DATA-FREE CONTINUAL LEARNING FOR WORLD ACTION MODEL WITH RECTIFIED GENERATIVE REPLAY

Anonymous authors
Paper under double-blind review

ABSTRACT

World action models (WAMs) built on pretrained video generation models adapt quickly to new robotic tasks, but they forget previously learned tasks under sequential fine-tuning. Experience replay (ER) mitigates such forgetting by rehearsing data of previous tasks, yet this data is often unavailable in robotics, where a policy is deployed as weights only and its training data is not shared along with it. In this work, we ask whether a WAM can achieve data-free continual learning by regenerating its own past. Our key observation is that a WAM can indeed autoregressively regenerate complete rollouts of previously learned tasks and replay them to mitigate forgetting. However, replaying such raw rollouts still lags far behind replaying real data, which we attribute to two issues. First, the WAM is unable to recognize task completion and keeps generating purposeless motion after the task is done. Second, under open-loop generation, the generated actions gradually drift away from the generated observations. To address both issues, we present rectified generative replay (R^2-WAM), which truncates generated rollouts at task completion with an agentic evaluator and filters out rollouts whose actions are physically inconsistent with the observations. On the four LIBERO suites and five real-world tasks, R^2-WAM reduces forgetting by over 90% relative to sequential fine-tuning and nearly matches ER without accessing any data from previous tasks. Our analysis further reveals that forgetting in WAMs concentrates in inverse dynamics modeling: a forgotten WAM still produces visually plausible observations of previous tasks while its generated actions fail.

1 INTRODUCTION

Robots deployed in the real world keep encountering new tasks, so a practical policy has to learn them continually while retaining the tasks it has already learned, a capability known as continual learning (Liu et al., 2023). Among robot policies, world action models (WAMs) have emerged as a promising paradigm, as they leverage powerful pretrained video generation models (Blattmann et al., 2023; Team Wan et al., 2025; NVIDIA, 2025) to jointly predict future observations and robot actions (Li et al., 2026; Kim et al., 2026a; Ye et al., 2026). We find that sequentially fine-tuned WAMs adapt well to new tasks yet collapse on previously learned ones, as shown in Fig. 1(a), i.e., they suffer from catastrophic forgetting.

To mitigate catastrophic forgetting, continual learning methods either constrain parameter updates or perform experience replay (ER). The former protects prior knowledge by regularizing (Kirkpatrick et al., 2017) or isolating (Rusu et al., 2016) parameters, but regularization retains little for robotic policies (Liu et al., 2023), and isolation needs a task identifier at test time. ER instead maintains a buffer of demonstrations from previous tasks and mixes them with new-task data during fine-tuning (Rolnick et al., 2019), as shown on the right of Fig. 1(a), and remains among the most effective and robust choices for robotic policies (Liu et al., 2023; 2026). However, ER assumes that demonstrations accumulate along with the tasks, whereas a policy in practice often arrives as weights alone, and the demonstrations that trained it are no longer at hand. A policy should therefore learn continually in a *data-free* manner, without access to any data from previous tasks. WAMs offer a potential way to achieve this, since they model how the world evolves and can autoregressively predict future observations and actions, rolling out complete trajectories of previously learned tasks for replay. This raises a natural question: *Can a WAM leverage its own generative capability to achieve data-free continual learning as effectively as ER?*

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.

In this work, we answer this question with rectified generative replay (R^2-WAM), which regenerates rollouts of previous tasks with the WAM itself and rectifies them before replay, as shown in Fig. 1(b). Building on LingBot-VA (Li et al., 2026), we find that a WAM can indeed regenerate complete rollouts of previously learned tasks from their instructions alone, and replaying them already mitigates forgetting. However, replaying such raw rollouts still lags far behind ER, which we attribute to two issues. **First**, a WAM is unable to recognize task completion and receives no external stop signal, so generation runs for a fixed number of steps and the rollout keeps moving after the task is done, producing purposeless post-completion motion. **Second**, open-loop generation feeds the WAM's own predictions back without physical grounding, so errors accumulate and the generated actions gradually drift away from the generated observations. R^2-WAM addresses both issues with completion-aware adaptive truncation and flow-based physical-consistency filtering. The former removes post-completion motion with an agentic evaluator that detects task completion and adaptively truncates each rollout. The latter checks whether the generated actions agree with the generated observations by comparing two optical flows of the robot: one simulated by executing the generated actions in a robot-only simulator, which is possible since the robot itself is fully specified by its URDF, and one estimated from the generated observations. We rank the rollouts by the agreement between the two flows and keep the most consistent ones for replay. The rectified rollouts are then mixed with new-task data for fine-tuning.

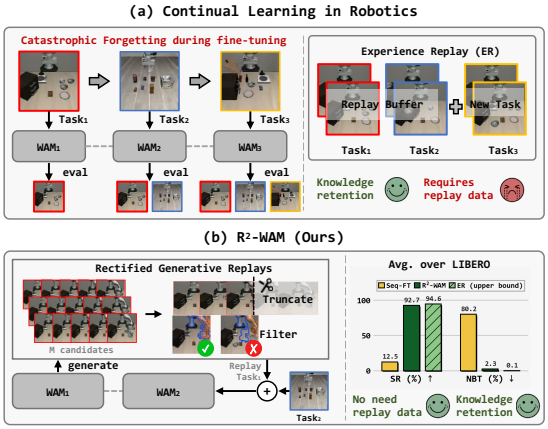


Figure 1: **(a)** In continual learning, a WAM is sequentially fine-tuned on new tasks and evaluated on all tasks learned so far. Experience replay (ER) mitigates forgetting of earlier tasks but requires access to their data. **(b)** R^2-WAM exploits the autoregressive generation of WAMs to achieve data-free continual learning, and its rectified generative replay attains performance comparable to ER.

We evaluate R^2-WAM on the four LIBERO suites (Liu et al., 2023) and a real-world suite of five tasks. Under the data-free continual learning setting, R^2-WAM reduces forgetting by over 90% relative to sequential fine-tuning on both LIBERO and the real world, and recovers almost all of the retention of ER without any data from previous tasks. Our ablations confirm that both truncation and filtering are necessary, and our analysis further reveals that forgetting in a WAM concentrates in inverse dynamics modeling, whereas its ability to predict observations is largely preserved. As a result, a forgotten WAM still produces visually plausible observations while its generated actions fail, which also points to the action side as the part to verify.

Our main contributions are summarized as follows:

- We show that, without any real data of previous tasks, a WAM can mitigate catastrophic forgetting by replaying its own generated rollouts and, once rectified, nearly match ER.
- We propose R^2-WAM, which rectifies the generated rollouts before replay with completion-aware adaptive truncation and flow-based physical-consistency filtering.
- In simulation and the real world, R^2-WAM eliminates over 90% of the forgetting of sequential fine-tuning and comes within about 2 pp of ER without any data from previous tasks, and find that forgetting in WAMs concentrates in inverse dynamics modeling.

2 Related Work

Continual learning in robotics. Continual learning methods mitigate forgetting by regularizing parameter updates or outputs (Kirkpatrick et al., 2017; Zenke et al., 2017; Li & Hoiem, 2018), isolating or adding task-specific parameters (Rusu et al., 2016; Mallya & Lazebnik, 2018; Hu et al., 2022; Wang et al., 2022), or storing a subset of previous-task data and rehearsing it, known as experience replay (Rolnick et al., 2019; Chaudhry et al., 2019; Caccia et al., 2022; Wang et al., 2024). In robot learning, experience replay remains a strong baseline from LIBERO to recent VLA

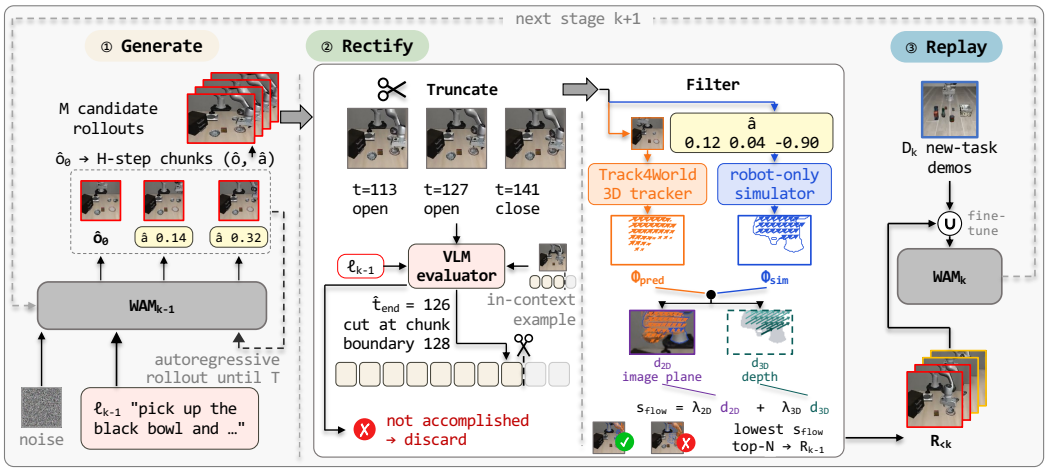


Figure 2: **Overview of R^2-WAM.** After learning task $k-1$, WAM_{k-1} ① **generates** M candidate rollouts of that task with the instruction ℓ_{k-1}, which are then ② **rectified** by truncating each rollout at task completion and filtering out physically inconsistent ones. The top-N retained rollouts $\mathcal{R}_{k-1}$ join those of earlier tasks to form the buffer $\mathcal{R}_{<k}$, which is ③ **replayed** with the new-task demonstrations $\mathcal{D}_k$ when learning task k. The same procedure then repeats for the next stage $k+1$.

policies (Liu et al., 2023; Wu et al., 2025b; Liu et al., 2026), while other methods rely on task-specific adapters (Liu et al., 2024; Römer et al., 2026), skill libraries or prompts (Wan et al., 2024; Yao et al., 2025; Zheng et al., 2025), or representation distillation (Roy et al., 2025). R^2-WAM follows the replay paradigm but stores no data from previous tasks, regenerating the replay data from their instructions instead, a setting known as data-free continual learning in image classification (Yin et al., 2020; Smith et al., 2021).

