVIDIA et al., 2025) do not natively support continual adaptation that reliably preserves prior skills. Imitation learning limits these policies to demonstrated behavior, including its flaws and gaps, but those same demonstrations provide a strong prior worth preserving. Reinforcement learning can improve and extend that prior through the robot's own experience, whether gathered on the real robot (Physical Intelligence, 2025; Dong et al., 2026), in simulation (Chen et al., 2025), or through imagined rollouts of a learned world model (Chandra et al., 2025; Sharma et al., 2026), but doing so without catastrophic forgetting remains a central challenge.

Where adaptation occurs determines where forgetting can happen. Fine-tuning places it in the policy itself, so learning one task can alter behavior on others, whether the update modifies all weights or only an adapter (Hu et al., 2022; Zhu et al., 2026). Test-time action-selection methods sidestep this by adapting nothing persistent. They keep the policy frozen, sample multiple action proposals, score them against an external evaluator, such as a world model or a critic, then execute the best one (Nakamoto et al., 2025; Cui et al., 2026; Qi et al., 2025), but pay this significant search cost at every decision. Input steering keeps the policy frozen while amortizing the search into a learned module. Since a diffusion policy maps noise to actions, the module can learn which noise to sample to steer the policy toward better behavior (Wagenmaker et al., 2025). Steering, however, only moves forgetting to the shared steering network. Training it on a new task changes the noise sampled for previous ones, so forgetting reappears one level down.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

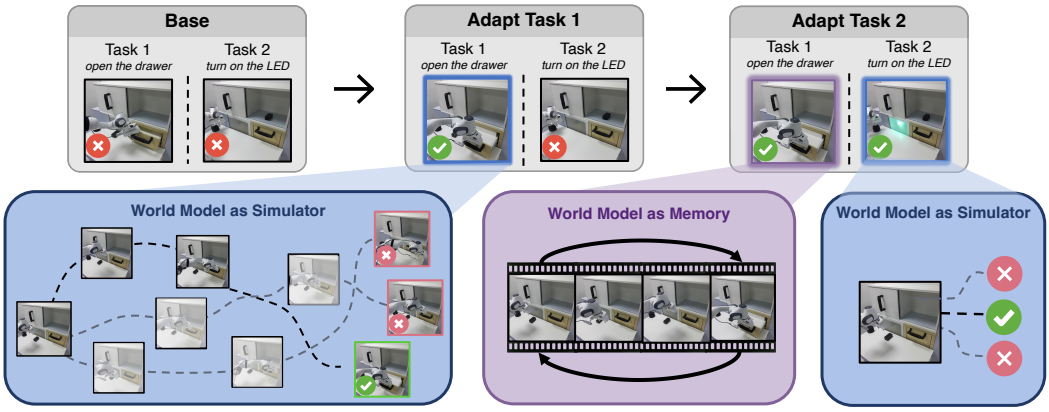


Figure 1: **Dream to Remember (D2R)**: We adapt a frozen generative policy sequentially by steering its input noise with a small module trained inside a learned world model. While the module learns the current task, the same world model regenerates imagined rollouts of prior tasks, and we regularize the module toward its previous behavior on those states. Skills thus accumulate without revisiting prior tasks on the robot or storing their trajectories. Rehearsal requires less world model fidelity than optimization, allowing even a model too coarse for learning a new task to preserve prior skills.

A common way to mitigate forgetting is rehearsal, where past tasks are revisited while learning a new one (Rolnick et al., 2019). Rehearsal can come from fresh robot interaction, at the cost of rerunning every past task in each round of adaptation, or from experience replay, which stores and reuses past trajectories (Buzzega et al., 2020), requiring a buffer of experience that grows over time (Du et al., 2026). Rehearsal therefore demands increasing robot interaction or storage.

We propose **Dream to Remember (D2R)**, which adapts a frozen generative policy through input steering and prevents the steering module from forgetting by rehearsing prior tasks in a learned world model. For the current task, whether refining or acquiring a skill, the steering module learns from real experience or, when the world model is sufficiently faithful, entirely from imagined rollouts. For prior tasks, the same world model provides rehearsal. Starting from an initial observation of each task, it regenerates rollouts under the steering behavior to be preserved, and the current module is regularized to match that behavior on the imagined states. Since the world model is frozen, the rehearsal distribution does not drift as new tasks arrive, and regenerating rollouts avoids costly repeated robot interaction and storing past trajectories. Importantly, we show that rehearsal requires less from the world model than learning a new task. A model that is not accurate enough for training the policy can still preserve prior skills by reproducing familiar states closely enough to anchor the steering module, whereas learning a new task requires reliable predictions wherever the module explores for reward. This allows D2R to use a frozen, off-the-shelf world model even when it was not trained on the target tasks. Such a model may not be accurate enough to learn a new task in imagination, but still sufficient to rehearse prior ones.

We evaluate D2R in three settings with different levels of world model fidelity. On CALVIN (Mees et al., 2022), where a world model trained on the robot's own play data is sufficiently faithful for policy optimization, D2R improves eight tasks sequentially and entirely in imagination, raising the average success rate across all tasks from 63.7% to 74.8%. Over the same sequence, LoRA fine-tuning reduces this average to 11.7%, while steering without rehearsal leaves it at the base level. On LIBERO-90, we instead use a frozen generalist policy, π_0 (Black et al., 2024), and a frozen pretrained world model, A2World (Huang et al., 2026), neither trained on the target tasks. Here, policies cloned from world model rollouts perform far worse than those cloned from the same real demonstrations, showing that the model is not accurate enough for policy optimization. Yet it remains effective for rehearsal, enabling D2R to learn unseen tasks through environment interaction while preserving π_0's performance on the LIBERO suites on which it was previously fine-tuned, ending 1.6 percentage points above its original success rate, whereas steering without rehearsal loses 30.8 points. Finally, on a real Franka robot, D2R sequentially improves three tasks using only imagined rollouts while retaining the gains on them.

In summary, our contributions are threefold. (i) **Steering in imagination.** We train the steering module of a frozen policy inside a learned world model, matching policy fine-tuning on the adapted

task while retaining other skills far better. (ii) **World model as memory.** We use a frozen world model to rehearse prior tasks by regenerating their rollouts on demand, enabling continual steering without repeated robot interaction or stored trajectories. (iii) **World-model rehearsal needs less fidelity than optimization.** We show that an off-the-shelf world model can be inaccurate for policy optimization yet sufficient for rehearsal, enabling continual adaptation with frozen π_0 and A2World. We make our code publicly available to support reproducibility.

2 Related Work

Improving pretrained generative policies. Prior work primarily adapts a pretrained generative policy to a single task. In weight space, reinforcement learning can update all weights through the denoising process (Ren et al., 2025; Chen et al., 2025) or be restricted to low-rank adapters (Hu et al., 2022). DiWA (Chandra et al., 2025) performs such fine-tuning inside a frozen latent world model (Hafner et al., 2020; DeMoss et al., 2023; Nematollahi et al., 2025), while generative video models have also been used as simulators for VLA fine-tuning (Xiao et al., 2025; Li et al., 2025; Sharma et al., 2026). In output space, methods either rank and select from sampled actions (Nakamoto et al., 2025; Qi et al., 2025; Cui et al., 2026; Kwok et al., 2025) or learn bounded action residuals (Dong et al., 2026). In input space, DSRL (Wagenmaker et al., 2025) learns the sampling noise with reinforcement learning on the robot, SCORE (Yu et al., 2026) learns it in simulation and transfers to the real robot, and FlowDAgger (Murray et al., 2026) finds that noise-space adaptation preserves held-out skills better than full fine-tuning, LoRA, or action residuals. These methods focus on individual-task adaptation. D2R instead carries noise-space steering across tasks and rehearses prior tasks in the world model, even when it is too coarse for policy optimization.

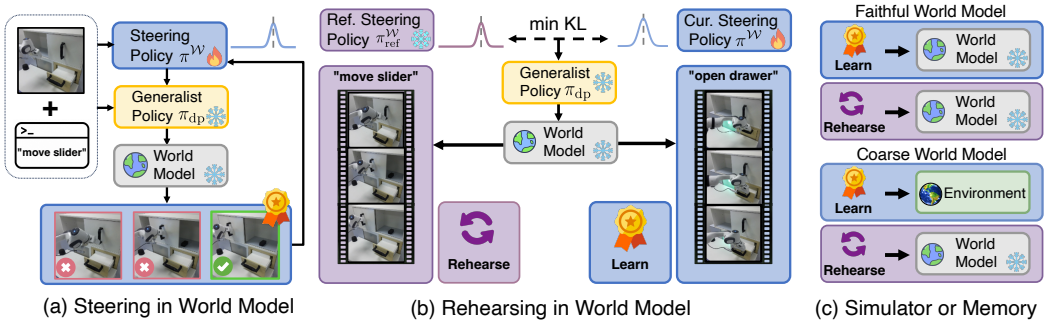
Continual learning without forgetting. Sequential learning requires balancing stability and plasticity (Kirkpatrick et al., 2017; Dohare et al., 2024; Khetarpal et al., 2022; Raghavan et al., 2024). Regularization limits changes to parameters or outputs important for earlier tasks (Kirkpatrick et al., 2017; Zenke et al., 2017; Li & Hoiem, 2018), parameter-isolation methods allocate separate capacity across tasks (Rusu et al., 2016; Mallya & Lazebnik, 2018), and rehearsal revisits stored or generated past data (Lopez-Paz & Ranzato, 2017; Rolnick et al., 2019; Buzzega et al., 2020; Shin et al., 2017; Yue et al., 2024). For robot policies, TAIL (Liu et al., 2024) adds task-specific adapters, CLARE and CORAL (Römer et al., 2026; Luo et al., 2026) expand lightweight modules with routing, and LOTUS (Wan et al., 2024) grows a library of skill policies, all reducing forgetting at the cost of model growth with each task. Forgetting also depends on the adaptation signal. On-policy RL through low-rank adapters can retain prior VLA skills well (Hu et al., 2026), consistent with broader findings that RL (Shenfeld et al., 2026) and LoRA (Biderman et al., 2024) can reduce forgetting, whereas sequential supervised fine-tuning shows far more forgetting and benefits from replay (He et al., 2026; Zhu et al., 2026). This retention relies on small updates within a large pretrained network and requires access to its weights, gradients, and on-policy interaction. D2R instead updates only a small steering module with forward access to the policy and uses a separate frozen world model for memory, keeping a single module without task identities and storing only initial observations of retained tasks.

3 Method

In this section, we present D2R, illustrated in Figure 2. We first define the continual setting, where tasks arrive sequentially and previously learned tasks must be retained (§3.1). We then describe how a small steering module adapts the frozen policy to the current task using imagined rollouts (§3.2), how world model rehearsal keeps the module from forgetting the other tasks (§3.3), and how a world model that is not accurate enough for policy optimization can still be used for rehearsal (§3.4).

3.1 Problem Formulation

Continual adaptation. We consider a robot with a pretrained generalist policy adapting to a sequence of language-conditioned tasks $\ell_1, \ldots, \ell_N$ that arrive one at a time. At stage k, the robot can gather experience only on ℓ_k, which may refine an existing skill or learn a new one. The goal is to improve success on ℓ_k while preserving the robot's existing skills, which form the *retention set* $\mathcal{R}_k$. Our goal is to preserve these skills without revisiting them on the robot or storing their trajectories, while assuming access only to a small set of initial observations per task.



(a) Steering in World Model (b) Rehearsing in World Model (c) Simulator or Memory

Figure 2: **Overview of D2R.** (a) **Steering in imagination:** A small steering module learns to shape the noise fed to the frozen policy through RL in a frozen world model. (b) **Rehearsal in the world model:** The world model imagines rollouts of prior tasks, where a KL penalty anchors the module to its earlier behavior. (c) **Simulator or memory:** A faithful world model supports both learning and rehearsal, while a coarser one can still support rehearsal as learning moves to the environment.

Frozen generative policy. Robotic control is naturally modelled as a visual POMDP, where the agent receives high-dimensional visual observations $o \in \mathcal{O}$ and task-dependent rewards R_k. To retain standard MDP notation, we define the environment as $\mathcal{M}_{\text{env}} = (\mathcal{S}, \mathcal{A}, P, R_k, \gamma)$, where $s \in \mathcal{S}$ denotes the Markovian state representation constructed from the observation and context history. The robot is controlled by a frozen generative policy π_{dp}, a diffusion or flow-matching model pretrained by imitation on multi-task demonstrations, that maps Gaussian noise $\boldsymbol{w} \sim \mathcal{N}(0, I)$, with $\boldsymbol{w} \in \mathcal{W} = \mathbb{R}^{H \cdot d}$, to an action chunk $\boldsymbol{a} \in \mathcal{A}^H$ conditioned on the state s and instruction ℓ. Under deterministic sampling, such as DDIM for diffusion (Song et al., 2020) or ODE integration for flow matching (Lipman et al., 2023), the action is a deterministic function of the initial noise and conditioning, denoted as $\boldsymbol{a} = \pi^{\mathcal{W}}_{\text{dp}}(s, \ell, \boldsymbol{w})$. We require only forward access to $\pi^{\mathcal{W}}_{\text{dp}}$, not its weights or gradients, so any off-the-shelf policy can be used.

World model. A learned world model simulates environment dynamics over state representations $s \in \mathcal{S}$. Concretely, an encoder E_ϕ maps observation histories to states $s_t = E_\phi(o_{\leq t})$, and a transition model $P_\phi(s' \mid s, \boldsymbol{a})$ predicts the next latent state from an action chunk (Hafner et al., 2020; Huang et al., 2026). Replacing the environment dynamics with P_ϕ defines the world model MDP $\mathcal{M}_{\text{wm}} = (\mathcal{S}, \mathcal{A}, P_\phi, R_k, \gamma)$, which shares the action space and reward with $\mathcal{M}_{\text{env}}$ but can be rolled out without environment interaction. In imagination, a learned success classifier evaluates R_k on latent states. We assume the world model is given and remains frozen throughout, whether trained on the robot's own data or pretrained as a generalist model without exposure to the target tasks.