Generative replay. Unlike experience replay, which must store data from previous tasks, generative replay rehearses samples drawn from a generative model (Robins, 1995; Shin et al., 2017), either a separate generator whose outputs are labeled by conditioning or by an old copy of the learner (Wu et al., 2018; Gao & Liu, 2023; Kim et al., 2024; Gao et al., 2021; Yue et al., 2024; Pan et al., 2025), or the learner itself prompted to produce its own replay data (van de Ven et al., 2020; Sun et al., 2020; Huang et al., 2024; Wang et al., 2025). Closest to our work, REGEN (Govind et al., 2026) and Dream Rehearsal (Nijjer, 2026) replay generated rollouts for continual imitation and reinforcement learning. Both start generation from real observations, and Dream Rehearsal further scores the rollouts with the RL reward. In contrast, R^2-WAM generates rollouts from the instruction alone and rectifies them without any reward.

World action models. Building on pretrained video generation models, world action models jointly predict future observations and robot actions (Wu et al., 2024; Guo et al., 2024; Li et al., 2025; Zhu et al., 2025; Liao et al., 2026; Kim et al., 2026a; Ye et al., 2026; Li et al., 2026), and have shown strong performance and generalization in manipulation. Beyond serving as a policy, a WAM, like a video world model (Jang et al., 2025; Guo et al., 2026), can also act as a data generator to synthesize additional training data, and the consistency between generated observations and actions has been used to filter such data with a learned classifier (Kim et al., 2026b), to select rollouts at test time (Ruan et al., 2026), and to terminate drifting rollouts during policy evaluation (Tseng et al., 2026). We extend this line to continual learning, regenerating previously learned tasks for data-free replay, and rectify the rollouts without training any additional model.

3 Method

We present R^2-WAM, which regenerates replay data for previously learned tasks from their instructions alone and rectifies it before replay, as illustrated in Fig. 2: the WAM generates candidate rollouts of each task in Sec. 3.2, which are rectified by completion-aware adaptive truncation and flow-based physical-consistency filtering in Sec. 3.3 and replayed with new-task demonstrations.

3.1 PRELIMINARIES

Continual learning in robotics. We model a robotic manipulation task $\mathcal{T}_k$ by a task instruction ℓ_k and an expert demonstration dataset $\mathcal{D}_k = \{\tau_i^k\}_{i=1}^{|\mathcal{D}_k|}$. Each trajectory $\tau_i^k = \{(o_t, a_t)\}_{t=0}^{L_i^k-1}$ contains visual observations o_t and robot actions a_t, with L_i^k denoting its length. Continual learning considers a robot that learns a sequence of tasks $\{\mathcal{T}_k\}_{k=1}^{K}$ with a single policy $\pi_\theta(a_t \mid o_{\le t}, \ell_k)$ shared across all tasks and conditioned on the instruction ℓ_k. At each stage, the policy is optimized by behavior cloning on the demonstrations available for the current task:

$$\mathcal{L}_{\mathrm{BC}}(\theta; \mathcal{D}_k) = \mathbb{E}_{\tau \sim \mathcal{D}_k}\left[\sum_{t=0}^{|\tau|-1} -\log \pi_\theta(a_t \mid o_{\le t}, \ell_k)\right]. \tag{1}$$

Sequential adaptation can degrade performance on previously learned tasks, resulting in catastrophic forgetting. Experience replay (ER) mitigates this by maintaining a replay buffer $\mathcal{B}_{<k}$ that stores a subset of demonstrations from earlier tasks, and optimizing the policy on $\mathcal{D}_k \cup \mathcal{B}_{<k}$ at stage k. In contrast, we study the *data-free* setting in which no data from previous tasks is retained. After adapting to a task, only its instruction and the updated model parameters carry over to later stages, while its demonstrations and task environment are no longer accessible.

World action models. Unlike a conventional policy that maps observations to actions, a WAM also predicts the future observations that its actions would produce (Li et al., 2025; 2026; Kim et al., 2026a; Ye et al., 2026). Given a task $\mathcal{T}_k$ with instruction ℓ_k and a trajectory prefix $\tau_{\le t}$, it models the joint conditional distribution:

$$p_\theta\big(o_{t+1:t+H}, a_{t:t+H-1} \mid \tau_{\le t}, \ell_k\big), \tag{2}$$

where H is the chunk size, i.e., the number of observations and actions predicted in one step. In this paper, we adopt LingBot-VA (Li et al., 2026) as a representative WAM, which implements this formulation with a causal autoregressive architecture that interleaves observation and action tokens.

3.2 GENERATING ROLLOUTS WITH WAM

We leverage the autoregressive rollout capability of WAMs to synthesize replay data, generating a rollout for each learned task $\mathcal{T}_j$ with its instruction ℓ_j as the only input. Built on a video generation model, the WAM can generate observations from text alone, an ability we retain during fine-tuning by keeping the initial observation as a prediction target conditioned on the instruction. A rollout thus starts by sampling the initial observation $\hat{o}_0 \sim p_\theta(o_0 \mid \ell_j)$ and proceeds chunk by chunk: at step t, the WAM first predicts the next H observations from the generated history $\hat{\tau}_{\le t}$, appends them to the model input, and then predicts the corresponding actions conditioned on them:

$$\hat{o}_{t+1:t+H} \sim p_\theta\big(\cdot \mid \hat{\tau}_{\le t}, \ell_j\big) \;\longrightarrow\; \hat{a}_{t:t+H-1} \sim p_\theta\big(\cdot \mid \hat{\tau}_{\le t}, \hat{o}_{t+1:t+H}, \ell_j\big), \tag{3}$$

which are appended to the model input as well. This repeats until a predefined horizon T, yielding the rollout $\hat{\tau}_j = \{(\hat{o}_t, \hat{a}_t)\}_{t=0}^{T-1}$.

3.3 RECTIFIED GENERATIVE REPLAY

Completion-aware adaptive truncation. As discussed in Sec. 3.2, we generate rollouts with a fixed horizon T for each task. Since the WAM is unable to recognize task completion, it keeps generating after the task is done, producing purposeless post-completion motion such as re-grasping the placed object or wandering around. Such motion enters training as if it were part of the task, which corrupts the learned behavior of the task. To remove it, we use a VLM as an agentic evaluator to identify the completion point of each of the M candidate rollouts generated for a task. Rather than feeding the whole video, we subsample candidate frames from the later part of the rollout and annotate each frame with its step index, gripper command, and end-effector speed from the generated actions, so that the evaluator can localize the completion precisely. We further provide one in-context example from the same task, a generated rollout whose truncation range is annotated by a human (one per task), to calibrate its judgment. Given the task instruction and these inputs, the evaluator determines whether the task goal has been achieved and returns the end step $\hat{t}_{\mathrm{end}}$, defined as the point where the completing motion has fully finished, e.g., the object has been released and the gripper has retreated. For tasks with multiple sub-goals, the end step corresponds to the completion

of the final sub-goal. We then truncate the rollout at the chunk boundary following $\hat{t}_{\text{end}}$, so that the completed interaction is retained while the motion after it is excluded. Rollouts judged as not accomplishing the task are discarded. The evaluator prompt and a truncation example are given in Fig. 12 and Fig. 9 in the appendix.

Flow-based physical-consistency filtering. The remaining issue is the observation–action drift caused by open-loop generation, which we address by scoring each rollout's drift and filtering out the worst. Measuring this drift requires knowing what actually happens when the generated actions are executed, which is unavailable since the environments of previous tasks are no longer accessible. Fortunately, the robot itself is fully specified by its URDF and camera parameters, so the motion of the robot under the generated actions can be simulated without any environment and serves as a reference. Concretely, we execute the actions of each truncated rollout $\tilde{\tau}$ in a robot-only simulator and record the resulting optical flow Φ_{sim} of the robot. In parallel, we estimate the optical flow Φ_{pred} from the generated observations with Track4World (Lu et al., 2026), a dense 3D tracker. Comparing the two flows on the robot pixels, we define the physical-consistency score as:

$$s_{\text{flow}}(\tilde{\tau}) = \lambda_{\text{2D}}\, d_{\text{2D}}\left(\Phi_{\text{pred}}, \Phi_{\text{sim}}\right) + \lambda_{\text{3D}}\, d_{\text{3D}}\left(\Phi_{\text{pred}}, \Phi_{\text{sim}}\right), \tag{4}$$

where d_{2D} and d_{3D} measure the discrepancy of the two flows in the image plane and in 3D camera space, respectively, and lower scores indicate better agreement. Appendix A.2 describes both terms and the weights λ_{2D} and λ_{3D}. We rank the candidates by s_{flow} and retain the top-N as the replay set $\mathcal{R}_j$ for task $\mathcal{T}_j$.