Objective. Given the frozen policy and world model, we learn a *steering policy* $\pi^{\mathcal{W}}(\boldsymbol{w} \mid s, \ell)$ that maximizes return on the current task while preserving behavior on the retention set. Formally,

$$\max_{\pi^{\mathcal{W}}} J_k(\pi^{\mathcal{W}}) \qquad \text{s.t.} \qquad \text{KL}(\pi^{\mathcal{W}}(\cdot \mid s, \ell_j) \| \pi^{\mathcal{W}}_{\text{ref}(j)}(\cdot \mid s, \ell_j)) \leq \epsilon \quad \forall \ell_j \in \mathcal{R}_k, \tag{1}$$

where J_k is the expected return on ℓ_k, $\pi^{\mathcal{W}}_{\text{ref}(j)}$ is the steering behavior to preserve for task ℓ_j, and ϵ bounds its allowed deviation, while adaptation updates only $\pi^{\mathcal{W}}$. Section 3.2 describes how D2R maximizes return via steering in imagination, while Section 3.3 enforces the retention constraints.

3.2 Steering a Frozen Policy in Imagination

We first address the return term in Eq. 1, improving the frozen policy on the current task without updating its weights and, when possible, without environment interaction. Since $\pi^{\mathcal{W}}_{\text{dp}}$ is fixed, choosing a noise vector $\boldsymbol{w}$ induces an action, allowing the noise space to serve as the action space of a transformed MDP. We define $P^{\mathcal{W}}(s' \mid s, \ell, \boldsymbol{w}) = P\big(s' \mid s, \pi^{\mathcal{W}}_{\text{dp}}(s, \ell, \boldsymbol{w})\big)$ and $R^{\mathcal{W}}_k(s, \ell, \boldsymbol{w}) = R_k\big(s, \pi^{\mathcal{W}}_{\text{dp}}(s, \ell, \boldsymbol{w})\big)$. The resulting *latent-noise MDP* $\mathcal{M}^{\mathcal{W}}_{\text{env}} = (\mathcal{S}, \mathcal{W}, P^{\mathcal{W}}, R^{\mathcal{W}}_k, \gamma)$ treats the frozen policy as part of the environment, allowing RL to optimize directly over $\mathcal{W}$. DSRL (Wagenmaker et al., 2025) learns $\pi^{\mathcal{W}}$ in this MDP through interactions with the real environment. We instead replace the environment dynamics with the world model, yielding the *imagined latent-noise MDP*

$$\mathcal{M}^{\mathcal{W}}_{\text{wm}} = \big(\mathcal{S}, \mathcal{W}, P^{\mathcal{W}}_\phi, R^{\mathcal{W}}_k, \gamma\big), \qquad P^{\mathcal{W}}_\phi(s' \mid s, \ell, \boldsymbol{w}) = P_\phi\big(s' \mid s, \pi^{\mathcal{W}}_{\text{dp}}(s, \ell, \boldsymbol{w})\big). \tag{2}$$

When the world model is sufficiently faithful, we train $\pi^{\mathcal{W}}$ in $\mathcal{M}^{\mathcal{W}}_{\text{wm}}$ using only forward passes through the frozen policy and world model, with no environment interaction. Otherwise, we train it in $\mathcal{M}^{\mathcal{W}}_{\text{env}}$, as discussed in §3.4. In either case, the current task is learned by minimizing the loss $\mathcal{L}_{\text{RL}}$ of a standard RL algorithm over $\mathcal{W}$ (see Appendix, Eqs. S.3, S.5). A small network implements the steering module $\pi^{\mathcal{W}}$ which is orders of magnitude smaller than π_{dp} and maps the state s and instruction ℓ to a diagonal Gaussian over the noise,

$$\pi^{\mathcal{W}}(\boldsymbol{w} \mid s, \ell) = \mathcal{N}\big(\boldsymbol{w};\ \mu(s,\ell),\ \operatorname{diag}\sigma^2(s,\ell)\big), \qquad \sigma \in [\sigma_{\min}, \sigma_{\max}]. \tag{3}$$

Setting $\mu = 0$ and $\sigma = 1$ recovers the original noise distribution, allowing steering to be disabled. Since every action is decoded by the frozen $\pi^{\mathcal{W}}_{\text{dp}}$, steering changes which actions are selected without modifying the policy itself. When trained in imagination, however, steering can exploit prediction errors in the learned world model. Following DiWA (Chandra et al., 2025), we add a behavior-cloning term $\mathcal{L}_{\text{BC}}$ that keeps decoded actions close to those produced from prior noise (Appendix, Eq. S.6).

3.3 Rehearsal in the World Model

Training the steering module on the current task can change how it steers other tasks, causing forgetting. To preserve prior skills, we relax the hard retention constraint in Eq. 1 into a soft penalty between the current steering policy and a task-specific reference. For each task $\ell_j \in \mathcal{R}_k$, the reference $\pi^{\mathcal{W}}_{\text{ref}(j)}$ is $\pi^{\mathcal{W}}_{k-1}$ if the task was previously steered, and the prior $\mathcal{N}(0, I)$ otherwise. Thus, pretrained skills are anchored to the unsteered policy, while previously steered skills are anchored to their earlier steering behavior. Since the current steering policy and its reference are both diagonal Gaussians, their KL divergence has a closed form. We average it over the retention set,

$$\mathcal{L}_{\text{KL}}(\pi^{\mathcal{W}}_k) = \frac{1}{|\mathcal{R}_k|} \sum_{\ell_j \in \mathcal{R}_k} \mathbb{E}_{s \sim \mathcal{D}^j_{\text{reh}}} \Big[\text{KL}\big(\pi^{\mathcal{W}}_k(\cdot \mid s, \ell_j) \,\big\|\, \pi^{\mathcal{W}}_{\text{ref}(j)}(\cdot \mid s, \ell_j)\big)\Big]. \tag{4}$$

Here, $\mathcal{D}^j_{\text{reh}}$ contains states of task ℓ_j generated by the world model for rehearsal. For each task $\ell_j \in \mathcal{R}_k$, a rollout is seeded from one of the task's initial observations, encoded by E_ϕ , and unrolled with $P^{\mathcal{W}}_\phi$, while the reference policy $\pi^{\mathcal{W}}_{\text{ref}(j)}$ selects the noise at each step. Rehearsal requires no reward, and rollouts run for a fixed horizon unless a success signal is available for early termination. Rollouts are regenerated at every update rather than stored, requiring world model compute but no trajectory buffer. Since the world model is frozen, the rehearsal distribution does not drift as the number of tasks grows. Rollouts follow the reference policy, so the KL penalty of Eq. 4 anchors the current module on states relevant to the retained skill.

3.4 World Model as Simulator or Memory

D2R uses the world model either as a simulator or as memory, and the two roles require different fidelity. As a simulator, the model must be accurate because the steering policy is optimized against it and can exploit prediction errors. As memory, it only needs to generate states representative of retained tasks, since the KL penalty matches a reference on those states rather than optimizing a policy through the world model dynamics. Whether the model serves as simulator or memory therefore depends on its coverage of the task. A model trained on data that covers the task dynamics can support optimization, while one outside that coverage may still serve as memory, so the same world model can play different roles across tasks. This assignment determines where the current task is learned, in the imagined latent-noise MDP $\mathcal{M}^{\mathcal{W}}_{\text{wm}}$ when the model serves as a simulator and in the environment latent-noise MDP $\mathcal{M}^{\mathcal{W}}_{\text{env}}$ when it serves only as memory. Both regimes admit any RL algorithm over $\mathcal{W}$, but their interaction costs differ. In $\mathcal{M}^{\mathcal{W}}_{\text{wm}}$, rollouts are cheap and parallelizable, so we use on-policy PPO with abundant on-policy data (Schulman et al., 2017; Chandra et al., 2025). In $\mathcal{M}^{\mathcal{W}}_{\text{env}}$, every transition is a real episode, so we use off-policy SAC and reuse real experience through replay (Haarnoja et al., 2018; Wagenmaker et al., 2025). Rehearsal is the same in both regimes, since retained tasks are always rehearsed in the world model and Eq. 4 needs no environment interaction.

At stage k, we train a new module $\pi^{\mathcal{W}}_k$ from scratch with $\pi^{\mathcal{W}}_{k-1}$ frozen as the reference, minimizing

$$\mathcal{L}(\pi^{\mathcal{W}}_k) = \mathcal{L}_{\text{RL}} + \alpha\, \mathcal{L}_{\text{BC}} + \lambda\, \mathcal{L}_{\text{KL}}, \tag{5}$$

where α is the behavior-cloning coefficient, set to 0 for training with environment interaction, and λ is the retention coefficient shared across retained tasks. After stage k, $\pi^{\mathcal{W}}_k$ becomes the sole steering module for all tasks seen so far, and $\pi^{\mathcal{W}}_{k-1}$ is discarded (See Algorithm S.1 in the Appendix).

4 SIMULATION EXPERIMENTS

We evaluate three aspects of D2R in simulation, namely steering in imagination for improving a task without forgetting others (§4.2), continual steering with world model rehearsal for learning tasks sequentially while retaining earlier ones (§4.3), and rehearsal with a coarse world model (§4.4).

4.1 EXPERIMENTAL SETUP

We evaluate D2R on two benchmarks spanning its two regimes, where the world model serves as both simulator and memory or as memory alone (§3.4). We follow two protocols. In *single-task adaptation*, a method is trained on one task and evaluated on all tasks, measuring improvement on adapted task and retention of the others. In *sequential adaptation*, a model learns tasks sequentially and is evaluated on all tasks after each stage, measuring forgetting and retention over the sequence.

CALVIN. On CALVIN (Mees et al., 2022), adaptation occurs entirely in imagination. The world model is a recurrent state-space model (RSSM) trained on task-agnostic play data from environment, while the frozen policy is a language-conditioned multi-task diffusion policy trained on 150 demonstrations per task. Since imagined rollouts are cheap and parallel, each task is improved with on-policy PPO in the world model, rewarded by a latent success classifier, and eight tasks are adapted in isolation and in sequence, rehearsing the other seven at each adaptation (see Appendices S.3 and S.4 for implementation details and the adaptation order).

LIBERO. On LIBERO (Liu et al., 2023), we use an off-the-shelf policy and world model. The policy is the public π_0 checkpoint (Black et al., 2024), fine-tuned on the Spatial, Object, Goal, and Long suites, while the world model is A2World (Huang et al., 2026), pretrained on 2.1M real-robot trajectories and fine-tuned on the same suites, with neither exposed to LIBERO-90. We learn eight LIBERO-90 tasks sequentially (order in Appendix S.4). Since A2World is not accurate enough for policy optimization on these tasks (see §4.4), each is learned in the environment, where every transition requires real interaction. All methods therefore learn off-policy with SAC and replay under the same interaction budget, while A2World serves only for rehearsal. Each stage rehearses the previously learned LIBERO-90 tasks and the four suites in A2World with reward-free rollouts.

D2R. In both benchmarks, the steering module is trained from scratch at each stage. To seed rehearsal rollouts, we store only the first observation of each demonstration, giving 150 observations per task on CALVIN and 50 on LIBERO (see Appendix S.3.5 for hyperparameters).

Baselines. We compare against the frozen base policy and three families of alternatives, corresponding to the three spaces in which adaptation can occur. Weight-space adaptation is represented by **DiWA** (Chandra et al., 2025), which fine-tunes all policy weights in imagination, in single-task adaptation, and by **LoRA** fine-tuning (Hu et al., 2022) in sequential adaptation. To separate the effect of rehearsal from the adaptation space, **DiWA + rehearsal** augments DiWA with our world model rehearsal, regularizing the fine-tuned policy toward the frozen base policy on imagined rehearsal states. LoRA trains low-rank adapters, on the full policy network on CALVIN and on the action expert of π_0 on LIBERO, with PPO in imagination on CALVIN and off-policy learning in the environment on LIBERO. Output-space adaptation is represented by **DP-TTC**, a test-time selection method on CALVIN in the style of Cui et al. (2026). It samples action chunks from the frozen policy, imagines their outcomes in the world model, scores them with the success classifier, and executes one of the highest-scoring candidates. Input-space adaptation without rehearsal is represented by **DSRL** (Wagenmaker et al., 2025), trained in the world model on CALVIN and in the environment on LIBERO. We also consider three ablations that add the rehearsal penalty to DSRL and differ only in the states it is applied to. **DSRL + instruction KL** applies the rehearsal penalty to current-task observations conditioned on retained-task instructions, without storing or generating states from the retained tasks. **DSRL + first-obs KL** applies it only at the stored first observations, without rolling them forward through the world model. **DSRL + replay KL** applies the penalty to retained-task rollouts collected once per stage with the previous steering module and stored (real trajectories on LIBERO and imagined trajectories on CALVIN). **D2R** instead applies the penalty to retained-task rollouts regenerated in the world model. All methods start from the same base policy and use the same world model and seed frames where applicable (see Appendix S.4.5 for baseline details).

Metrics. We report three quantities. The first is the **success rate**. On CALVIN, results average success over all validation initial states per task and three training seeds. On LIBERO, they average

Table 1: **Single-task adaptation on CALVIN.** Success on the adapted task and retention, defined as the mean change in success over all tasks relative to the base policy in percentage points (▲ gain, ▼ loss). Baselines trade retention for task gains, while D2R is the only method to improve both.

ID	Task	Base DP	DP-TTC	DiWA		DiWA + rehearsal		DSRL		D2R (Ours)	
		Success	Success	Success	Retention	Success	Retention	Success	Retention	Success	Retention
1	close-drawer	77.0±5.0	79.3±8.6	88.5±13.1	▼41.7	94.3±5.0	▼11.4	**98.9±5.0**	▼3.7	97.7±5.0	▲**2.2**
2	move-slider-left	39.6±16.2	43.8±13.5	49.0±4.5	▼52.9	59.4±3.0	▼12.8	**62.5±15.5**	▼5.1	54.2±33.8	0.0
3	move-slider-right	42.5±32.4	49.4±9.9	69.0±8.6	▼48.2	74.7±5.0	▼16.8	**85.1±9.9**	▼0.8	79.3±13.1	▲**3.7**
4	open-drawer	50.0±8.3	41.1±4.8	**87.8±12.7**	▼46.9	63.3±21.9	▼21.8	78.9±4.7	▼3.1	77.8±9.6	▲**3.6**
5	turn-off-LED	85.4±16.2	87.5±13.4	**96.9±7.8**	▼35.9	94.8±9.0	▼10.4	91.7±4.5	▼7.0	86.5±9.0	▲**0.5**
6	turn-off-lightbulb	40.2±26.2	43.7±9.9	**60.9±13.1**	▼43.4	51.7±12.1	▼10.2	48.3±8.6	▼5.5	58.6±21.6	▲**0.6**
7	turn-on-LED	82.8±14.8	85.1±5.0	90.8±13.1	▼35.2	**96.6±8.6**	▼10.0	92.0±13.1	▼10.5	88.5±9.9	▲**1.2**
8	turn-on-lightbulb	91.9±4.4	90.9±13.0	90.9±7.5	▼47.5	88.9±19.9	▼8.1	**97.0±7.5**	▼7.0	96.0±4.4	▲**1.1**
	Average	63.7±6.5	65.1±1.9	79.2±3.7	▼43.9	78.0±3.1	▼12.7	**81.8±2.9**	▼5.3	79.8±3.8	▲**1.6**

30 evaluation episodes from the benchmark's initial states over three evaluation seeds for one model. We report 95% confidence intervals throughout. We define **retention** as the change in success, in percentage points, relative to the frozen base policy. In sequential adaptation, we define **forgetting** as the drop relative to the stage at which the task was learned (Appendix S.4.6). Retention measures performance relative to the original policy, while forgetting measures only the loss after learning.

4.2 Steering in Imagination Without Forgetting

Table 1 reports single-task adaptation on the eight CALVIN tasks, with all methods trained entirely in the world model and no environment interaction. D2R improves the frozen policy on every task, raising average success from 63.7% (Base DP) to 79.8%, and is the only method with positive retention. Average retention is -43.9 points for DiWA, -12.7 for DiWA with rehearsal, -5.3 for DSRL, and $+1.6$ for D2R. In success rate, D2R matches DiWA, which fine-tunes the policy weights in imagination, showing that input-noise steering can achieve the same gains without updating the policy. DSRL, which steers without rehearsal, scores two points higher on the target task, while D2R trades this small margin for retaining prior skills. DP-TTC leaves the policy unchanged and thus induces no forgetting, but underperforms the adaptation methods despite querying the world model and classifier at every step, as it can only select among actions proposed by the frozen policy.

These results reflect the consequences of the adaptation space. Fine-tuning updates the shared policy weights, so learning one task changes the actions produced on all others, and DiWA loses retention regardless of which task is adapted. Steering leaves the policy frozen and changes only which noise is fed to it, so every action is still produced by the same pretrained policy (Yu et al., 2026; Murray et al., 2026), while the rehearsal penalty constrains the module on the other tasks. Adding the same rehearsal to DiWA recovers most, but not all, of the retention loss. Rehearsal constrains the fine-tuned policy only on imagined states visited by the reference, whereas changed weights can still alter behavior elsewhere. In the noise space, an unsteered task also has a simple reference, the Gaussian prior. Rehearsal thus benefits both adaptation spaces, but is most effective when combined with steering (see Figure S.3 for retention heatmaps).

4.3 Continual Steering Across Tasks

We next ask whether a single steering module can learn tasks one after another without forgetting. The two benchmarks test this in different ways. On CALVIN, the frozen policy already performs all eight tasks in a single scene, and the module refines them one at a time in imagination. On LIBERO, all eight tasks are new to the policy and span different scenes. Since the world model is not accurate for policy optimization, each task is learned in the environment while the world model is used only for rehearsal. Table 2 reports the average **success** over all tasks at the end of the sequence, together with **retention** and **forgetting** as defined above (see Table S.3 for per-stage columns). Figure 3 shows the LIBERO sequence stage by stage, where each row is a stage, the outlined diagonal cell gives the success on the task learned at that stage, and the cells below give how many points that task has lost since. D2R learns each sequence with a single module, raising average success from 63.7% to 74.8% on CALVIN and from 10.4% to 74.9% on LIBERO, with only 5.7 and 8.3 points of forgetting, while the four LIBERO suites remain at their base performance ($+1.6$ retention).

Table 2: Continual steering over eight tasks on CALVIN (in imagination) and LIBERO (in the environment). Storage is what each method keeps besides the frozen policy after the full sequence.

	CALVIN			LIBERO			
Method	Success↑	Retention↑	Forgetting↓	Success↑	Retention↑	Forgetting↓	Storage↓
Base	63.7 ± 6.5	0.0	0.0	10.4 ± 2.7	0.0	0.0	–
LoRA	11.7 ± 1.9	-52.0	53.2	1.0 ± 1.6	-41.6	61.4	–
DSRL	63.6 ± 4.8	-0.1	**1.6**	16.9 ± 5.2	-30.8	78.3	–
+ instruction KL	70.9 ± 8.1	$+7.2$	3.6	24.9 ± 2.6	-23.3	61.4	–
+ first-obs KL	65.1 ± 3.6	$+1.4$	12.9	36.4 ± 0.6	-20.9	52.5	∼400 frames
+ replay KL	74.0 ± 3.5	$+10.3$	5.8	73.9 ± 6.3	$+\mathbf{2.0}$	11.6	∼59k frames
D2R (Ours)	$\mathbf{74.8 \pm 3.7}$	$+\mathbf{11.1}$	5.7	$\mathbf{74.9 \pm 2.6}$	$+1.6$	**8.3**	∼400 frames
Per-task modules	81.8 ± 2.9	$+18.1$	0.0	91.3 ± 1.0	$+80.9$	0.0	8 modules
Multi-task module	67.3 ± 3.9	×	×	54.6 ± 8.5	×	×	1 module

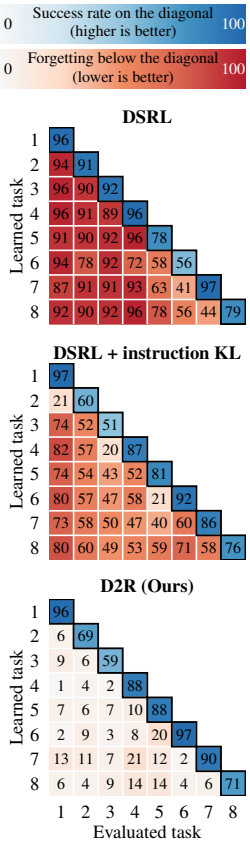


Figure 3: **Continual steering on LIBERO.** Diagonal cells show the success when each task is learned, and cells below show the forgetting, the points lost after each later stage. DSRL forgets each task as soon as the next is learned, instruction KL reduces this, and D2R keeps the tasks to the end.

Neither weight updates nor steering without rehearsal achieves this balance of adaptation and retention. LoRA fine-tuning learns each task but quickly forgets earlier ones, losing 53.2 points on CALVIN and 61.4 on LIBERO, and finishes well below the base policy on both benchmarks. DSRL, which steers without rehearsal, also ends near the base level on both, but for different reasons. On LIBERO, it forgets 78.3 points and reduces performance on the four pretrained suites by 30.8 points. The DSRL panel of Figure 3 similarly shows each LIBERO task turning red one stage after it is learned. On CALVIN, however, DSRL forgets only 1.6 points because it never improves much beyond the base policy. Without rehearsal, adaptation therefore either overwrites earlier skills or fails to accumulate gains across tasks. The three ablations differ from D2R only in the states used for rehearsal, revealing what the memory must provide. DSRL + instruction KL uses no states from retained tasks and reaches 70.9% on CALVIN but only 24.9% on LIBERO, a gap we explain in §4.4. DSRL + first-obs KL uses only the stored first observations, and forgets 12.9 points on CALVIN and 52.5 on LIBERO. DSRL + replay KL uses full rollouts of the retained tasks, collected once per stage and stored, and comes closest. D2R achieves comparable performance while keeping about 400 frames over the whole sequence, roughly 150 times fewer than a trajectory buffer. Rehearsal therefore requires states that retained skills actually visit, but those states need not be real or stored. We also compare against two reference settings. Task-specific modules, one per task with the task identity given at test time, reach 81.8% on CALVIN and 91.3% on LIBERO and cannot forget by construction. D2R comes within 7 and 16 points of them using a single module. A single module trained jointly on all tasks reaches only 67.3% and 54.6% despite a larger budget, because joint training must optimize sparse rewards for all eight tasks simultaneously, whereas rehearsal anchors each retained task to a reference that already solves it. D2R keeps one module and a few initial frames per task, and among methods that neither grow with the sequence nor store trajectories, it is the only one that both learns the tasks and retains them.

4.4 A Coarse World Model as Memory

On LIBERO the world model serves only as memory, because it is not accurate enough to learn in. A2World was fine-tuned on the four LIBERO suites but never saw LIBERO-90. To test whether it can serve as a simulator for the target tasks, we clone real demonstrations of each task and A2World-imagined replays of the same demonstrations with the same behavior-cloning recipe (Appendix S.5.7). Policies cloned from real demonstrations reach 74.9% on the LIBERO-90 tasks and those cloned from imagined replays only 19.6%, whereas on the four suites the two are within a few points. The model reproduces the dynamics it was trained on but not the target tasks, so a steering policy optimized inside it would exploit its prediction errors rather than learn the task. The same model nonetheless serves as memory. The three panels of Figure 3 differ only in what anchors the

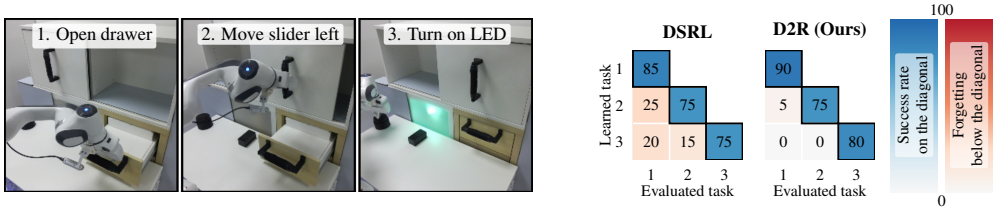


Figure 4: **Continual steering on a real Franka robot.** Heatmaps follow Figure 3, with the success rate when each task is learned on the diagonal and the forgetting score below it. D2R retains prior gains across the sequence, whereas DSRL without rehearsal loses about half of them. Adaptation occurs entirely in imagination, with the robot used only for evaluation.

module, with darker red indicating stronger forgetting. Without rehearsal, every column of DSRL turns dark red within one stage. With instruction KL the red is lighter, and with D2R the columns remain pale to the end. The difference is where the penalty is applied. Instruction KL anchors the module on observations from the current task paired with retained instructions. This works on CALVIN, where all tasks share a scene, but not on LIBERO, where scenes change and current-task observations do not cover retained-task states. D2R obtains these states by unrolling each retained task in A2World from stored initial frames. Although these imagined trajectories are too inaccurate to learn from, their states are sufficient for anchoring, and rehearsal on them matches rehearsal on real stored trajectories. A model accurate enough to optimize against is therefore a stronger requirement than one accurate enough to rehearse in. An off-the-shelf world model can serve as memory without any fine-tuning on the tasks being learned, for the pretrained suites and the new tasks alike.

5 Real-World Experiments

Setup. We evaluate D2R on a Franka Emika Panda robot in a tabletop environment with a cabinet and drawer, observed by a static and a gripper-mounted camera. The world model is an RSSM trained once on six hours of teleoperated play data and then frozen. The base policy is a single language-conditioned diffusion policy trained on 100 demonstrations for each of seven tabletop tasks (see Appendix S.4.2), with one success classifier for each task trained on the same demonstrations. The policy, classifiers, and world model remain frozen throughout. We adapt three of these tasks sequentially, open drawer, move slider left, and turn on LED, rehearsing prior tasks at each later stage. As on CALVIN, the world model is trained on the robot's own play data and is sufficiently faithful for policy optimization, so learning and rehearsal take place in imagination with PPO, and the robot is used only for evaluation. After each stage, we run 20 episodes per adapted task from fixed initial configurations and report success rates. We compare against the frozen base policy and DSRL trained in the same world model without rehearsal (see Appendix S.3.5 for hyperparameters).

Results. Figure 4 shows the three adaptation stages. For each adapted task, steering in imagination improves execution on the real robot by 40, 20, and 30 points over a frozen base policy that initially succeeds on 50 to 55% of episodes. The methods diverge only after later tasks are learned. Under D2R, those gains are preserved to the end of the sequence, for an average of 30 points above the base policy. Under DSRL, without rehearsal, about half of them are lost, leaving an average of 15 points. World model rehearsal thus preserves prior gains on the real robot without robot time for learning.

6 Conclusion

We presented D2R, a framework for adapting a pretrained generative robot policy to tasks arriving sequentially without forgetting its existing skills. D2R steers the frozen policy in its noise space with a single small module and rehearses the retained skills on rollouts imagined in a frozen world model. Adaptation therefore requires no weight updates, no stored trajectories, and no robot interaction for rehearsal. On two simulated benchmarks and a real Franka robot, steering matches weight fine-tuning on the adapted task while preserving prior skills substantially better. Our experiments further show that a world model needs lower fidelity for rehearsal than for policy optimization, so a model too inaccurate to learn in can still serve as memory. However, D2R still stores a few initial frames per task to prompt the world model, which future work could instead generate on demand.

AI USE STATEMENT

In preparing this paper, we used large language models (LLMs) as general-purpose writing aids, including proofreading, language polishing, and suggestions for improving manuscript clarity. We did not use generative AI tools to formulate or prove mathematical claims, propose or refine hypotheses, develop theoretical models, generate synthetic datasets, clean or reformat datasets, design the methodology or experiments, implement the method, interpret experimental results, or perform qualitative or thematic data analysis. The remaining required-disclosure tasks were not applicable. All AI-assisted text was reviewed and verified by the authors. The research ideas, problem formulation, methodology, experimental design, data analysis, and code development were conceived and executed by the authors. Generative AI was not used to generate any scientific content or conclusions. The authors take responsibility for the entire content of the paper.

REPRODUCIBILITY STATEMENT

We have taken several measures to maximize the reproducibility of our work. The main paper describes the method in Section 3, and Appendices S.1 to S.5 give the training objectives, the algorithm, the architectures and hyperparameters, the task sequences, the evaluation protocol, and the baseline definitions. We provide an anonymized codebase and videos of the real-robot rollouts in the supplementary material. In simulation we use the public CALVIN and LIBERO benchmarks and the released π_0 and A2World checkpoints, and we will release the Franka data upon acceptance.