Replay with rectified rollouts. Finally, R^2-WAM replays the rectified rollouts exactly as ER replays real demonstrations in Sec. 3.1, replacing the replay buffer $\mathcal{B}_{<k}$ with the rectified rollouts $\mathcal{R}_{<k} = \bigcup_{j<k} \mathcal{R}_j$ and fine-tuning the WAM on $\mathcal{D}_k \cup \mathcal{R}_{<k}$ at stage k by maximizing the likelihood of Eq. (2). Each $\mathcal{R}_j$ is generated once and cached for all later stages, so $\mathcal{R}_{<k}$ matches ER's $\mathcal{B}_{<k}$ in size without any real data.

4 Experiments

4.1 Experiment Setup

Benchmarks. We evaluate continual learning in simulation on LIBERO and in the real world on five sequentially learned manipulation tasks. **For simulation**, we use LIBERO (Liu et al., 2023), a benchmark designed for continual learning in robotic manipulation. Its four suites, LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-10, each define a sequence of 10 tasks with 50 demonstrations per task. The four suites differ in what changes across tasks: the spatial layout, the manipulated object, the task goal, and long-horizon compositions of the three. **For the real world**, we design a sequence of five sandwich-making tasks on a robot with a 7-DoF arm and a 20-DoF dexterous hand, which observes the scene through a head camera and a wrist camera mounted on the hand (Fig. 3). Each task is instructed as "make a sandwich with ⟨ingredient⟩", with the ingredient being steak, luncheon meat, cheese, a fried egg, or a meat slice in learning order, and requires the robot to grasp the ingredient, place it on the bread, grasp the top bread, and cover the sandwich (Fig. 8). We collect 50 teleoperated demonstrations per task; hardware and protocol details are given in Appendix A.1.

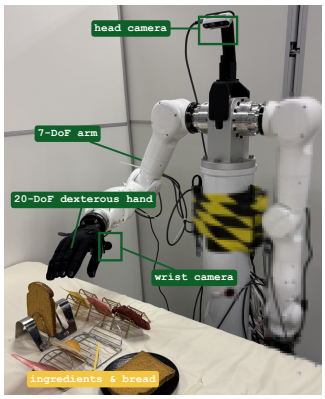


Figure 3: **Real-robot setup.**

Protocol and metrics. Following LIBERO, tasks are learned one after another, and after each task the model is evaluated on every task learned so far, which yields a lower-triangular task matrix. Tasks in every LIBERO suite are ordered lexicographically by their instructions, and the five real-robot tasks follow a fixed order shared by all methods; the full sequences are listed in Appendix A.6. In simulation, each task is evaluated for 50 trials and we record its success rate as defined by LIBERO. On the real robot, a binary success hides partial progress on these long-horizon tasks, so we report the progress rate instead: each trial is scored by how many of the four steps it completes, divided by

Table 1: **Continual learning results on the four LIBERO suites.** We compare methods without replay (top) and with replay (bottom); all replay methods use 20 episodes per previous task. ER (gray, italic) replays real demonstrations and serves as an upper bound. R-WAM† replays generated rollouts without rectification. ± denotes the standard deviation over three training seeds. Plast. denotes Plasticity; all metrics follow Section 4.1.

Method	LIBERO-Spatial			LIBERO-Object			LIBERO-Goal			LIBERO-10			Average		
	SR ↑	NBT ↓	Plast. ↑	SR ↑	NBT ↓	Plast. ↑	SR ↑	NBT ↓	Plast. ↑	SR ↑	NBT ↓	Plast. ↑	SR ↑	NBT ↓	Plast. ↑
Seq-FT	11.2	86.07	95.8	21.2	67.84	97.4	9.8	84.87	94.8	7.6	82.00	89.6	12.5	80.19	94.4
Seq-LoRA	9.6	75.42	86.2	17.6	52.47	87.0	9.8	68.72	84.2	7.4	71.24	80.4	11.1	66.96	84.4
EWC	12.0	85.59	96.2	23.6	62.19	98.0	10.0	84.55	95.2	7.8	82.24	90.6	13.3	78.64	95.0
LwF	13.0	85.14	96.0	27.6	61.76	96.8	10.4	83.70	95.0	8.6	78.96	88.8	14.9	77.39	94.2
ER	*94.0* ±0.3	*0.54* ±0.60	*94.3* ±1.1	*97.3* ±0.8	*−1.83* ±1.16	*95.5* ±1.7	*92.8* ±2.8	*2.11* ±1.45	*95.7* ±0.9	*94.1* ±0.3	*−0.46* ±0.43	*92.7* ±0.3	*94.6* ±0.5	*0.09* ±0.79	*94.6* ±0.7
R-WAM†	76.1 ±8.7	11.97 ±2.58	93.6 ±4.0	83.1 ±6.8	18.93 ±5.93	94.2 ±1.4	51.2 ±4.9	27.31 ±1.75	83.3 ±3.6	71.3 ±2.6	7.63 ±2.69	85.1 ±4.4	70.4 ±0.4	16.46 ±1.13	89.1 ±1.1
R^2-WAM (ours)	94.0 ±1.1	2.90 ±0.32	95.7 ±0.8	95.5 ±0.4	−0.31 ±1.05	95.3 ±0.8	92.2 ±2.3	2.00 ±3.17	95.5 ±3.1	88.9 ±4.2	4.73 ±2.28	92.0 ±1.8	92.7 ±1.2	2.33 ±1.08	94.7 ±0.4

four (Appendix A.1). Let $c_{i,k}$ denote the performance of task i after learning task k, averaged over its evaluation trials: the success rate over 50 trials in simulation, or the progress rate over 20 trials on the real robot. Let K denote the number of tasks. We compute three continual learning metrics from the task matrix:

$$\mathrm{SR} = \frac{1}{K}\sum_i c_{i,K}, \quad \mathrm{NBT} = \frac{1}{K}\sum_{k<K}\frac{1}{K-k}\sum_{j>k}(c_{k,k}-c_{k,j}), \quad \mathrm{Plasticity} = \frac{1}{K}\sum_k c_{k,k}. \tag{5}$$

SR is the average performance over all tasks evaluated with the final checkpoint, i.e., after learning the last task; NBT (negative backward transfer) (Liu et al., 2023) measures forgetting as the average drop of each task from the checkpoint that has just learned it to all later checkpoints; Plasticity is the average performance of each task at the checkpoint that has just learned it. Higher SR and Plasticity and lower NBT are better.

Baselines. All methods share the same LingBot-VA backbone (Li et al., 2026), training schedule, and evaluation protocol. We consider two groups of baselines. The first group does not replay: sequential fine-tuning (Seq-FT), sequential fine-tuning of LoRA adapters (Hu et al., 2022) (Seq-LoRA), EWC (Kirkpatrick et al., 2017), and LwF (Li & Hoiem, 2018). EWC constrains parameter updates with a quadratic penalty weighted by the Fisher importance of each parameter to previous tasks, and LwF regularizes the model by matching its predictions on new-task inputs to those of the previous model. The second group replays 20 episodes per previous task: experience replay (ER) (Rolnick et al., 2019) replays real demonstrations and serves as an upper bound, while R-WAM replays the generated rollouts without rectification. R-WAM keeps all $N = 20$ rollouts it generates, whereas R^2-WAM selects the same number from $M = 40$ candidates.

Implementation. After learning each task, R^2-WAM generates $M = 40$ candidate rollouts from the instruction alone, truncates them with a VLM evaluator, and retains the $N = 20$ rollouts with the best physical-consistency scores (all experiments use $M = 2N$), computed with Track4World (Lu et al., 2026) flow against a robot-only MuJoCo simulation. The retained rollouts are generated once and reused in all later stages. Replay data and new-task demonstrations are mixed with equal sampling weight. Training hyperparameters are given in Appendix A.1; generation horizons and the evaluator prompt are given in Appendix A.2.

4.2 Effectiveness of Rectified Generative Replay

R^2-WAM nearly matches ER without any data from previous tasks. Table 1 compares R^2-WAM with the no-replay and replay baselines on the four LIBERO suites; all replay methods use the same buffer size of 20 episodes per previous task. We draw three observations. **First**, without replay, the WAM forgets almost everything it has learned: Seq-FT, Seq-LoRA, EWC, and LwF reach an average SR of only 11–15% with an NBT of 67–80, although their Plasticity stays above 84%. **Second**, without any real data from previous tasks, R^2-WAM reaches an average SR of 92.7% and an NBT of 2.33, only 1.9 pp below ER on average, which serves as an upper bound since it replays real demonstrations of previous tasks. The gap is largest on LIBERO-10 (5.2 pp), consistent with its long-horizon tasks, where observation–action drift may accumulate over longer rollouts. **Finally**,

Table 2: **Real-world results on five sequential tasks.** Metrics are computed on the progress rate (Appendix A.1). ER (gray, italic) serves as an upper bound.