REFERENCES

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. LoRA learns less and forgets less. *Transactions on Machine Learning Research*, 2024.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Akshay L Chandra, Iman Nematollahi, Chenguang Huang, Tim Welschehold, Wolfram Burgard, and Abhinav Valada. Diwa: Diffusion policy adaptation with world models. In *Conference on Robot Learning*, pp. 3378–3400. PMLR, 2025.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, et al. πrl: Online rl fine-tuning for flow-based vision-language-action models. *arXiv preprint arXiv:2510.25889*, 2025.

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, 2025.

Hanchen Cui, Sergio Arnaud, Arjun Majumdar, Daniel Dugas, Elie Aljalbout, Karthik Desingh, Krishna Murthy Jatavallabhula, and Franziska Meier. Dreamsteer: Latent world models can steer vla policies during deployment without any finetuning. *arXiv preprint arXiv:2607.02865*, 2026.

Branton DeMoss, Paul Duckworth, Jakob Foerster, Nick Hawes, and Ingmar Posner. Ditto: Offline imitation learning with world models. *arXiv preprint arXiv:2302.03086*, 2023.

Shibhansh Dohare, J Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026): 768–774, 2024.

Perry Dong, Kuo-Han Hung, Tian Gao, Dorsa Sadigh, and Chelsea Finn. Expo-ft: Sample-efficient reinforcement learning finetuning for vision-language-action models. *arXiv preprint arXiv:2605.25477*, 2026.

Maximilian Du, Zhanyi Sun, Chen Xu, Paarth Shah, Masha Itkina, and Shuran Song. Memory anchors for continual robot learning. *arXiv preprint arXiv:2608.26545*, 2026.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

Qijun He, Yuxuan Li, Mingqi Yuan, Xiaoquan Sun, Wen-Tse Chen, Jeff Schneider, and Jiayu Chen. Continual policy consolidation for lifelong robot learning. *arXiv preprint arXiv:2601.22475*, 2026.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Jiaheng Hu, Jay Shim, Chen Tang, Yoonchang Sung, Bo Liu, Peter Stone, and Roberto Martín-Martín. Simple recipe works: Vision-language-action models are natural continual learners with reinforcement learning. *Reinforcement Learning Journal*, 2026.

Ze Huang, Jiahui Zhang, Hairuo Liu, Chenxi Zhang, Ran Cheng, and Li Zhang. Learning transferable dynamics priors from action to world modeling. In *European Conference on Computer Vision*. Springer, 2026.

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Jacky Kwok, Christopher Agia, Rohan Sinha, Matt Foutter, Shulu Li, Ion Stoica, Azalia Mirhoseini, and Marco Pavone. Robomonkey: Scaling test-time sampling and verification for vision-language-action models. In *Conference on Robot Learning*. PMLR, 2025.

Hengtao Li, Pengxiang Ding, Runze Suo, Yihao Wang, Zirui Ge, Dongyuan Zang, Kexian Yu, Mingyang Sun, Hongyin Zhang, Donglin Wang, and Weihua Su. Vla-rft: Vision-language-action reinforcement fine-tuning with verified rewards in world simulators. *arXiv preprint arXiv:2510.00406*, 2025.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Zuxin Liu, Jesse Zhang, Kavosh Asadi, Yao Liu, Ding Zhao, Shoham Sabach, and Rasool Fakoor. TAIL: Task-specific adapters for imitation learning with large pretrained models. In *International Conference on Learning Representations*, 2024.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Yuankai Luo, Woping Chen, Tong Liang, and Zhenguo Li. Coral: Scalable multi-task robot learning via lora experts. *arXiv preprint arXiv:2603.09298*, 2026.

Arun Mallya and Svetlana Lazebnik. PackNet: Adding multiple tasks to a single network by iterative pruning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 2022.

Michael Murray, Daphne Chen, Simran Bagaria, Dean Fortier, Tess Hellebrekers, Galen Mullins, Harshavardhan Gajarla, Oier Mees, Maya Cakmak, and Andrey Kolobov. FlowDAgger: Human-in-the-loop adaptation of generative robot policies in latent space. *arXiv preprint arXiv:2607.08877*, 2026.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4996–5013. PMLR, 2025.

Iman Nematollahi, Branton DeMoss, Akshay L. Chandra, Nick Hawes, Wolfram Burgard, and Ingmar Posner. Lumos: Language-conditioned imitation learning with world models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Physical Intelligence. $\pi^*_{0.6}$: a VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

Han Qi, Haocheng Yin, Aris Zhu, Yilun Du, and Heng Yang. Inference-time enhancement of generative robot policies via predictive world modeling. *IEEE Robotics and Automation Letters*, 2025.

Guruprasad Raghavan, Bahey Tharwat, Surya Narayanan Hari, Dhruvil Satani, Rex Liu, and Matt Thomson. Engineering flexible machine learning systems by traversing functionally invariant paths. *Nature Machine Intelligence*, 6:1179–1196, 2024. doi: 10.1038/s42256-024-00902-x.

Allen Ren, Justin Lidard, Lars Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations*, volume 2025, pp. 77288–77329, 2025.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019.

Ralf Römer, Yi Zhang, Yuming Li, and Angela P Schoellig. Clare: Continual learning for vision-language-action models via autonomous adapter routing and expansion. *IEEE Robotics and Automation Letters*, 2026.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Ansh Kumar Sharma, Yixiang Sun, Ninghao Lu, Yunzhe Zhang, Jiarao Liu, and Sherry Yang. World-gymnast: Training robots with reinforcement learning in a world model. *arXiv preprint arXiv:2602.02454*, 2026.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. RL's razor: Why online reinforcement learning forgets less. In *International Conference on Learning Representations*, 2026.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

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

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In *Conference on Robot Learning*. PMLR, 2025.

Weikang Wan, Yifeng Zhu, Rutav Shah, and Yuke Zhu. LOTUS: Continual imitation learning for robot manipulation through unsupervised skill discovery. In *IEEE International Conference on Robotics and Automation*, 2024.

Junjin Xiao, Yandan Yang, Xinyuan Chang, Ronghan Chen, Feng Xiong, Mu Xu, Wei-Shi Zheng, and Qing Zhang. World-env: Leveraging world model as a virtual environment for vla post-training. *arXiv preprint arXiv:2509.24948*, 2025.

Raymond Yu, William Huey, Mustafa Mukadam, Anusha Nagabandi, and Abhishek Gupta. Support-constrained RL enables real-world policy improvement without real-world experience. *arXiv preprint arXiv:2606.27475*, 2026.

William Yue, Bo Liu, and Peter Stone. t-DGR: A trajectory-based deep generative replay method for continual learning in decision making. *arXiv preprint arXiv:2401.02576*, 2024.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, 2017.

Jiarun Zhu, Yijun Hong, Xiaoquan Sun, Zetian Xu, Qijun He, Haijier Chen, Zhiyong Wang, Mingqi Yuan, Wenjun Zeng, and Jiayu Chen. Can vision-language-action models learn from real-world data continually without forgetting? *arXiv preprint arXiv:2605.26820*, 2026.

APPENDIX

S.1 Training Objectives **15**
S.1.1 Reward in Imagination . . . 15
S.1.2 Policy Optimization with PPO . . . 15
S.1.3 Policy Optimization with SAC . . . 15
S.1.4 Behavior-Cloning Loss . . . 16
S.1.5 Rehearsal Loss . . . 16
S.2 Algorithm **16**
S.3 Implementation Details **17**
S.3.1 World Models . . . 17
S.3.2 Base Policies . . . 17
S.3.3 Rewards . . . 17
S.3.4 Steering Module and Critics . . . 17
S.3.5 Hyperparameters . . . 19
S.3.6 Compute and Wall-Clock . . . 20
S.4 Experimental Setup Details **20**
S.4.1 Environments and Task Sequences . . . 20
S.4.2 Real-World Setup . . . 20
S.4.3 Action Space . . . 21
S.4.4 Rehearsal Seed Frames and Storage Cost . . . 21
S.4.5 Baselines . . . 22
S.4.6 Metrics . . . 23
S.5 Additional Results and Ablations **24**
S.5.1 Full Sequential Results . . . 24
S.5.2 Full Per-Task Results . . . 24
S.5.3 Initialization of the Steering Module . . . 24
S.5.4 Task Order Sensitivity . . . 25
S.5.5 Retention Coefficient λ . . . 26
S.5.6 Behavior Cloning Coefficient α . . . 26
S.5.7 World Model Faithfulness Check . . . 27
S.5.8 Qualitative Imagined Rehearsal Rollouts . . . 27

S.1 Training Objectives

This section states all objectives used by D2R, in the order they appear in Eq. 5. Throughout, s denotes the state representation conditioned on by the steering module, ℓ an instruction, and $\boldsymbol{w} \in \mathcal{W} = \mathbb{R}^{H \cdot d}$ the input noise for an H-step action chunk in a d-dimensional action space.

S.1.1 Reward in Imagination

Training inside the world model requires a reward that can be evaluated on imagined latent states, without querying the environment. We therefore reuse the latent success classifier of DiWA (Chandra et al., 2025), a small network on top of the frozen world model encoder that predicts whether a latent state satisfies the task. One classifier is trained per task on the demonstrations of that task, with states after task completion as positives and a sample of the remaining states as negatives, and all classifiers stay frozen during adaptation.

At every step of an imagined rollout on task ℓ_k, the reward is the predicted class of that task's classifier,

$$r_t = \mathbb{1}\big[\, C_{\psi_k}(s_t) = 1 \,\big] \in \{0, 1\}, \tag{S.1}$$

so the return in Eq. S.2 counts the imagined steps in which the task is predicted solved. A rollout also ends early once the classifier of the task being rolled out exceeds a confidence of 0.90, which keeps rollouts short and prevents the module from collecting reward past completion. The same mechanism terminates rehearsal rollouts, using the classifier of the retained task rather than the current one. Rehearsal itself uses no reward, since Eq. S.7 only compares the module to its reference on the visited states. On LIBERO no such classifier is used. The task is learned in the environment, where the reward is the benchmark's own success signal, consumed by Eq. S.5 in place of Eq. S.2. We do not train a reward model in A2World's latent space, since the model is not accurate enough to optimize a policy inside it (§4.4), so rehearsal rollouts there run to a fixed length rather than stopping at a predicted success.

S.1.2 Policy Optimization with PPO

When the world model is faithful enough to act as a simulator (§3.4), the current task is learned inside it. Imagined rollouts cost only forward passes through the frozen policy and world model and run in parallel, so on-policy data is cheap and we optimize the module with PPO (Schulman et al., 2017). This is the regime used on CALVIN and on the real robot. Each iteration collects a batch of imagined rollouts with the current module, giving $\mathcal{D}_{\text{cur}}$, with rewards from Eq. S.1 normalized by a running estimate of their scale. On these rollouts we use generalized advantage estimation with the value function V_ν,

$$\hat{A}_t = \sum_{l \geq 0} (\gamma \lambda_{\text{GAE}})^l \, \delta_{t+l}, \qquad \delta_t = r_t + \gamma V_\nu(s_{t+1}) - V_\nu(s_t), \qquad \hat{R}_t = \hat{A}_t + V_\nu(s_t), \tag{S.2}$$

where $\hat{R}_t$ is the return target for the value loss. Both the current module and the one that collected the rollouts are diagonal Gaussians over $\mathcal{W}$, so the importance ratio $\rho = \pi_k^{\mathcal{W}}(\boldsymbol{w} \mid s, \ell_k) / \pi_{\text{old}}^{\mathcal{W}}(\boldsymbol{w} \mid s, \ell_k)$ is available in closed form and

$$\mathcal{L}_{\text{PPO}} = -\mathbb{E}_{(s, \boldsymbol{w}, \hat{A}) \sim \mathcal{D}_{\text{cur}}} \Big[\min \big(\rho \hat{A},\ \text{clip}(\rho, 1 - \epsilon, 1 + \epsilon)\, \hat{A} \big) \Big]. \tag{S.3}$$

The value function is fit on the same batch,

$$\mathcal{L}_V = \tfrac{1}{2}\, \mathbb{E}_{(s, \hat{R}) \sim \mathcal{D}_{\text{cur}}} \big[(V_\nu(s) - \hat{R})^2 \big], \tag{S.4}$$

and is not part of the module's objective in Eq. 5. Each batch is reused for several epochs, stopping early once the KL between the updated module and the one that collected the rollouts exceeds a threshold. Hyperparameters are listed in Appendix S.3.5.

S.1.3 Policy Optimization with SAC

When the world model is not accurate enough to optimize a policy inside it (§3.4), the current task is learned in the environment instead. Every transition then costs a real episode, so the data is scarce

and has to be reused, and we train the module off-policy with SAC over the noise space, following DSRL (Wagenmaker et al., 2025). This is the regime used on LIBERO. Rehearsal is unaffected by the change, since the KL term of Eq. S.7 is still computed on rollouts imagined in the world model, so the robot is used only for the current task.

In this regime the module is a tanh-squashed Gaussian over $\mathcal{W}$, which keeps the sampled noise bounded without the clipping used in imagination. A pair of critics Q_{ν_1}, Q_{ν_2} on state and noise is fit by temporal-difference learning on $\mathcal{D}_{\text{cur}}$, which here is the replay buffer of transitions collected so far on the current task, and the actor maximizes the entropy-regularized value of its noise,

$$\begin{aligned} \mathcal{L}_{\text{SAC}} &= \mathbb{E}_{s\sim\mathcal{D}_{\text{cur}},\ \boldsymbol{w}\sim\pi_k^{\mathcal{W}}(\cdot|s,\ell_k)}\big[\beta \log \pi_k^{\mathcal{W}}(\boldsymbol{w} \mid s, \ell_k) - \min_i Q_{\nu_i}(s, \boldsymbol{w})\big], \\ \mathcal{L}_Q &= \sum_i \mathbb{E}\big[(Q_{\nu_i}(s, \boldsymbol{w}) - r - \gamma \bar{V}(s'))^2\big], \\ \bar{V}(s') &= \mathbb{E}_{\boldsymbol{w}'\sim\pi_k^{\mathcal{W}}}\big[\min_i \bar{Q}_{\nu_i}(s', \boldsymbol{w}') - \beta \log \pi_k^{\mathcal{W}}(\boldsymbol{w}' \mid s', \ell_k)\big], \end{aligned} \tag{S.5}$$

where the target critics $\bar{Q}_{\nu_i}$ are updated by Polyak averaging. The temperature β is tuned automatically against a target entropy of $-|\mathcal{W}|/2$ rather than held fixed. Hyperparameters are listed in Appendix S.3.5.