Method	SR ↑	NBT ↓	Plast. ↑
ER	*91.8*	*2.27*	*94.8*
Seq-FT	28.5	65.62	**98.8**
R^2-WAM (ours)	**89.5**	**3.45**	93.8

Table 3: **Ablation of R^2-WAM components** on LIBERO-Spatial. Each row removes one rectification component, and R-WAM applies none.

Variant	SR ↑	NBT ↓	Plast. ↑
R-WAM (no rectification)	76.1	11.97	93.6
R^2-WAM w/o truncation	70.8	13.73	89.2
R^2-WAM w/o filtering	80.8	7.02	91.4
R^2-WAM (ours)	**94.0**	**2.90**	**95.7**

replaying raw rollouts substantially reduces forgetting relative to Seq-FT, raising the average SR from 12.5% to 70.4%, but still lags ER by a wide margin, which shows the necessity of rectification.

Real-world results. Table 2 compares R^2-WAM with Seq-FT and ER on the five real-world tasks, where the trend matches simulation: Seq-FT suffers catastrophic forgetting (28.5% SR, 65.62 NBT), whereas R^2-WAM retains previous tasks nearly as well as ER (89.5% versus 91.8% SR, 3.45 versus 2.27 NBT) without any real data from previous tasks. Both replay methods also give up 4–5 pp of Plasticity relative to Seq-FT, which we attribute to interference among co-trained tasks (Yu et al., 2020; Wu et al., 2025a), here between tasks that differ only in a visually similar target. For more case studies and qualitative results, see Appendix A.4 and the supplementary video.

4.3 Ablation Study on R^2-WAM

Both components are necessary. Table 3 removes each rectification component on LIBERO-Spatial. **First**, removing either component degrades performance, from 94.0% SR to 80.8% without filtering and to 70.8% without truncation, so both are necessary. **Second**, removing truncation performs no better than R-WAM (70.8% vs. 76.1%), which replays the raw rollouts, suggesting that the purposeless motion at the end of a rollout may interfere with the consistency score and thus make filtering unreliable.

Ablation of completion-aware adaptive truncation. To examine adaptive truncation itself, we sample five generated rollouts from each of the 40 LIBERO tasks, annotate an acceptable range of truncation steps for each of the 200 samples, and count a predicted truncation as correct if it falls inside the range. Table 4 ablates the VLM evaluator and its prompt on this set. **First**, stronger VLMs truncate more accurately, from 73.5% for Qwen3-VL to 93.5% for GPT-5.6 Sol; we use Terra (91.0%) in all other experiments to balance accuracy against cost. **Second**, removing the in-context example lowers accuracy to 78.0%, likely because where a task ends is inherently ambiguous and the example anchors the VLM's notion of completion. **Finally**, removing the action annotations lowers accuracy to 76.0%, suggesting that the moment of completion is often hard to tell from observations alone and the motion cues of the actions help.

Table 4: **Ablation of completion-aware adaptive truncation.** Accuracy is the fraction of predicted truncations that fall inside the human-annotated range. We ablate the VLM (top) and the prompt design (bottom); Terra with the full prompt is our default.

	Setting	Acc. (%) ↑
VLM	GPT-5.6 Sol (OpenAI, 2026)	**93.5**
	GPT-5.6 Terra	91.0
	GPT-5.6 Luna	88.0
	Qwen3-VL (Bai et al., 2025)	73.5
Prompt	Full prompt	**91.0**
	w/o in-context example	78.0
	w/o action annotations	76.0

The physical-consistency score s_{flow} identifies executable rollouts. To verify this, we use the WAM to generate 400 rollouts on LIBERO-Spatial, starting from real initial observations so that their actions can be executed open-loop in simulation. We regard a rollout as executable if its execution accomplishes the task, and report the executable fraction of rollouts at different levels of s_{flow} in Fig. 4(a). The fraction decreases steadily from 82.5% for the most consistent 20% to 41.2% for the least consistent 20%, showing that s_{flow} indeed reflects whether the generated actions agree with the generated observations, and filtering by it keeps the executable rollouts. In Fig. 4(b), we further visualize the optical flows behind s_{flow}, where blue is simulated from the generated actions and orange is estimated from the generated observations. In a low-s_{flow} rollout the two flows point in the same direction throughout, whereas in a high-s_{flow} rollout they diverge over time and gradually separate, which eventually leads to observation–action drift.

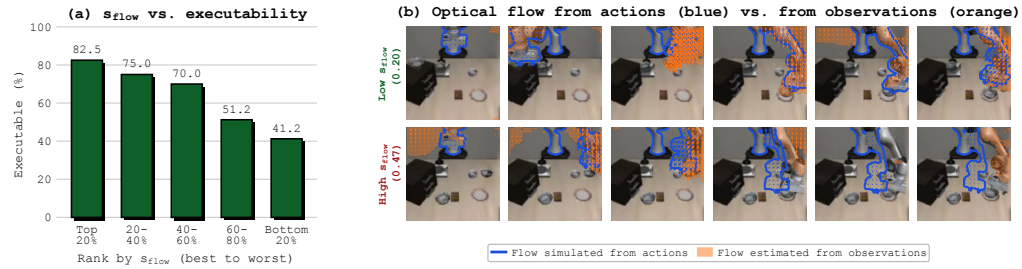


Figure 4: **What the physical-consistency score measures. (a)** We use the WAM to generate 400 rollouts on LIBERO-Spatial, group them by s_{flow} from most to least consistent, and execute their actions in simulation. Each bar is the fraction of rollouts in that group whose actions complete the task. **(b)** Optical flows behind s_{flow} for a low-score and a high-score rollout: blue is simulated from the generated actions, orange is estimated from the generated observations.

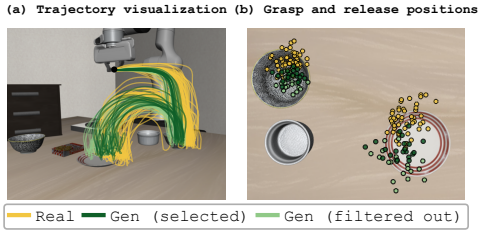


Figure 5: **Visualization of real and generated trajectories on LIBERO. (a)** Trajectories of 50 real demonstrations and 40 generated rollouts on a LIBERO-Spatial task, 20 selected by R^2-WAM and 20 filtered out. **(b)** Top view of where each trajectory grasps the bowl and releases it on the plate.

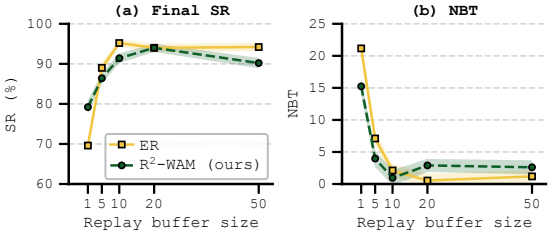


Figure 6: **Scaling the replay buffer on LIBERO-Spatial.** Final SR and NBT as the replay buffer grows from 1 to 50 episodes per previous task. ER replays real demonstrations of previous tasks, whereas R^2-WAM replays generated rollouts and uses no real data at all. Shading denotes the standard error over 50 trials per task.

Rectified rollouts overlap with real demonstrations but do not fully match them. To compare the replay data of ER and R^2-WAM more directly, we visualize the trajectories of the real demonstrations and the generated rollouts of a LIBERO-Spatial task in Fig. 5, where the 40 generated rollouts are all adaptively truncated and then split by the physical-consistency filtering into 20 selected and 20 filtered out. We draw three observations from the comparison. **First**, the generated trajectories follow the same pattern as the real ones in Fig. 5(a), but they gradually deviate from the real ones as the rollout proceeds, which is evident in Fig. 5(b), where the release positions deviate much more than the grasp positions. Since the generated videos are visually indistinguishable from real ones, this deviation reflects the observation–action drift accumulating over open-loop generation. **Second**, the physical-consistency score identifies this drift well, as the selected rollouts stay much closer to the real trajectories than the filtered-out ones. This also explains why the higher-ranked rollouts in Fig. 4(a) are more often executable. **Finally**, filtering only mitigates the drift rather than removing it, as some deviation remains in the selected rollouts, which is consistent with the residual gap to ER that remains on the longer-horizon suite.

Generated replay tracks real replay across buffer sizes. To examine the effect of the replay buffer size, we scale it from 1 to 50 episodes per task on LIBERO-Spatial and compare R^2-WAM with ER at every size, as shown in Fig. 6. **First**, both methods improve rapidly with the buffer size and no longer improve beyond 20 episodes, so we use a buffer of 20 in the main experiments to balance performance against generation cost. **Second**, R^2-WAM even outperforms ER at a buffer size of 1 (79.2% versus 69.6% SR), though ER overtakes it quickly and stays slightly ahead. We conjecture that a generated rollout reflects the typical behavior the model has learned for a task and thus anchors it better than a single real demonstration, which may be atypical or noisy. With more real demonstrations, however, ER covers the true distribution, while generated rollouts add little new coverage and still carry some residual drift even after rectification. The task matrices behind Fig. 6 are given in Fig. 10 in the appendix.

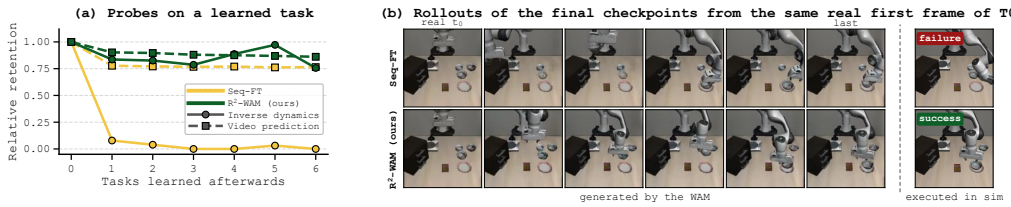


Figure 7: **Which capabilities of a WAM are forgotten. (a)** Relative retention $r(\Delta)$ (Eq. 6) of two probed capabilities, Δ tasks after a task was learned. Circles with solid lines: inverse dynamics modeling (actions from real observations); squares with dashed lines: video prediction (observations from real actions). Yellow: Seq-FT; green: R^2-WAM. **(b)** Rollouts generated by the final checkpoints from the same real first frame of T0, with the R^2-WAM rollout shown after truncation. The rightmost frame is rendered by executing the generated actions in simulation, which reveals whether they accomplish the task.

4.4 Diagnostic Study: Forgetting Concentrates in Inverse Dynamics

A WAM couples two capabilities, inverse dynamics modeling and video prediction, and we ask which of them is lost during continual learning. We probe each capability separately on LIBERO-Spatial with teacher forcing. For inverse dynamics modeling, the model receives real observations of a task and predicts the actions, which we execute in simulation and score by success rate. For video prediction, it receives the real actions and predicts the observations, scored by PSNR, which measures how closely the predicted frames match the real ones. Let $p_{i,k}$ be the score of a probe on task i after learning task k ($k \geq i$), over 20 samples. We measure forgetting by relative retention

$$r(\Delta) = \frac{1}{K-\Delta} \sum_{i=0}^{K-1-\Delta} \frac{p_{i,i+\Delta}}{p_{i,i}}, \tag{6}$$

i.e., the fraction of a task's initial score that remains Δ tasks later, averaged over tasks, where lower values mean more forgetting.

Fig. 7 reports the results, from which we draw three observations. **First**, Seq-FT forgets inverse dynamics modeling almost entirely: after learning a new task, the retention of this capability drops to 8% and never recovers, whereas under R^2-WAM it stays above 75% across all subsequent tasks. **Second**, video prediction is largely preserved, retaining 78% under Seq-FT and over 85% under R^2-WAM, so even a WAM whose actions have collapsed still imagines visually plausible observations. **Finally**, Fig. 7(b) visualizes rollouts generated by the final checkpoints of Seq-FT and R^2-WAM from the same real first frame of T0: both produce a fluent video of the task, but only the actions of R^2-WAM accomplish it when executed in simulation.

5 Conclusion

We presented R^2-WAM, a data-free approach to continual learning for world action models. A WAM can regenerate rollouts of previously learned tasks from their instructions alone, but such rollouts cannot be replayed directly for two reasons: the WAM does not stop generating at task completion, and its generated observations gradually drift from the generated actions. R^2-WAM addresses both issues with rectified generative replay, which truncates each rollout at task completion and filters out physically inconsistent ones before replay. On the four LIBERO suites and five real-world tasks, R^2-WAM reduces forgetting by over 90% relative to sequential fine-tuning and nearly reaches experience replay without retaining any data from previous tasks. Ablations confirm that both truncation and filtering are necessary, and our analysis shows that forgetting in a WAM concentrates in inverse dynamics modeling while video prediction is largely preserved. This explains why a WAM can regenerate its past for replay and points to the action side as the part to verify.

Limitations. Two limitations point to future work. First, a small gap to ER remains and is largest on long-horizon tasks, suggesting that filtering mitigates but does not fully resolve observation–action drift, and aligning the two at training time or grounding generation with an external world model are directions worth exploring. Second, the reduced plasticity of replay-based methods in our real-world experiments suggests that interference between tasks also needs to be addressed.

AI USE STATEMENT

In this work, we used generative AI tools for two tasks that require disclosure: implementing methods (assistants wrote code for the proposed method and the experimental pipeline under the authors' specifications) and cleaning and reformatting data (converting training data into the format required by the training pipeline, and aggregating evaluation results into the task matrices and tables reported in the paper). We have not used generative AI tools for the other tasks with required disclosure, and the remaining ones are not applicable to this work. Additionally, we used generative AI tools for drafting and editing the paper, polishing grammar and wording, producing plots and illustrations, identifying relevant literature, and formatting references. We have reviewed all AI-assisted work: all code was verified and tested by the authors, and all reported numbers were traced back to the experimental logs. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 4.1 and Appendix A give the full experimental protocol, including training details, the evaluator prompt, and task sequences. A supplementary video shows the real-robot experiments. Code will be released upon publication.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, Varun Jampani, and Robin Rombach. Stable Video Diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. In *International Conference on Learning Representations*, 2022.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with A-GEM. In *International Conference on Learning Representations*, 2019.

Chongkai Gao, Haichuan Gao, Shangqi Guo, Tianren Zhang, and Feng Chen. CRIL: Continual robot imitation learning via generative and prediction model. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 6747–6754, 2021. doi: 10.1109/IROS51168.2021.9636069.

Rui Gao and Weiwei Liu. DDGR: Continual learning with deep diffusion-based generative replay. In *International Conference on Machine Learning*, pp. 10744–10763, 2023.

Manish Kumar Govind, Dominick Reilly, Smit Patel, Hieu Le, and Srijan Das. World action models enable continual imitation learning with recurrent generative replays. *arXiv preprint arXiv:2606.27374*, 2026.

Yanjiang Guo, Yucheng Hu, Jianke Zhang, Yen-Jen Wang, Xiaoyu Chen, Chaochao Lu, and Jianyu Chen. Prediction with action: Visual policy learning via joint denoising process. In *Advances in Neural Information Processing Systems*, 2024.

Yanjiang Guo, Lucy Xiaoyang Shi, Jianyu Chen, and Chelsea Finn. Ctrl-World: A controllable generative world model for robot manipulation. In *International Conference on Learning Representations*, 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.

Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal. In *Annual Meeting of the Association for Computational Linguistics*, pp. 1416–1428, 2024.

Joel Jang, Seonghyeon Ye, Zongyu Lin, Jiannan Xiang, Johan Bjorck, Yu Fang, Fengyuan Hu, Spencer Huang, Kaushil Kundalia, Yen-Chen Lin, Loïc Magne, Ajay Mandlekar, Avnish Narayan, You Liang Tan, Guanzhi Wang, Jing Wang, Qi Wang, Yinzhen Xu, Xiaohui Zeng, Kaiyuan Zheng, Ruijie Zheng, Ming-Yu Liu, Luke Zettlemoyer, Dieter Fox, Jan Kautz, Scott Reed, Yuke Zhu, and Linxi Fan. DreamGen: Unlocking generalization in robot learning through video world models. In *Conference on Robot Learning*, pp. 5170–5194, 2025.

Junsu Kim, Hoseong Cho, Jihyeon Kim, Yihalem Yimolal Tiruneh, and Seungryul Baek. SDDGR: Stable diffusion-based deep generative replay for class incremental object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 28772–28781, 2024.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos Policy: Fine-tuning video models for visuomotor control and planning. In *International Conference on Learning Representations*, 2026a.

Seungku Kim, Suhyeok Jang, Byungjun Yoon, Dongyoung Kim, John Won, and Jinwoo Shin. RoboCurate: Harnessing diversity with action-verified neural trajectory for robot learning. *arXiv preprint arXiv:2602.18742*, 2026b.

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.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026.

Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. In *Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.074.

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

Yue Liao, Pengfei Zhou, Siyuan Huang, Donglin Yang, Shengcong Chen, Yuxin Jiang, Yue Hu, Si Liu, Jianlan Luo, Liliang Chen, Shuicheng Yan, Maoqing Yao, and Guanghui Ren. Genie Envisioner: A unified world foundation platform for robotic manipulation. In *International Conference on Learning Representations*, 2026.

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 (Datasets and Benchmarks Track)*, 2023.

Huihan Liu, Changyeon Kim, Bo Liu, Minghuan Liu, and Yuke Zhu. Pretrained vision-language-action models are surprisingly resistant to forgetting in continual learning. In *International Conference on Machine Learning*, 2026.

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.

Jiahao Lu, Jiayi Xu, Wenbo Hu, Ruijie Zhu, Chengfeng Zhao, Sai-Kit Yeung, Ying Shan, and Yuan Liu. Track4World: Feedforward world-centric dense 3D tracking of all pixels. In *European Conference on Computer Vision*, pp. 184–204, 2026. doi: 10.1007/978-3-032-37023-5_11.

Arun Mallya and Svetlana Lazebnik. PackNet: Adding multiple tasks to a single network by iterative pruning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7765–7773, 2018. doi: 10.1109/CVPR.2018.00810.