S.1.4 Behavior-Cloning Loss

Optimizing against a learned model invites the module to find noise that the model rewards but the robot does not. Following DiWA (Chandra et al., 2025), we therefore anchor the steered actions to the ones the frozen policy produces on its own. The anchor costs nothing to obtain, since drawing noise from the prior recovers the unsteered policy, so no teacher network has to be stored alongside the module. With $\mu_k(s, \ell_k)$ the mean of the steering module,

$$\mathcal{L}_{\text{BC}} = \mathbb{E}_{s\sim\mathcal{D}_{\text{cur}},\ \boldsymbol{w}_0\sim\mathcal{N}(0,I)}\Big[\big\|\pi_{\text{dp}}^{\mathcal{W}}\big(s, \ell_k, \mu_k(s, \ell_k)\big) - \pi_{\text{dp}}^{\mathcal{W}}(s, \ell_k, \boldsymbol{w}_0)\big\|_2^2\Big], \tag{S.6}$$

computed over the executed steps of the chunk and averaged over the current-task states. The term is only needed where the reward comes from the world model. When the task is learned in the environment the reward is real, so there is nothing to exploit and $\alpha = 0$ in Eq. 5. Its value in imagination is listed in Table S.1, and §S.5.6 reports how performance depends on it.

S.1.5 Rehearsal Loss

Rehearsal is what prevents earlier skills from drifting while a new task is learned. It is applied the same way in both regimes, since the retained tasks are always rolled out inside the frozen world model and the robot is never revisited for a task that is not being learned.

For each retained task $\ell_j \in \mathcal{R}_k$, the states $\mathcal{D}_{\text{reh}}^j$ come from rollouts seeded at a stored initial observation of ℓ_j and unrolled with the reference $\pi_{\text{ref}(j)}^{\mathcal{W}}$ choosing the noise, so they follow the behavior that is to be preserved. The reference is the previous module $\pi_{k-1}^{\mathcal{W}}$ for tasks that were already steered and the prior $\mathcal{N}(0, I)$ for the rest. On CALVIN these rollouts are regenerated at every iteration rather than stored, and since the world model is frozen their distribution does not drift as the sequence grows. The penalty is Eq. 4 in the main text, averaged over the retained tasks with a single coefficient λ. Both distributions are diagonal Gaussians, so the divergence of the current module from the reference is available in closed form,

$$\text{KL}\big(\mathcal{N}(\mu, \sigma^2)\,\|\,\mathcal{N}(\mu', \sigma'^2)\big) = \tfrac{1}{2}\sum_i \Big[\frac{\sigma_i^2 + (\mu_i - \mu_i')^2}{\sigma_i'^2} - 1 + \log \sigma_i'^2 - \log \sigma_i^2\Big], \tag{S.7}$$

with $\mu' = 0$ and $\sigma' = 1$ when the reference is the prior. The value of λ is listed in Table S.1, and §S.5.5 reports how performance depends on it.

S.2 Algorithm

Algorithm S.1 summarizes one stage of D2R. Each iteration collects experience on the current task, regenerates rehearsal rollouts for the retained tasks inside the world model, and updates the module

on the combined objective of Eq. 5, with the choice of MDP and reinforcement learning algorithm following §3.4.

S.3 Implementation Details

This section describes the components D2R builds on and the one it trains. The world models, base policies, and reward classifiers stay frozen throughout, and only the steering module and its critic are learned.

S.3.1 World Models

On CALVIN and on the real robot we use the recurrent state-space model of DiWA (Chandra et al., 2025) and LUMOS (Nematollahi et al., 2025) without modification, trained once on task-agnostic play data and frozen thereafter. It encodes a static and a gripper camera view into a latent state and predicts the next latent from an action chunk, and we refer to those works for the architecture and the training objective. The CALVIN model is trained on play data from environment D, and the real-robot model on six hours of teleoperated play collected on the same Franka, with the same architecture and settings. On LIBERO we use A2World (Huang et al., 2026) off the shelf, pretrained on 2.1M real-robot trajectories and fine-tuned on the four suites, so that neither it nor π_0 has seen the LIBERO-90 target tasks. It is likewise frozen.

The two regimes use these models differently. On CALVIN and the real robot the model supplies both the rollouts the module learns in and the latent states it conditions on. On LIBERO it supplies only rehearsal rollouts, and the module never reads an A2World latent, conditioning instead on the raw observation (Table S.1).

S.3.2 Base Policies

On CALVIN and on the real robot the base policy is a single language-conditioned diffusion policy trained by behavior cloning, following DiWA (Chandra et al., 2025). It differs from DiWA in covering all tasks at once rather than one policy per skill, and it conditions on the same world-model latents the module later sees in imagination, with the instruction embedding fused into those features by FiLM conditioning (Perez et al., 2018). The CALVIN policy is trained on 150 demonstrations for each of the eight tasks and the real-robot policy on 100 demonstrations for each of the seven tabletop tasks. On LIBERO the base policy is the publicly released π_0 checkpoint (Black et al., 2024) fine-tuned on the Spatial, Object, Goal, and Long suites, which we use as is.

What the steering module controls in all three cases is the initial noise the sampler starts from, so the sampler itself is left untouched, DDIM on CALVIN and the real robot and the flow ODE of π_0 on LIBERO, with step counts and embedding dimensions in Table S.1. All base policies remain frozen throughout.

S.3.3 Rewards

The latent success classifier is two small multilayer perceptrons on the frozen 2048-dimensional world-model latent, an encoder to a 256-dimensional embedding and a two-way classification head, both with a hidden width of 1024. It is trained with the contrastive and cross-entropy objective of DiWA (Chandra et al., 2025), to which we refer for the objective itself and for the comparison against a vision-based classifier. We reuse the classifiers released with DiWA rather than retraining them. On the real robot one classifier is trained per task on the same demonstrations as the base policy. No classifier is trained for A2World, for the reason given in §S.1.1.

S.3.4 Steering Module and Critics

The steering module is a three-layer perceptron of width 512 with layer normalization that outputs a diagonal Gaussian over $\mathcal{W}$, with a state-dependent standard deviation bounded to the range in Table S.1. What it conditions on differs by benchmark. On CALVIN and the real robot it reads the world-model latent, while on LIBERO it encodes the raw observation with its own encoder, and in

Algorithm S.1 D2R: Continual Steering of a Frozen Generative Policy with World-Model Rehearsal

Input: frozen policy $\pi^{\mathcal{W}}_{\text{dp}}$, frozen world model P_ϕ, task rewards R_j, task sequence $\ell_1, \dots, \ell_N$, initial observations $\mathcal{O}_j$ from the demonstrations of each task j.
Set the prior $\pi^{\mathcal{W}}_{\text{prior}} \leftarrow \mathcal{N}(0, I)$, the unsteered policy's noise distribution.
for task $k = 1, \dots, N$ in sequence **do**
  Freeze the previous module $\pi^{\mathcal{W}}_{k-1}$.
  Initialize a new steering module $\pi^{\mathcal{W}}_{k}$ from scratch.
  Initialize the critic of the chosen RL algorithm (V_ν for PPO, Q_ν for SAC).
  Choose the MDP for task k, $\mathcal{M}_{\text{wm}}$ if the world model covers the task and $\mathcal{M}_{\text{env}}$ otherwise.
  Define the retention set $\mathcal{R}_k$, the tasks to be rehearsed while learning task k.
  For each $j \in \mathcal{R}_k$, set its reference $\text{ref}(j)$ to $\pi^{\mathcal{W}}_{k-1}$ if task j was steered before, else to $\pi^{\mathcal{W}}_{\text{prior}}$.
  for iteration $= 1, 2, \dots$ **do**
    Initialize buffers $\mathcal{D}_{\text{cur}}$ and $\mathcal{D}_{\text{reh}}$ ($\mathcal{D}_{\text{cur}}$ persists for off-policy algorithms).

    1. Adaptation
    for episode $= 1, \dots, N_{\text{cur}}$ in parallel **do**
      Sample initial observation $o_0 \sim \mathcal{O}_k$ and encode it to s_0.
      for step $t = 0, \dots, T-1$ **do**
        Sample noise $\boldsymbol{w}_t \sim \pi^{\mathcal{W}}_{k}(\cdot \mid s_t, \ell_k)$ (Eq. 3).
        Decode action chunk $\boldsymbol{a}_t \leftarrow \pi^{\mathcal{W}}_{\text{dp}}(s_t, \ell_k, \boldsymbol{w}_t)$.
        if task k is learned in $\mathcal{M}_{\text{wm}}$ **then**
          Step the world model: $s_{t+1} \sim P_\phi(\cdot \mid s_t, \boldsymbol{a}_t)$.
        else
          Step the environment with $\boldsymbol{a}_t$, observe o_{t+1}, and encode it to s_{t+1}.
        end if
        Observe reward $r_t = R_k(s_{t+1})$.
        Add $(s_t, \boldsymbol{w}_t, r_t)$ to $\mathcal{D}_{\text{cur}}$.
      end for
    end for

    2. Rehearsal
    for rehearsed task $j \in \mathcal{R}_k$, episode $= 1, \dots, N_{\text{reh}}$ in parallel **do**
      Sample initial observation $o_0 \sim \mathcal{O}_j$ and encode it to s_0.
      for step $t = 0, \dots, T-1$ **do**
        Set noise to the reference mean: $\boldsymbol{w}_t \leftarrow \text{mean}\big[\pi^{\mathcal{W}}_{\text{ref}(j)}(\cdot \mid s_t, \ell_j)\big]$.
        Decode action chunk $\boldsymbol{a}_t \leftarrow \pi^{\mathcal{W}}_{\text{dp}}(s_t, \ell_j, \boldsymbol{w}_t)$.
        Step the world model: $s_{t+1} \sim P_\phi(\cdot \mid s_t, \boldsymbol{a}_t)$.
        Add (s_t, j) to $\mathcal{D}_{\text{reh}}$.
        Terminate on success if a detector exists, otherwise at step T.
      end for
    end for

    3. Update
    For on-policy algorithms, compute GAE advantages and returns on $\mathcal{D}_{\text{cur}}$ (Eq. S.2).
    for update $= 1, \dots, U$ **do**
      for minibatch $= 1, \dots, B$ **do**
        Sample a current-task minibatch from $\mathcal{D}_{\text{cur}}$ and rehearsal states (s, j) from $\mathcal{D}_{\text{reh}}$.
        Compute the RL loss $\mathcal{L}_{\text{RL}}$, e.g. PPO (Eq. S.3) or SAC (Eq. S.5).
        Compute the rehearsal loss $\mathcal{L}_{\text{KL}}$ on $\mathcal{D}_{\text{reh}}$ against $\pi^{\mathcal{W}}_{\text{ref}(j)}$ (Eq. 4).
        Compute the BC loss $\mathcal{L}_{\text{BC}}$ if task k is learned in $\mathcal{M}_{\text{wm}}$, else 0 (Eq. S.6).
        Update $\pi^{\mathcal{W}}_{k}$ on $\mathcal{L}_{\text{RL}} + \alpha\mathcal{L}_{\text{BC}} + \lambda\mathcal{L}_{\text{KL}}$ (Eq. 5).
        Update the critic on its loss (Eq. S.4 or Eq. S.5).
      end for
    end for
  end for
end for
return $\pi^{\mathcal{W}}_{N}$, deployed on every task. Disabling it recovers π_{dp}.

both cases the instruction embedding is fused into those features by FiLM conditioning, as in the base policy. Samples are clipped in imagination and tanh-squashed in the environment (§S.1.3).

The critic follows the algorithm of the regime, a single value network in imagination and a pair of Q networks in the environment, with the widths listed in Table S.1. Both the module and its critic are initialized from scratch at every stage, and the critic is discarded once the stage ends, so the module is the only component carried forward.

S.3.5 HYPERPARAMETERS

Table S.1 lists the hyperparameters of D2R in both regimes. The frozen components keep the settings of the works they come from. On CALVIN the module emits one value per action dimension and executed step, so $|\mathcal{W}| = H \cdot d$. On LIBERO it emits a single d-dimensional vector that the sampler applies across the whole chunk.

Table S.1: Hyperparameters of D2R. CALVIN adapts in imagination with PPO, LIBERO adapts in the environment with SAC. Rehearsal runs in the world model in both cases. The real robot follows the CALVIN column.

Parameter	**Symbol**	**CALVIN**	**LIBERO**
Frozen policy and noise space			
Action chunk length	H	4	50
Executed actions	—	4	20
Action dimension	d	7	32
Noise dimension	$\|\mathcal{W}\|$	28	32
Sampler	—	DDIM	flow ODE
Sampling steps	—	10 of 20	10 of 10
Steering module			
State input	s	world-model feature (2048)	input image (64x64)
Instruction input	ℓ	CLIP embedding (512)	PaliGemma embedding (2048)
Hidden layers	—	[512, 512, 512], LayerNorm	[512, 512, 512]
Std bounds	$[\sigma_{\min}, \sigma_{\max}]$	[0.1, 1.5]	[0, 1]
Sample clipping	c	[-3, 3]	[-1, 1]
Initialization per task	—	from scratch	from scratch
Adaptation			
Algorithm	—	PPO	SAC
Iterations / env. steps	—	800 iterations	250k grad. steps
Parallel rollouts	—	50	1
Rollout length (chunks)	T	36	—
Discount	γ	0.999	0.999
GAE parameter	λ_{GAE}	0.95	—
PPO clip	ϵ	0.01	—
Update epochs per iteration	—	10	—
Batch size	—	7500	256
Actor / critic learning rate	—	10^{-4} / 10^{-3}	10^{-4} / 3×10^{-4}
Critic hidden layers	—	[256, 256, 256]	[512, 512, 512]
Entropy temperature	β	—	learned, $\beta_0 = 1$
Update-to-data ratio	—	—	20
Rehearsal			
Seed frames per task	—	150 demo initial frames	50 demo initial frames
Parallel rollouts per task	—	10	1
Rollout length (chunks)	T	36	200
Early termination	—	classifier ≥ 0.9	none
Loss weights (Eq. 5)			
Behavior cloning	α	5×10^{-2}	0
Retention	λ	5×10^{-3}	5×10^{-3}

S.3.6 Compute and Wall-Clock

All adaptation runs use a single A100 GPU, except for the rehearsal rollouts on LIBERO, which are generated across 5 GPUs before each stage. One training stage takes about an hour on CALVIN, eight hours on LIBERO, and 30 minutes on the real robot.