Gurp Nijjer. The world model remembers, the actor forgets: Dream rehearsal for continual model-based RL. *arXiv preprint arXiv:2607.19749*, 2026.

NVIDIA. Cosmos-Predict2: World simulation model for physical AI, 2025. URL `https://github.com/nvidia-cosmos/cosmos-predict2`.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition, July 2026. URL `https://openai.com/index/gpt-5-6/`.

Minting Pan, Wendong Zhang, Geng Chen, Xiangming Zhu, Siyu Gao, Yunbo Wang, and Xiaokang Yang. Continual visual reinforcement learning with a life-long world model. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 146–162, 2025.

Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2): 123–146, 1995. doi: 10.1080/09540099550039318.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, 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*, 11(7):8331–8338, 2026. doi: 10.1109/LRA.2026.3693992.

Kaushik Roy, Akila Dissanayake, Brendan Tidd, and Peyman Moghadam. M2Distill: Multi-modal distillation for lifelong imitation learning. In *IEEE International Conference on Robotics and Automation*, pp. 1429–1435, 2025. doi: 10.1109/ICRA55743.2025.11128857.

Bo-Kai Ruan, Teng-Fang Hsiao, Ling Lo, and Hong-Han Shuai. Is the future compatible? Diagnosing dynamic consistency in world action models. *arXiv preprint arXiv:2605.07514*, 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.

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

James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. Always be dreaming: A new approach for data-free class-incremental learning. In *IEEE/CVF International Conference on Computer Vision*, pp. 9354–9364, 2021. doi: 10.1109/ICCV48922.2021.00924.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. LAMOL: LAnguage MOdeling for Lifelong Language Learning. In *International Conference on Learning Representations*, 2020.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Wei-Cheng Tseng, Gashon Hussein, Yuzhu Dong, Allen Z. Ren, Lucy X. Shi, XuDong Wang, Sergey Levine, Zhaoshuo Li, Jinwei Gu, Florian Shkurti, Ming-Yu Liu, and Quan Vuong. SC3-Eval: Evaluating robot foundation models via self-consistent video generation. *arXiv preprint arXiv:2606.18610*, 2026.

Gido M. van de Ven, Hava T. Siegelmann, and Andreas S. Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications*, 11(1):4069, 2020. doi: 10.1038/s41467-020-17866-2.

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*, pp. 537–544, 2024. doi: 10.1109/ICRA57147.2024.10611129.

Jun Wang, Liang Ding, Shuai Wang, Hongyu Li, Yong Luo, Huangxuan Zhao, Han Hu, and Bo Du. Self-evolving pseudo-rehearsal for catastrophic forgetting with task similarity in LLMs. In *Advances in Neural Information Processing Systems*, 2025.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024. doi: 10.1109/TPAMI.2024.3367329.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer G. Dy, and Tomas Pfister. Learning to prompt for continual learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 139–149, 2022. doi: 10.1109/CVPR52688.2022.00024.

Chenshen Wu, Luis Herranz, Xialei Liu, Yaxing Wang, Joost van de Weijer, and Bogdan Raducanu. Memory replay GANs: Learning to generate new categories without forgetting. In *Advances in Neural Information Processing Systems*, 2018.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, 2024.

Kun Wu, Yichen Zhu, Jinming Li, Junjie Wen, Ning Liu, Zhiyuan Xu, and Jian Tang. Discrete policy: Learning disentangled action space for multi-task robotic manipulation. In *IEEE International Conference on Robotics and Automation*, pp. 8811–8818, 2025a. doi: 10.1109/ICRA55743.2025.11127630.

Yuxuan Wu, Guangming Wang, Zhiheng Yang, Tianchen Deng, Maoqing Yao, Brian Sheil, and Hesheng Wang. Continually evolving skill knowledge in vision language action model. *arXiv preprint arXiv:2511.18085*, 2025b.

Yuanqi Yao, Siao Liu, Haoming Song, Delin Qu, Qizhi Chen, Yan Ding, Bin Zhao, Zhigang Wang, Xuelong Li, and Dong Wang. Think small, act big: Primitive prompt learning for lifelong robot manipulation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22573–22583, 2025. doi: 10.1109/CVPR52734.2025.02102.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi Fan, and Joel Jang. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Hongxu Yin, Pavlo Molchanov, Jose M. Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K. Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via DeepInversion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8715–8724, 2020.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, 2020.

William Yue, Bo Liu, and Peter Stone. t-DGR: A trajectory-based deep generative replay method for continual learning in decision making. In *Conference on Lifelong Learning Agents*, pp. 481–497, 2024.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pp. 3987–3995, 2017.

Zexin Zheng, Jia-Feng Cai, Xiao-Ming Wu, Yi-Lin Wei, Yu-Ming Tang, Ancong Wu, and Wei-Shi Zheng. iManip: Skill-incremental learning for robotic manipulation. In *IEEE/CVF International Conference on Computer Vision*, pp. 13890–13900, 2025. doi: 10.1109/ICCV51701.2025.01289.

Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. In *Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.015.

A APPENDIX

Supplementary video. To better understand our work, we supplement an intuitive introduction of R^2-WAM and real-robot experimental results in **`supplementary_video.mp4`**.

A.1 EXPERIMENTAL SETUP

Training and evaluation. Every task is trained for 1,000 steps with a global batch size of 64 and a learning rate of 2×10^{-5}, saving a checkpoint every 250 steps; the checkpoint with the best success on the current task over 16 validation trials is kept, using initial states disjoint from the 50 test trials. Replay data and new-task demonstrations are mixed with equal per-sample weight. All LIBERO methods use the same set of training seeds and the same evaluation scenes.

Backbone and generation. LingBot-VA couples a Wan-family video diffusion transformer with an action branch: a 5.1B-parameter DiT (30 layers, hidden size 3072, FFN 14336) over 48-channel latents produced by a Wan VAE, conditioned on a frozen UMT5-XXL text encoder. On LIBERO it observes two 128×128 views and predicts four latent frames per chunk, i.e. $H = 16$ actions, which is also the granularity at which truncation cuts. Generation is autoregressive across chunks and diffusion-based within a chunk, and we keep the released inference configuration unchanged: a flow-matching sampler with 20 denoising steps for observations and 50 for actions, and classifier-free guidance 5.0 on observations and 1.0 on actions. The one component we change is action normalization: each task's actions are normalized by their own 1st and 99th percentiles instead of a single global range.

Real-robot setup. ① **Platform and data.** As shown in Fig. 3, the robot is a 7-DoF arm with a 20-DoF dexterous hand, observed by a head camera and a wrist camera. Observations are recorded at 30 Hz and resized to 256×320, and the action is the 27-dimensional joint command. We fine-tune the WAM on the 50 demonstrations of each task with the same schedule as in LIBERO. ② **Tasks.** The five tasks share a tabletop with a plate holding a slice of bread, a rack of five ingredients, and a stack of bread, as illustrated in Fig. 8. They differ only in the instructed ingredient and the grasp it demands, so forgetting manifests as picking the wrong ingredient or failing the grasp. ③ **Scoring.** Each trial is scored on four steps independently, one point each: the instructed ingredient is lifted off the rack, it rests on the bread, the top bread is lifted, and the sandwich is covered. The first two steps earn no point if a wrong ingredient is used, whereas the last two are identical across tasks, so a policy that has forgotten the ingredient can still earn up to half the points. The progress rate is the score divided by four, averaged over 20 trials per task, and SR, NBT, and Plasticity are all computed on it in place of the binary success.

A.2 DETAILS OF RECTIFIED GENERATIVE REPLAY

Generation. For each learned task, R^2-WAM samples 40 candidate rollouts from the instruction alone. The generation horizon of a task is set to $\lfloor \bar{L}/16 \rfloor + 2$ action chunks of 16 steps, where $\bar{L}$ is the mean length of its demonstrations, i.e., about two chunks longer than a typical demonstration; the resulting ranges for the four suites are given in Table 5. We use the demonstration length only as a convenient way to set this horizon, and it carries no information about the task beyond a rough duration: in practice the horizon can equally be set from a human estimate of how long the task takes, or by probing the model with a few generated rollouts of the task before replay. Moreover, because completion-aware truncation cuts every rollout at task completion, the horizon only needs to be long enough, and its exact value has little effect on the replay data.

Table 5: Demonstration lengths and generation horizons per suite (range over tasks, in steps).