Rehearsal is the only cost D2R adds over steering without it. On CALVIN, where the world model and the base policy are both small, the retained-task rollouts are regenerated at every iteration, which lengthens a stage by 25% relative to DSRL under the same budget. On LIBERO they are instead generated once at the start of a stage, since every imagined step queries both A2World and π_0 and placing that inside the update loop would dominate the wall clock. Generating them takes 30 minutes for 50 episodes per retained task, which is amortized over the gradient steps that follow. Since the world model, the base policy, and the reward classifiers are frozen and reused across the sequence, the figures above exclude their training.

S.4 Experimental Setup Details

S.4.1 Environments and Task Sequences

Table S.2 lists the tasks and their instructions for the CALVIN and LIBERO benchmarks. On the Franka robot we use seven tabletop tasks, open drawer, close drawer, move slider left, move slider right, turn on LED, pick the block from the table and place it in the drawer, and pick the block from the drawer and place it on the table, all shown in Figure S.2. Five of them also appear in CALVIN, while the two block tasks do not. The three adapted tasks are open drawer, move slider left, and turn on LED, learned in this order.

S.4.2 Real-World Setup

Experiments on hardware use a 7-DoF Franka Emika Panda in the tabletop environment of Figure S.1, which mirrors the CALVIN scene with a cabinet, a drawer, a sliding door, an LED button, and a movable block. The policy observes the scene through a static camera placed over the shoulder of the robot and a camera mounted on the gripper, both at 64×64 resolution, and acts through the same 7-dimensional end-effector command as in simulation (§S.4.3). All play data and demonstrations are teleoperated through a VR controller.

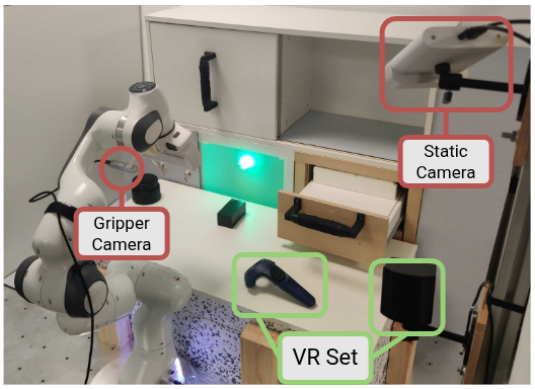


Figure S.1: **Real-world setup.** A Franka Emika Panda in a tabletop environment with a cabinet, a drawer, a sliding door, and an LED button. The policy observes the scene through a static camera and a gripper-mounted camera. Play data and demonstrations are teleoperated with a VR controller.

Three components are learned once on this platform and then frozen. The world model is an RSSM trained on six hours of teleoperated play data. The base policy is a single language-conditioned diffusion policy trained on 100 demonstrations for each of the seven tasks, and each task additionally has a success classifier trained on the same demonstrations. D2R and DSRL are then trained entirely in imagination, so the robot is used only for evaluation. After each stage we run 20 episodes per

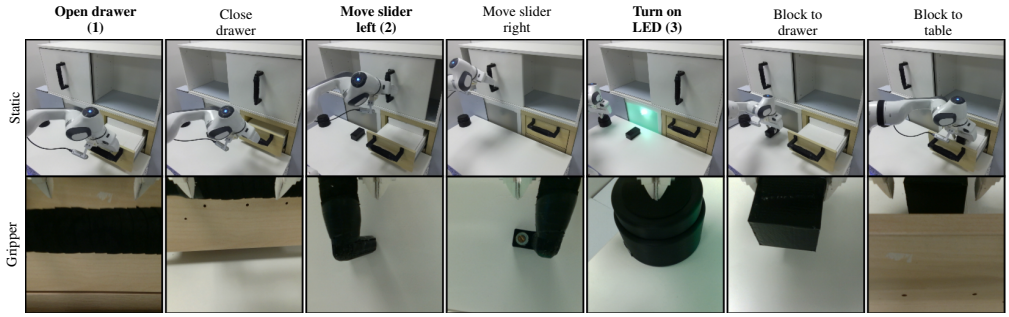


Figure S.2: **The seven real-robot tasks.** Each column is one task, seen by the static camera (top) and the gripper camera (bottom). The three tasks adapted in sequence are set in bold with their stage number.

adapted task of at most 72 steps, starting from 20 fixed initial configurations that are reset by hand between episodes.

S.4.3 Action Space

All experiments, in simulation and on the real robot, use the same 7-dimensional relative action space,

$$\mathbf{a} = [\delta x, \delta y, \delta z, \delta\phi, \delta\theta, \delta\psi, a_{\text{gripper}}]. \tag{S.8}$$

The first three entries are changes in end-effector position and the next three are changes in its orientation, expressed as Euler angles, each taking continuous values in $[-1, 1]$. The last entry commands the gripper. On LIBERO, π_0 emits a 32-dimensional action vector, of which only the first seven dimensions are sent to the robot and the rest are discarded. The steering module still acts on the full 32-dimensional noise vector of the sampler, which Table S.1 reports as the noise dimension.

S.4.4 Rehearsal Seed Frames and Storage Cost

Rehearsal needs a starting point for every retained task, and these starting points are the only thing D2R keeps from a task once its stage is over. For each task we store one frame per demonstration, taken from the first step of that demonstration, extracted once before the sequence begins and never updated afterwards. On CALVIN a frame is kept as its 2048-dimensional world-model feature, and on LIBERO as the initial observation of the benchmark episode. The demonstrations are drawn at random under a fixed seed. Rollouts are then generated from these frames inside the frozen world model as described in §S.1.5, so no trajectory of a retained task is ever stored or revisited.

The storage column of Table 2 compares this against a replay buffer that keeps the retained-task trajectories themselves, which is what the replay KL baseline requires. Since we keep one frame per demonstration, the two hold the same episodes and differ only in how much of each episode they keep, so

$$\text{replay frames per task} \;=\; (\text{seed frames per task}) \times \bar{T}, \tag{S.9}$$

where $\bar{T}$ is the mean length of those demonstrations, 60.2 frames on CALVIN and 148.7 frames on the LIBERO-90 tasks that make up the sequence. After the full eight-task sequence the two hold

Kept after the sequence	**CALVIN**	**LIBERO**
Seed frames per task	150	50
Seed frames, eight tasks	1,200	400
Mean episode length $\bar{T}$	60.2	148.7
Replay frames per task	9,030	7,435
Replay frames, eight tasks	72,240	59,480

and Table 2 reports the LIBERO figures, about 400 frames against about 59,000. The gap is exactly the trajectory dimension, since the ratio on each benchmark is its mean episode length, 60

on CALVIN and 149 on LIBERO. What D2R keeps grows only with the number of retained tasks, whereas a replay buffer grows with the length of every stored episode as well, so the ratio widens as tasks get longer.

S.4.5 BASELINES

The baselines are introduced in §4.1. This section gives only what is held fixed across them and the settings in which they differ. All of them start from the same frozen base policy and use the same world model, reward classifiers, seed frames, task order, and evaluation protocol, and those that train do so under the same per-stage budget. Any hyperparameter not named below is the one D2R uses in Table S.1.

DiWA. The published method (Chandra et al., 2025), which adapts by fine-tuning all policy weights in imagination with PPO rather than by steering the noise. Since every method starts from the same base policy, it fine-tunes the multi-task policy of §S.3.2 rather than the per-skill policies of the released setup.

DiWA + rehearsal. DiWA with the world-model rehearsal of D2R added, run in single-task adaptation on CALVIN only. The policy weights θ are fine-tuned on the adapted task while the other seven are rehearsed on the same imagined rollouts D2R uses. Since the fine-tuned policy has no tractable action distribution, the KL penalty of Eq. 4 is replaced by an action-level anchor to the frozen base policy on the rehearsal states,

$$\mathcal{L}_{\text{reh}}^{\text{DiWA}} = \sum_{j \neq k} \mathbb{E}_{s \sim \mathcal{D}_{\text{reh}}^{j},\, \boldsymbol{w} \sim \mathcal{N}(0,I)} \left[\left\| \pi_\theta(s, \ell_j, \boldsymbol{w}) - \pi_{\text{dp}}^{\mathcal{W}}(s, \ell_j, \boldsymbol{w}) \right\|_2^2 \right], \tag{S.10}$$

with the same noise fed to both policies, weighted by the same coefficient λ as D2R. It is not run in the sequence experiments, where LoRA fine-tuning is the weight-space method, because fine-tuning all weights at every stage is the setting DiWA itself does not target.

LoRA fine-tuning. On CALVIN, DiWA's PPO fine-tuning in imagination with the policy weights frozen and LoRA adapters (Hu et al., 2022) of rank 64 with scaling factor 64, no dropout, injected into every linear layer of the policy's denoising network. Training follows the schedule of D2R apart from the actor learning rate, 10^{-5}. On LIBERO, adapters of rank 8 with scaling factor 8 are applied only to the action expert of π_0, across its 18 transformer layers, with the vision-language backbone frozen, and trained off policy in the environment. We tried both SAC and an IQL-style objective and report the latter, which performed better, since π_0 exposes no exact action log-probability and the entropy term of SAC has no closed form. It uses twin Q networks, a value network with expectile 0.7, and an advantage-weighted flow-matching actor update with behavior-cloning coefficient 0.1 and an anchor to the initial adapters.

DP-TTC. Test-time selection on CALVIN, which trains nothing and instead queries the world model and the success classifier at every decision. It draws 25 candidate chunks at temperature 1, imagines each one chunk ahead, and executes one drawn uniformly from the 10 best by the classifier's predicted change in success, without refining the candidates.

DSRL. The steering approach of Wagenmaker et al. (2025) without rehearsal, so the module of §S.3.4 is trained on the current task alone and Eq. 5 is used with $\lambda = 0$. It follows the regime of D2R on each benchmark, PPO in the world model on CALVIN and SAC in the environment on LIBERO, which departs from the released setup where steering is always learned from environment interaction.

The three remaining input-space baselines add the rehearsal penalty of Eq. 4 to DSRL with the same coefficient λ as D2R and differ only in the states it is applied to, so each is described by that set alone.

DSRL + instruction KL. The penalty is applied to observations of the current task paired with instructions of the retained tasks, so no state of a retained task is ever stored or generated. On LIBERO the mismatched instruction is drawn from the four suites π_0 was fine-tuned on, one per observation.

DSRL + first-obs KL. The penalty is applied to the stored seed frames of §S.4.4 directly, without rolling them forward in the world model, so it sees only states in which a retained task has not yet begun.

DSRL + replay KL. The penalty is applied to rollouts of the retained tasks collected once at the start of each stage with the previous module and stored for the rest of it, real trajectories on LIBERO and imagined ones on CALVIN. It differs from D2R on CALVIN only in that the rollouts are stale rather than regenerated at every iteration.

The last two rows of Table 2 are reference settings rather than baselines. Both use the steering module and the objective of D2R, and both remove the sequence in a different way, so together they bracket what sequential steering can reach.

Per-task modules. One module is trained for each task and selected at evaluation by the task identity. Each module therefore sees only its own task, nothing has to be retained, and Eq. 5 is used with $\lambda = 0$. This is the ceiling for steering at the same module capacity, paid for by storing one module per task and by requiring the task identity at test time, neither of which D2R needs.

Multi-task module. A single module is trained on all eight tasks at once rather than in sequence, with a larger budget than any single stage of D2R, namely longer training, more collected experience, and more parallel environments, at the same module capacity. Every task is present throughout, so again nothing has to be retained and $\lambda = 0$. Retention and forgetting are undefined here because no task has a stage at which it was learned, which is why those cells are marked in Table 2.

S.4.6 METRICS

Tasks arrive in the order $\ell_1, \ldots, \ell_N$. After the stage on task ℓ_k we evaluate the resulting system on every task and record its success rate $S_{k,j}$ on task ℓ_j, in percent. This gives a success matrix S whose rows are the stages $0, \ldots, N$ and whose columns are the evaluated tasks, where row 0 holds the success $S_{0,j}$ of the frozen base policy. The three metrics of Table 2 are functions of this matrix.

Final average success. The success of the system obtained after the last stage, averaged over the sequence, is the mean of the last row,

$$\text{Final avg.} \;=\; \frac{1}{N}\sum_{j=1}^{N} S_{N,j}, \tag{S.11}$$

reported as Success in Table 2. It coincides with the AVG metric of Hu et al. (2026). For the base policy the last row equals $S_{0,\cdot}$, so its entry is the base average.

Forgetting. Forgetting measures how much success on a task drops after later tasks are learned. Since each task is learned once, we measure the drop relative to the success right after the task's own stage and average over the tasks learned before the last stage,

$$\text{Forgetting} \;=\; \frac{1}{N-1}\sum_{j=1}^{N-1} \left(S_{j,j} - S_{N,j}\right), \tag{S.12}$$

which is the negative backward transfer (NBT) of Liu et al. (2023) and Hu et al. (2026), and the negative of backward transfer (Lopez-Paz & Ranzato, 2017). More generally, $S_{j,j} - S_{k,j}$ is the forgetting score on task ℓ_j after stage k. Positive values are losses, and zero means no forgetting on average. The last task is excluded because nothing is learned after it. Task-specific modules never change after their own stage, so their forgetting is zero by construction.

Retention. Retention measures the change in success relative to the frozen base policy, so it is defined for tasks that were never a stage of the sequence as well. After stage k the retention on task ℓ_j is $S_{k,j} - S_{0,j}$. Table 2 reports its mean after the last stage over the retention set $\mathcal{R}$,

$$\text{Retention} \;=\; \frac{1}{|\mathcal{R}|}\sum_{j\in\mathcal{R}} \left(S_{N,j} - S_{0,j}\right), \tag{S.13}$$

where $\mathcal{R}$ is the eight tasks of the sequence on CALVIN and the four suites on LIBERO that π_0 was fine-tuned on and that never appear in the sequence. On CALVIN this equals Success minus the base average. On LIBERO it is the change in the held-out success ZS of Hu et al. (2026) relative to π_0. In single-task adaptation (Table 1) there is no sequence. Writing $S_j^{(i)}$ for the success on task ℓ_j after adapting to task ℓ_i alone, the retention reported for task ℓ_i is the mean change over all tasks,

$$\text{Retention}^{(i)} \;=\; \frac{1}{N}\sum_{j=1}^{N} \left(S_j^{(i)} - S_{0,j}\right), \tag{S.14}$$

including the adapted task itself, so that gains on ℓ_i and losses on the other tasks enter with their sign. Positive values mean the adapted system ends above the base policy, negative values mean it ends below.

S.5 Additional Results and Ablations

S.5.1 Full Sequential Results

Table S.3 repeats Table 2 with two further columns per benchmark. *Single* is the average success when each task is adapted on its own, outside the sequence, and *New* is the mean of the diagonal $\frac{1}{N}\sum_j S_{j,j}$, the success on each task immediately after its own adaptation stage. The difference between Single and New is the cost of learning a task inside the sequence rather than in isolation, and the difference between New and Success is $(N-1)/N$ times the forgetting of Eq. S.12, which averages the same per-task drops but excludes the last task. A module trained on a single task has nothing to retain, so the Single column of DSRL is the per-task modules setting of Table 2.

Table S.3: Continual steering over eight tasks on CALVIN (in imagination) and LIBERO (in the environment), with the Single and New columns that Table 2 omits.

	CALVIN					LIBERO				
Method	Single↑	New↑	Success↑	Retention↑	Forgetting↓	Single↑	New↑	Success↑	Retention↑	Forgetting↓
Base	63.7 ± 6.5	63.7 ± 6.5	63.7 ± 6.5	0.0	0.0	10.4 ± 2.7	10.4 ± 2.7	10.4 ± 2.7	0.0	0.0
LoRA	67.1 ± 4.9	58.2 ± 5.9	11.7 ± 1.9	-52.0	53.2	59.2 ± 4.5	54.7 ± 1.6	1.0 ± 1.6	-41.6	61.4
DSRL	$\mathbf{81.8 \pm 2.9}$	65.0 ± 5.6	63.6 ± 4.8	-0.1	**1.6**	$\mathbf{91.3 \pm 1.0}$	$\mathbf{85.4 \pm 0.0}$	16.9 ± 5.2	-30.8	78.3
+ instruction KL	79.8 ± 4.6	74.0 ± 7.5	70.9 ± 8.1	$+7.2$	3.6	81.7 ± 1.0	78.6 ± 10.9	24.9 ± 2.6	-23.3	61.4
+ first-obs KL	76.0 ± 7.3	76.5 ± 1.6	65.1 ± 3.6	$+1.4$	12.9	82.2 ± 2.6	82.3 ± 6.2	36.4 ± 0.6	-20.9	52.5
+ replay KL	79.3 ± 4.1	79.1 ± 3.5	74.0 ± 3.5	$+10.3$	5.8	81.3 ± 6.3	84.0 ± 6.6	73.9 ± 6.3	$\mathbf{+2.0}$	11.6
D2R (Ours)	79.8 ± 3.8	$\mathbf{79.8 \pm 3.8}$	$\mathbf{74.8 \pm 3.7}$	$\mathbf{+11.1}$	5.7	82.2 ± 9.3	82.1 ± 2.7	$\mathbf{74.9 \pm 2.6}$	$+1.6$	**8.3**
Per-task modules	81.8 ± 2.9	81.8 ± 2.9	81.8 ± 2.9	$+18.1$	0.0	91.3 ± 1.0	91.3 ± 1.0	91.3 ± 1.0	$+80.9$	0.0
Multi-task module	67.3 ± 3.9	67.3 ± 3.9	67.3 ± 3.9	×	×	54.6 ± 8.5	54.6 ± 8.5	54.6 ± 8.5	×	×

S.5.2 Full Per-Task Results

Every cell of Figures S.3 and S.4 is the retention $S_{k,j} - S_{0,j}$ of §S.4.6, the change in success on the evaluated task relative to the frozen base policy. Green marks a gain, pink a loss, and white a system that ends where it started. This is not the encoding of Figures 3 and 4 in the main text, whose cells carry two different quantities on two scales, the success reached on the diagonal and the points lost below it, and whose upper triangle is therefore empty. Here one signed quantity fills the whole matrix, which separates two things those figures combine. A method that forgets heavily can still end above the base policy, and a method that forgets nothing can end below it.

The two figures differ in what a row means. In Figure S.3 each row is an independent adaptation to one task, so the rows are eight separate runs and carry no order. In Figure S.4 each row is a stage of the sequence, so reading down a column shows how one task fares as later tasks are learned, and the Avg column is the mean over the eight tasks.

Figure S.5 does the same for LIBERO, where the retained skills are the four suites π_0 was fine-tuned on rather than individual tasks. Table 2 reports only their mean after the last stage, so the figure adds the trajectory, showing whether the pretrained skills decay gradually or drop at one stage and whether some suites suffer more than others.

S.5.3 Initialization of the Steering Module

At each stage the steering module $\pi_k^{\mathcal{W}}$ can either continue from the module of the previous stage, $\pi_{k-1}^{\mathcal{W}}$, or start from scratch. Carrying the weights forward is the obvious choice, but fine-tuning the same network through a sequence of stages costs plasticity (Dohare et al., 2024), leaving the module worse at learning each new task. Starting from scratch does not discard the previous stage, since the rehearsal penalty of Eq. 4 still anchors $\pi_k^{\mathcal{W}}$ to $\pi_{k-1}^{\mathcal{W}}$ on the retained tasks, so behavior is carried by the loss rather than by the weights.

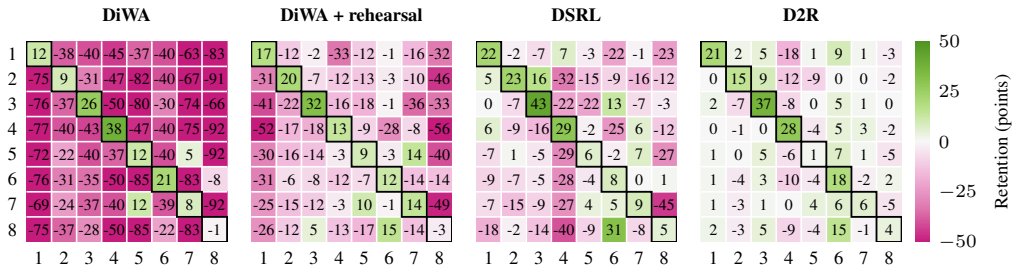


Figure S.3: Retention in single-task adaptation on CALVIN for all steered and fine-tuned methods of Table 1. Each cell shows the change in success relative to the base policy in points after adapting to the row task and evaluating on the column task, and outlined cells mark the adapted task. DiWA + rehearsal recovers most of DiWA's loss on the other tasks but remains below the base policy, DSRL drifts on several tasks, and D2R stays within a few points everywhere.

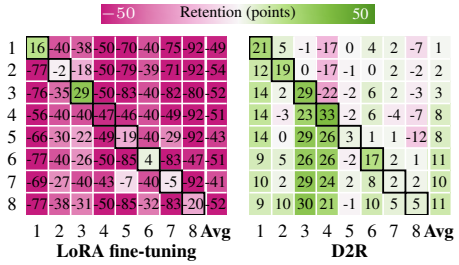


Figure S.4: Continual steering on CALVIN, the sequence counterpart of Figure 3. Change in success relative to the base policy after each stage (rows) on each task (columns) for LoRA fine-tuning and D2R, outlined cells mark the task learned at that stage, Avg is the mean over the eight tasks.

Table S.4 compares the two with everything else held fixed. Reinitializing is better on final success and on retention on both benchmarks, by 7.9 points on each axis on CALVIN and by 8.1 and 0.4 points on LIBERO, and it cuts forgetting on LIBERO from 18.6 to 8.3. Forgetting on CALVIN is unchanged. We therefore reinitialize at every stage, as recorded in Table S.1.

Table S.4: Initialization of the steering module and sensitivity to task order. Above the rule, initializing $\pi_k^{\mathcal{W}}$ from the previous stage $\pi_{k-1}^{\mathcal{W}}$ against initializing it from scratch at every stage, both on the default order A. Below the rule, three further orders, all initialized from scratch.

	CALVIN			LIBERO		
Setting	Success↑	Retention↑	Forgetting↓	Success↑	Retention↑	Forgetting↓
Base	63.7 ± 6.5	0.0	0.0	10.4 ± 2.7	0.0	0.0
From previous	66.9 ± 3.0	+3.2	5.4	66.8 ± 5.9	+1.2	18.6
From scratch	74.8 ± 3.7	+11.1	5.7	$\mathbf{74.9 \pm 2.6}$	**+1.6**	**8.3**
Order B	74.6 ± 2.0	+10.9	2.1	–	–	–
Order C	75.9 ± 7.1	+12.2	1.7	–	–	–
Order D	$\mathbf{78.9 \pm 1.9}$	**+15.2**	**1.0**	–	–	–

S.5.4 Task Order Sensitivity

We evaluate four task orders on CALVIN. Order A is the default used throughout the paper, and orders B, C and D are independent random permutations of the same eight tasks, with every other setting held fixed.

Table S.4 places all four within a few points of one another on every metric. Final success spans 74.6% to 78.9%, retention spans +10.9 to +15.2 points, and forgetting spans 1.0 to 5.7 points. Within that spread the default order is not a favorable one, ranking third of the four on success and on retention and last on forgetting, so the sequence reported in the main paper does not flatter the method.

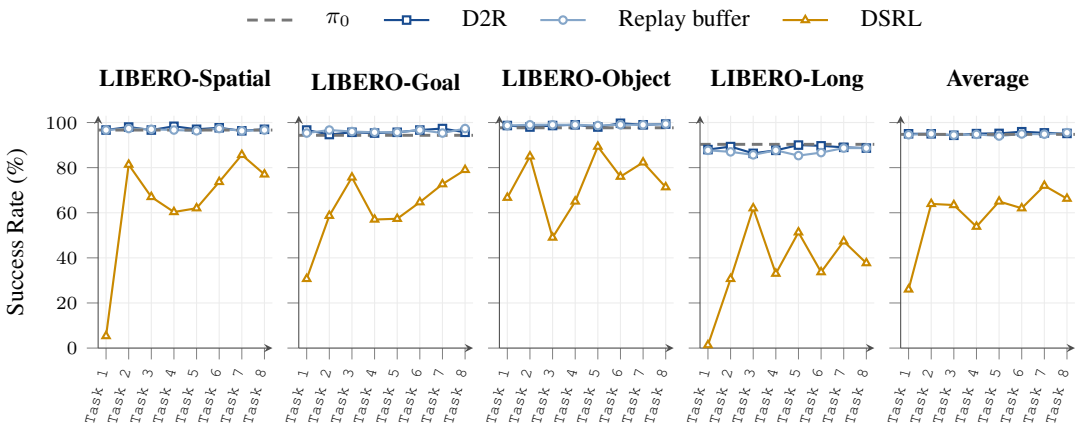


Figure S.5: **Retention of π_0's pretrained skills on LIBERO.** Success on each of the four suites π_0 was fine-tuned on after every stage of the eight-task sequence, with their mean in the last panel. The dashed line is the initial π_0, and the mean after the last stage is the LIBERO retention of Table 2.

S.5.5 Retention Coefficient λ

The retention coefficient λ controls the trade-off between learning the current task and preserving performance on previously learned tasks. As shown in Figure S.6(a) on the `move_slider_left` task from CALVIN benchmark, setting $\lambda = 0$ leads to lower retention, whereas an excessively large value limits adaptation to the current task. We use $\lambda = 0.005$, which provides the best balance between current task success and average retained success.

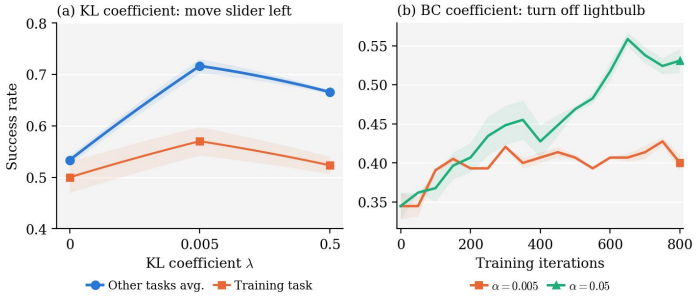


Figure S.6: **Loss coefficients on CALVIN. (a)** Effect of the retention coefficient λ on the success of the target task (`move_slider_left`) and the average success across previously learned tasks. **(b)** Effect of the behavior-cloning coefficient α on learning the target task (`turn_off_lightbulb`) over training.

S.5.6 Behavior Cloning Coefficient α

The behavior cloning coefficient α controls the contribution of the behavior cloning objective during steering policy optimization. Figure S.6(b) compares $\alpha = 0.005$ and $\alpha = 0.05$ on the `turn_off_lightbulb` task from CALVIN benchmark. A small coefficient can result in shortcut learning, world model exploitation, or reward hacking, which leads to poor performance when tested on the real environment. A balanced coefficient results in faster learning and a higher final success rate, so we use $\alpha = 0.05$ in our experiments. A larger value could instead reduce the exploration of the policy and limit it to the base policy.

S.5.7 World Model Faithfulness Check

Figure S.7 shows that A2World cannot serve as a learning simulator for the LIBERO-90 tasks, which lie outside its training data, while remaining faithful on the four suites it was fine-tuned on. For each suite we train two behavior-cloning policies with the same architecture and hyperparameters and different training data. The first is trained on 50 real expert demonstrations from that suite. The second is trained on 50 imagined demonstrations, obtained by initializing A2World with the initial scene of each expert demonstration and rolling it out under the corresponding expert action sequence. If the world model were faithful, the two policies would reach comparable success.