	Spatial	Object	Goal	LIBERO-10
Mean demonstration length	101–150	133–161	89–204	189–416
Generation horizon	128–176	160–192	112–224	208–432

Completion-aware adaptive truncation. The evaluator is a VLM queried with the task instruction and two parts of visual evidence, with the full prompt given in Fig. 12 of Appendix A.5. Part

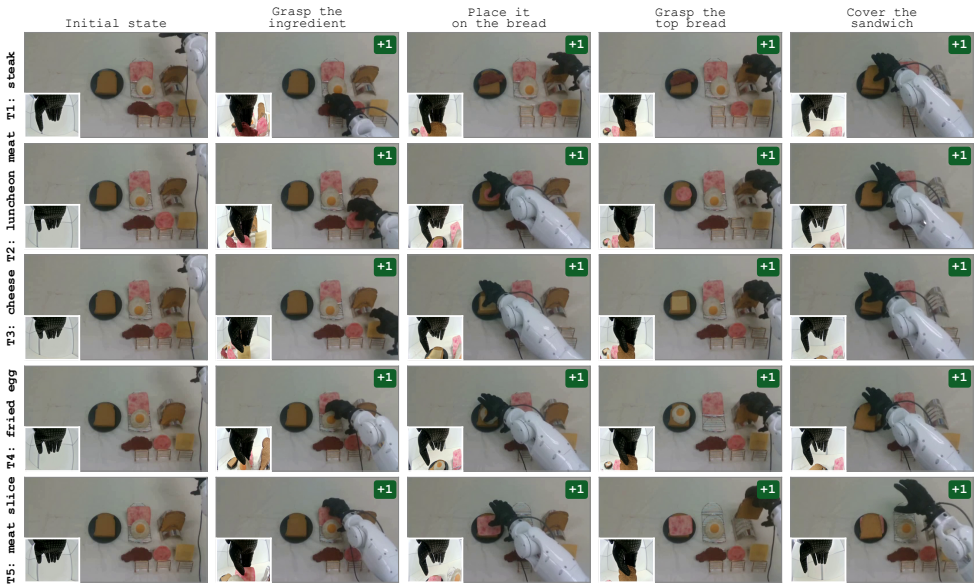


Figure 8: **The five real-world sandwich tasks and their scored steps.** Each row shows one demonstration of a task from the head camera, with the wrist camera inset at the lower left; columns show the initial state and the four scored steps, each worth one point of the four-point progress score. Tasks are learned in the order T1 to T5 and share the same scene, so a policy must attend to the instruction to pick the right ingredient.

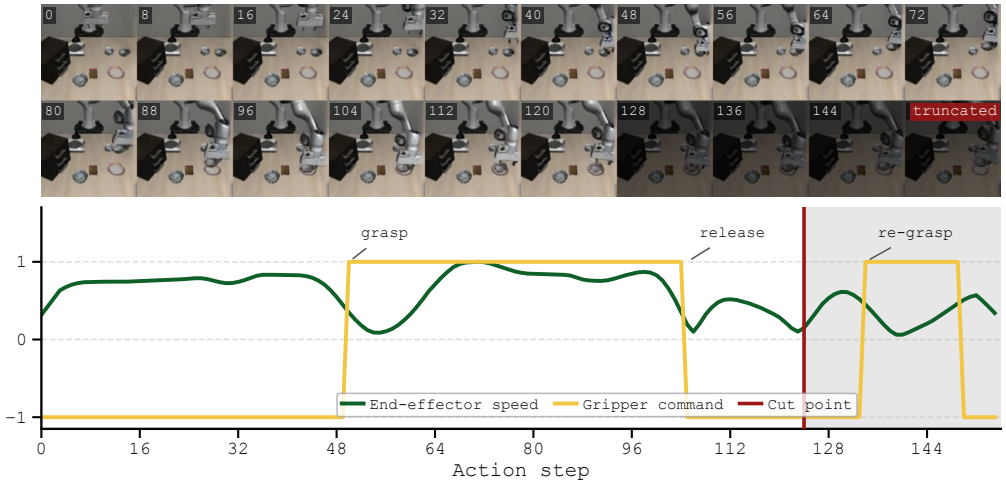


Figure 9: **A truncated rollout for the task "put the bowl on the plate".** Top: frames every 8 steps; shaded frames are removed. Bottom: normalized end-effector speed and gripper command, with the cut point of the evaluator. The placement, the release, and the upward retreat are retained, while the subsequent descent and re-grasping loop are removed.

1 is an in-context example: a generated rollout of the same task in which a human has marked the range of acceptable truncation steps, so that the VLM learns what completion looks like for this task. Part 2 is the candidate rollout to be judged. Instead of the whole video, we uniformly sample frames from the last part of the rollout, every 2 steps over the final 20% or at least 40 steps. Frames at which a motion ends or the gripper releases, as detected from the generated actions, are tagged `[BOUNDARY]` and the others `[SAMPLE]`, since the correct truncation usually falls at a boundary. Every frame pairs the external and the wrist view of one step, and we additionally attach its step index, gripper command, and end-effector velocity, along with a summary of the motion segments detected from the generated actions. Fig. 9 shows an example, where the placement, the release, and the upward retreat are retained while the subsequent re-grasping loop is removed.

Table 6: Per-task cost of R^2-WAM on LIBERO-Spatial (4×H200). Rectification stages are measured on $M = 40$ candidate rollouts of 128–176 steps.

Stage	Time per task
Generating $M = 40$ candidate rollouts	11 min
Completion-aware truncation (40 VLM calls, ≈1.8M input tokens)	3.5 min
Track4World flow estimation	1.5 min
Robot-only simulation, scoring, and top-N selection	2 min
Rectified generative replay in total	18 min
Fine-tuning on the new task (1,000 steps)	1.8 h

Table 7: **Comparison with ReGen-style generative replay** on LIBERO-Spatial. The ReGen-style variant initializes each rollout from a real observation of the task and applies no consistency filtering; both variants are truncated at task completion by the same VLM evaluator.

Variant	SR ↑	NBT ↓	Plast. ↑
ReGen-style replay	68.0	19.15	92.2
R^2-WAM (ours)	**94.0**	**2.90**	**95.7**

Physical-consistency score. Both flows are computed between frames of the truncated rollout: Φ_{sim} by rendering the robot in a robot-only MuJoCo scene driven by the generated actions, and Φ_{pred} by Track4World on the generated video. The comparison is restricted to robot pixels. A pixel is included only if the simulator marks it as part of the moving robot in both frames of the pair. The score therefore measures whether the generated video moves the robot the way the generated actions would, and task objects are not scored. d_{2D} and d_{3D} summarize the disagreement between the two flows in the image plane and in 3D camera space, respectively, as a normalized discrepancy in $[0, 1]$ averaged over all robot pixels and frame pairs of the rollout. We set $\lambda_{\text{2D}} = \lambda_{\text{3D}} = 0.5$ and use them unchanged for all suites and the real robot.

Real-robot variant. For the dexterous hand, the gripper command shown to the evaluator is replaced by the mean distance between the thumb and the other fingertips, mapped to the same open/close scale, and the end-effector velocity is computed from the arm joint commands by forward kinematics; the flow filter uses the head camera only. The prompt is otherwise unchanged.

Cost of rectified generative replay. Table 6 lists the wall-clock time of each stage for one LIBERO-Spatial task on a single node with four H200 GPUs, averaged over the tasks of the chain behind Table 1. Generating and rectifying the replay set of a task takes about 18 minutes, roughly one sixth of fine-tuning on that task, and is incurred once per task since the rectified rollouts are reused in all later stages. Generation scales linearly with the number of candidates M and with the horizon; truncation is bounded by the VLM API rather than by compute.

A.3 Additional Results

Comparison with ReGen-style generative replay. REGEN (Govind et al., 2026) also replays generated rollouts, but initializes each rollout from a real observation of the previous task, which requires retaining those observations. To quantify what this buys, we build a ReGen-style variant of our own pipeline: each rollout is initialized from a real initial observation of the task instead of one generated from the instruction, and no consistency filtering is applied, since REGEN has no counterpart to it. REGEN ends a rollout with the value head of its Cosmos Policy backbone (Kim et al., 2026a), which LingBot-VA does not have; we therefore truncate its rollouts with the same VLM evaluator as R^2-WAM, so that both variants stop at task completion. As shown in Table 7, the ReGen-style variant reaches 68.0% SR on LIBERO-Spatial, far below the 94.0% of R^2-WAM, indicating that retaining real observations to start from does not substitute for regenerating and rectifying rollouts from the instruction alone.

Replay buffer task matrices. Fig. 10 shows the per-task success curves behind Fig. 6. At a buffer size of 1, the deficit of ER comes from a few tasks (T2, T4, and T7) that collapse to 22–42% once later tasks are learned, whereas R^2-WAM degrades mildly on several tasks but never falls below 50% on any of them, which supports the conjecture that a generated rollout is a more stable anchor than

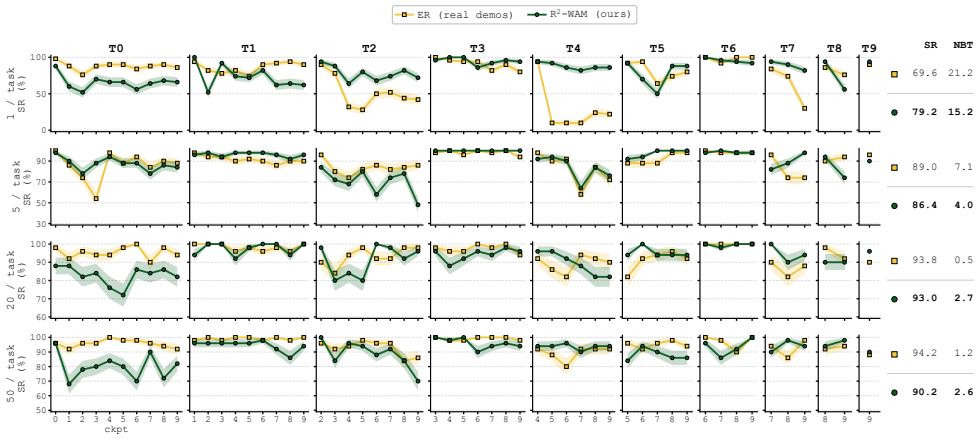


Figure 10: **Task matrices when scaling the replay buffer on LIBERO-Spatial.** Rows are replay buffer sizes of 1, 5, 20, and 50 episodes and columns are the ten tasks. Each panel shows how the success rate of one task evolves as the model continues to learn the following tasks, and the last column lists the final SR and NBT of each row. ER (yellow) replays real demonstrations, and R^2-WAM (green) replays generated rollouts without real data. Shading is the standard error.

a single real demonstration. From 20 episodes onwards, ER stays above 80% on every task, while the remaining gap of R^2-WAM concentrates on a few early tasks, most notably T0, whose success keeps drifting slowly as more tasks are learned.