On the four suites they do. Cloning from imagined replays reaches 70.3% against 74.5% on Spatial, 82.0% against 82.5% on Goal, 88.9% against 82.4% on Object, and 46.0% against 45.8% on Long. On the eight LIBERO-90 tasks of the sequence the gap is wide, 19.6% against 74.9%. A2World reproduces the dynamics it was trained on and not those of the target tasks, so a steering module optimized inside it would exploit its prediction errors rather than learn the task, which is why LIBERO adapts in the environment and uses the world model only for rehearsal.

The BC policy is a Gaussian policy consisting of four convolutional layers, text conditioning with FiLM fusion, and three multilayer perceptron (MLP) layers that predict the mean and standard deviation of the action distribution. The policy predicts action chunks of length 20 and is trained for 100 epochs using a negative log-likelihood objective, a batch size of 256, Adam optimizer, and a learning rate of 10^{-4}. We evaluate both policies on each LIBERO suite using 30 episodes per task and three different evaluation seeds.

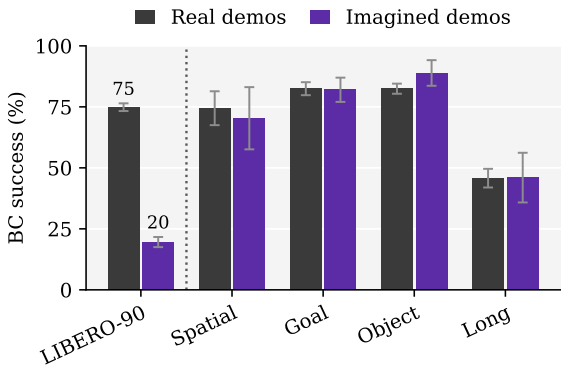


Figure S.7: **Faithfulness of A2World on LIBERO.** Success of behavior cloning on real demonstrations against A2World-imagined replays of the same demonstrations. The two agree on the four suites A2World was fine-tuned on and diverge on the LIBERO-90 tasks of the sequence.

S.5.8 Qualitative Imagined Rehearsal Rollouts

We provide qualitative comparisons between imagined rehearsal rollouts and their corresponding ground-truth trajectories for the world models used in our experiments. Figure S.8 shows A2World rollouts across the LIBERO suites. Although A2World does not consistently generate complete and successful task trajectories on LIBERO-90, it captures the environment dynamics and produces plausible transitions between different states. These rollouts therefore remain useful for rehearsal, where the objective is to preserve the steering policy's behavior on previously learned tasks rather than to learn a new task entirely from imagination.

Figure S.9 presents analogous results for the DreamerV2 world model used in the CALVIN and real-world experiments. Compared with A2World on the unseen LIBERO-90 tasks, DreamerV2 produces imagined trajectories that more closely follow the corresponding ground-truth task progression. This higher fidelity enables DreamerV2 to serve both as a simulator for learning new tasks in imagination and as a generative memory for rehearsing previously learned tasks.

Table S.2: Task instructions used in CALVIN and LIBERO. The real tasks are shown in Figure S.2.

CALVIN		LIBERO-90	
ID	**Task Instruction**	**ID**	**Task Instruction**
1	Close the drawer.	0	Close the top drawer of the cabinet.
2	Move the slider to the left.	7	Open the top drawer of the cabinet.
3	Move the slider to the right.	9	Put the black bowl on the plate.
4	Open the drawer.	10	Put the black bowl on top of the cabinet.
5	Turn off the LED.	56	Pick up the butter and put it in the tray.
6	Turn off the lightbulb.	57	Pick up the cream cheese and put it in the tray.
7	Turn on the LED.	20	Turn on the stove.
8	Turn on the lightbulb.	35	Open the microwave.
LIBERO-Spatial		**LIBERO-Goal**	
ID	**Task Instruction**	**ID**	**Task Instruction**
0	Pick up the black bowl between the plate and the ramekin and place it on the plate.	0	Open the middle drawer of the cabinet.
1	Pick up the black bowl next to the ramekin and place it on the plate.	1	Put the bowl on the stove.
2	Pick up the black bowl from the table center and place it on the plate.	2	Put the wine bottle on top of the cabinet.
3	Pick up the black bowl on the cookie box and place it on the plate.	3	Open the top drawer and put the bowl inside.
4	Pick up the black bowl in the top drawer of the wooden cabinet and place it on the plate.	4	Put the bowl on top of the cabinet.
5	Pick up the black bowl on the ramekin and place it on the plate.	5	Push the plate to the front of the stove.
6	Pick up the black bowl next to the cookie box and place it on the plate.	6	Put the cream cheese in the bowl.
7	Pick up the black bowl on the stove and place it on the plate.	7	Turn on the stove.
8	Pick up the black bowl next to the plate and place it on the plate.	8	Put the bowl on the plate.
9	Pick up the black bowl on the wooden cabinet and place it on the plate.	9	Put the wine bottle on the rack.
LIBERO-Object		**LIBERO-Long**	
ID	**Task Instruction**	**ID**	**Task Instruction**
0	Pick up the alphabet soup and place it in the basket.	0	Put both the alphabet soup and the tomato sauce in the basket.
1	Pick up the cream cheese and place it in the basket.	1	Put both the cream cheese box and the butter in the basket.
2	Pick up the salad dressing and place it in the basket.	2	Turn on the stove and put the moka pot on it.
3	Pick up the BBQ sauce and place it in the basket.	3	Put the black bowl in the bottom drawer of the cabinet and close it.
4	Pick up the ketchup and place it in the basket.	4	Put the white mug on the left plate and put the yellow and white mug on the right plate.
5	Pick up the tomato sauce and place it in the basket.	5	Pick up the book and place it in the back compartment of the caddy.
6	Pick up the butter and place it in the basket.	6	Put the white mug on the plate and put the chocolate pudding to the right of the plate.
7	Pick up the milk and place it in the basket.	7	Put both the alphabet soup and the cream cheese box in the basket.
8	Pick up the chocolate pudding and place it in the basket.	8	Put both moka pots on the stove.
9	Pick up the orange juice and place it in the basket.	9	Put the yellow and white mug in the microwave and close it.

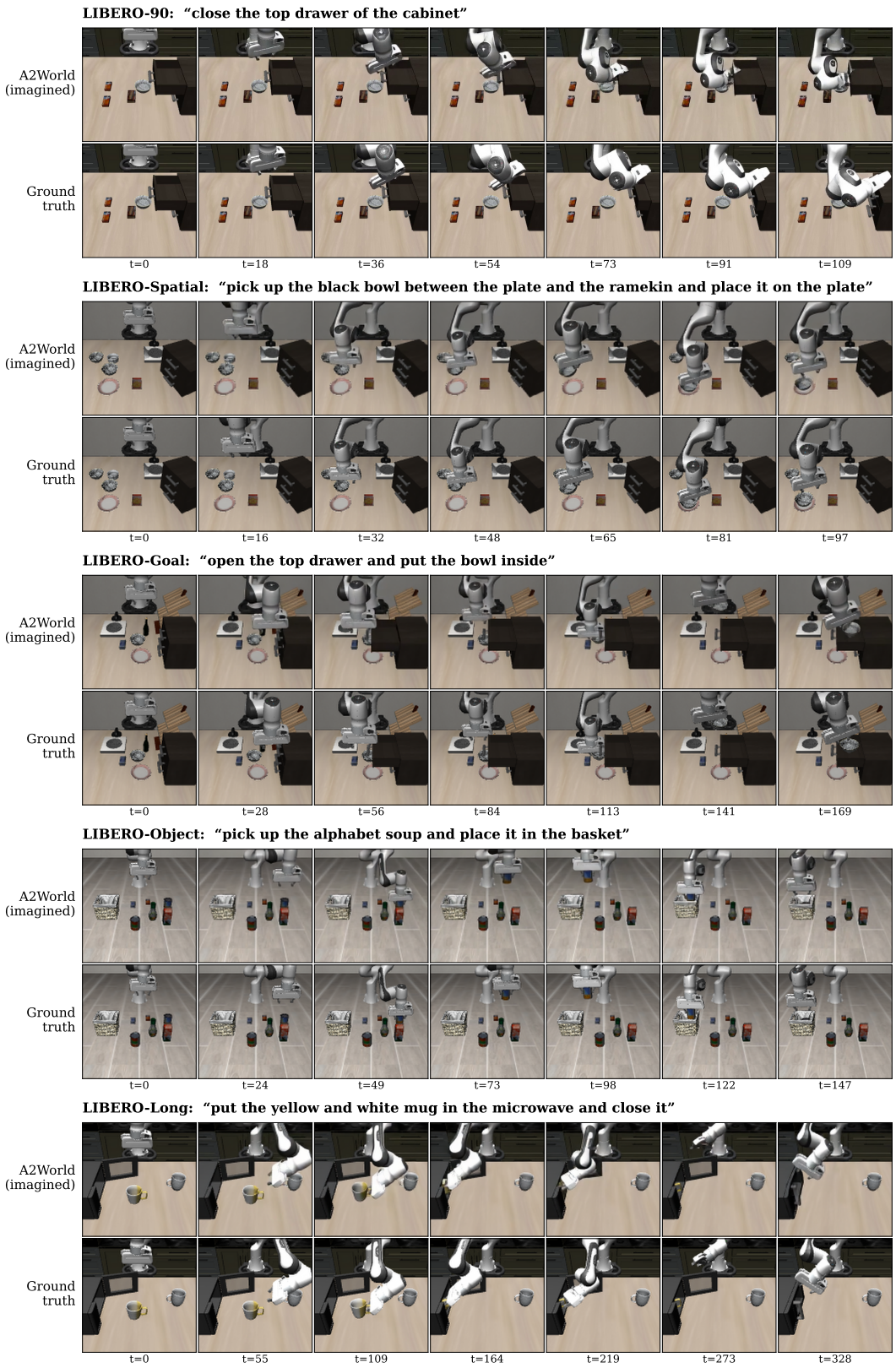


Figure S.8: Qualitative comparison between trajectories imagined by A2World and the corresponding ground-truth trajectories across the LIBERO suites. Each pair of rows shows temporally aligned frames from an A2World rollout and the associated ground-truth demonstration.

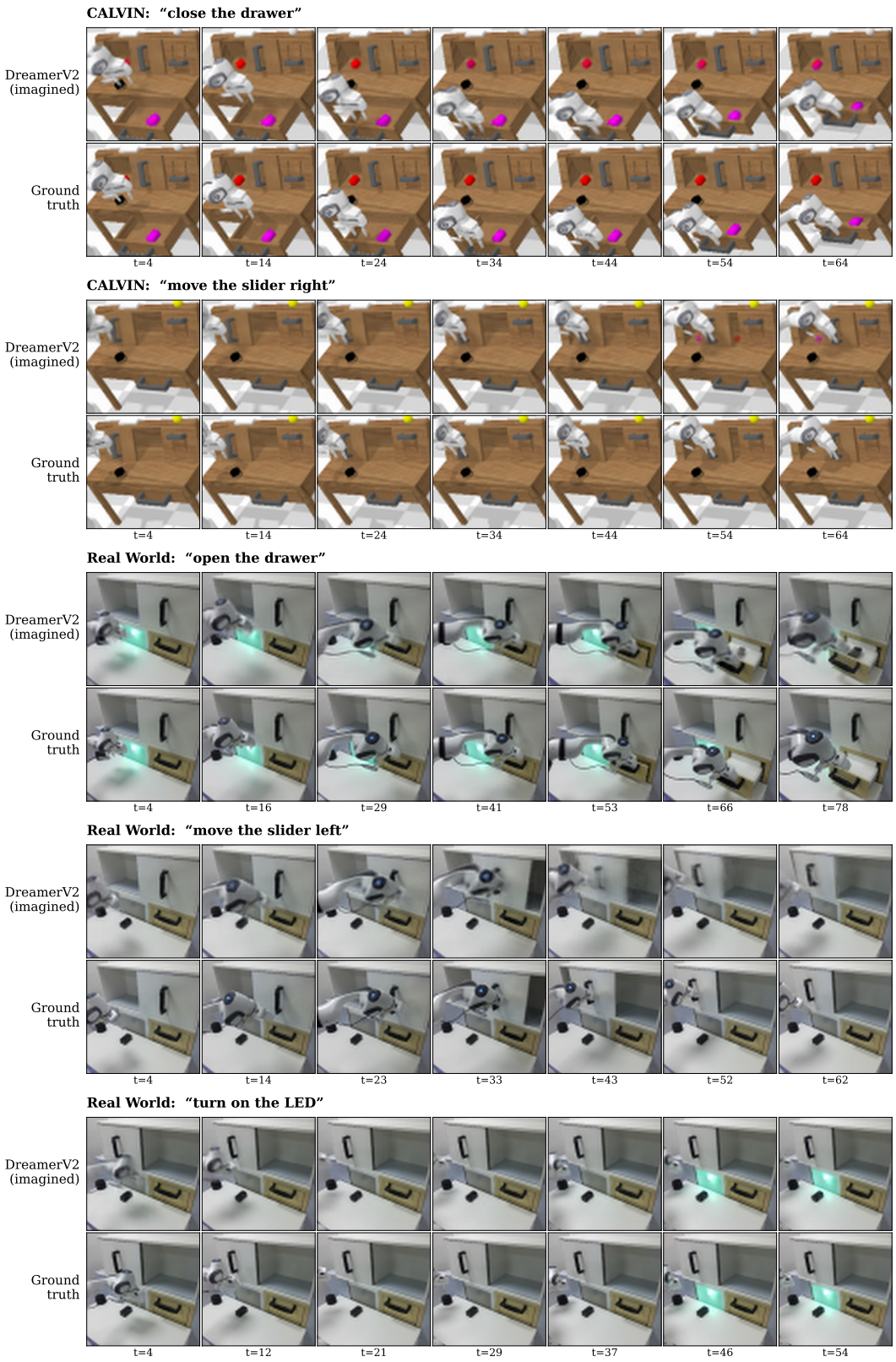


Figure S.9: Qualitative comparison between trajectories imagined by DreamerV2 and the corresponding ground-truth trajectories across the CALVIN benchmark and Real-world experiment. Each pair of rows shows temporally aligned frames from an imagined rollout and the associated ground-truth demonstration.