A.4 Qualitative Real-Robot Rollouts

Fig. 11 shows representative trials behind Table 2, from which we draw three observations. **First**, Seq-FT is highly plastic but forgets, and the forgetting worsens as more tasks are learned, as shown from top to bottom in Fig. 11(a). Right after learning a task, the robot completes all four steps. After one more task is learned, it already skips the steak and completes only the two bread steps. As training continues, the bread steps degrade as well: three tasks later the robot still grasps the bread but releases it off the plate, and four tasks later it completes no step at all. **Second**, ER and R^2-WAM both retain previous tasks, and four tasks later both still complete the first task with the same motion, as shown in Fig. 11(b). **Finally**, both replay methods still lose some precision on previous tasks, and their failures are alike. In all three trials of Fig. 11(c), the robot grasps the instructed ingredient and completes both bread steps, and fails only at placing: the fried egg slips from the hand in transit, for ER and R^2-WAM alike, or the cheese is released at the edge of the bread and slides off. This loss of placement precision is consistent with the task interference discussed in Section 4.2.

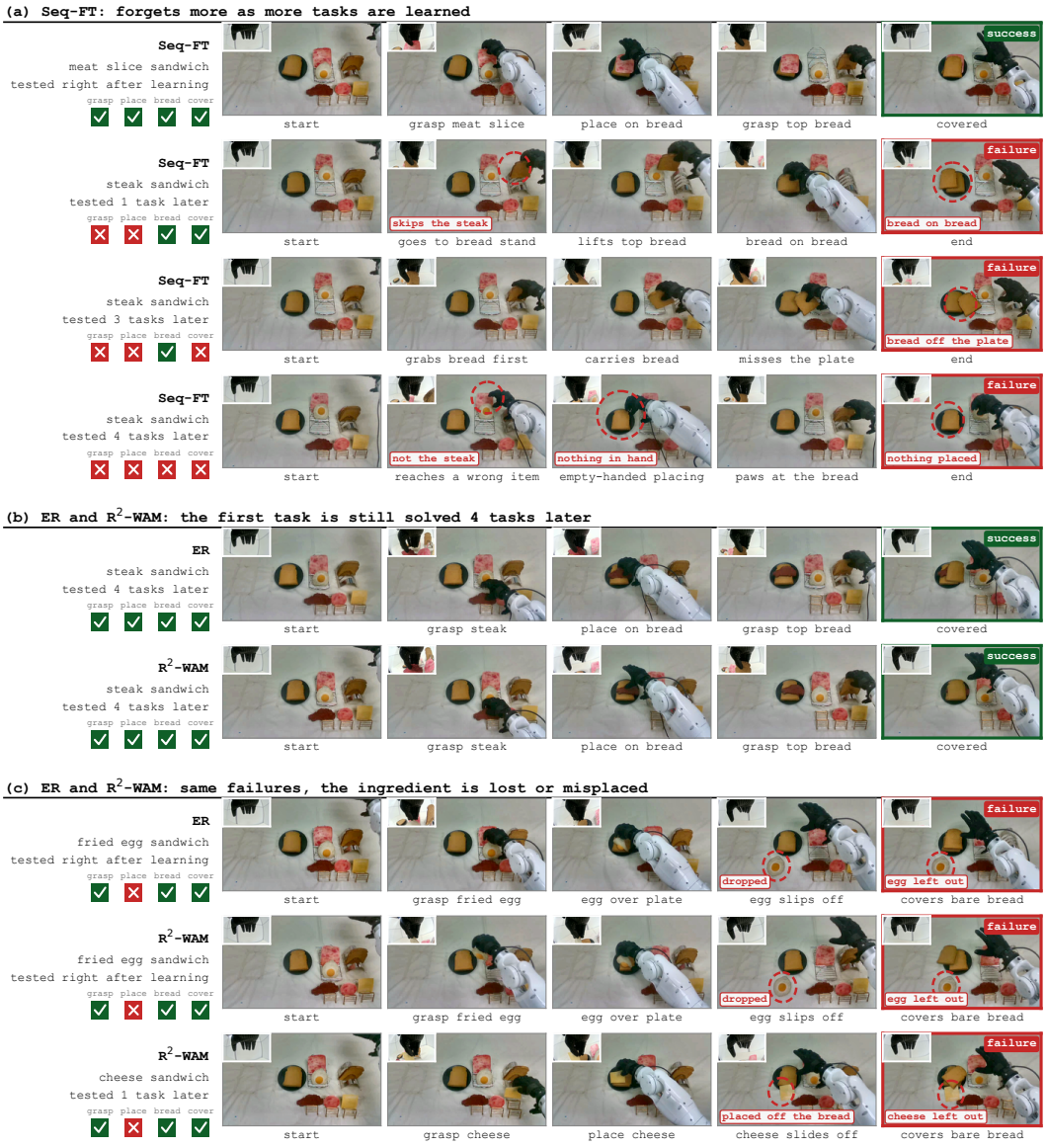


Figure 11: **Qualitative real-robot rollouts.** The robot makes a sandwich with the instructed ingredient in four steps: grasp the ingredient, place it on the bread, grasp the top bread, and cover the sandwich. Each row is one trial from the head camera with the wrist view inset. Its label gives the method, the instruction, and how many further tasks were learned before testing. The four boxes mark which steps the robot completes, and red ellipses mark the error.

A.5 Prompt of the Completion Evaluator

The full prompt of the completion evaluator is reproduced in Fig. 12, instantiated for a LIBERO-Spatial task.

System (system message)

```
You are labelling where a robot manipulation episode should be CUT:
everything after your chosen frame will be deleted from the training
data.  You will be shown still frames in chronological order.  Each
frame is preceded by a line of text giving its absolute STEP index,
```

the GRIPPER command at that step, and the commanded end-effector VELOCITY vel=[x,y,z] (z>0 is upward) with its speed. Gripper scale: dimension 7 of the action vector, in $[-1,+1]$; near $+1$ commands the fingers to close, near -1 to open; a sustained switch from $+1$ to -1 right after an object reaches its target is what a deliberate release looks like. The robot runs at 20 steps per second. Each image shows two views of the same moment: LEFT = external camera, RIGHT = wrist camera. FIRST work out, from the instruction alone, what the world must look like for it to be fully carried out, its GOAL STATE. THEN find where the episode reaches it. FINALLY answer with a CUT POINT inside the acceptable zone: after the motion that completes the task has fully finished, and before the arm clearly moves on to anything else. Answer with JSON only.

User (user message)

TASK GIVEN TO THE ROBOT: "pick up the black bowl between the plate and the ramekin and place it on the plate"

Part 1: Annotated example

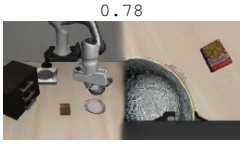
=== PART 1: ANNOTATED EXAMPLE --- one complete episode from the SAME generator and SAME task, whose correct cut zone was marked by a human. Frames are sparse early and denser as the cut zone approaches, and the discarded junk after it is shown too. Each frame is labelled task still in progress, CUT ZONE (any cut inside this zone is correct: it never severs the completing motion and never keeps off-task motion), or DISCARDED (junk after the cut). Use it to calibrate what completion, the cut zone, and junk look like for this task on generated data.
--- ANNOTATED EXAMPLE: correct cut zone = steps 92-100

EXAMPLE step 27 gripper -0.99 | speed 0.97



task still in progress

EXAMPLE step 66 gripper +1.00 | speed 0.78



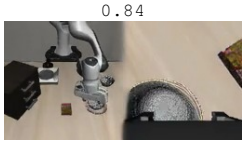
task still in progress

EXAMPLE step 96 gripper -0.98 | speed 0.64



*** CUT ZONE: any cut in this zone is correct ***

EXAMPLE step 106 gripper -0.98 | speed 0.84



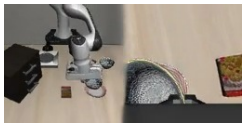
DISCARDED: junk after the cut

4 of the 12 example frames are shown; the line under each frame is the label the prompt attaches to it.

Part 2: Candidate

=== PART 2: CANDIDATE --- the episode you must label, produced by a world model that always runs a fixed number of steps with no stopping criterion, so it usually keeps acting after the task is already complete: hovering, drifting, or picking the object up again. The full episode is 128 steps (6.4 s); the task is known to be still unfinished early on, so steps 0--51 are not shown. The 39 candidate frames below are in chronological order and are of two kinds, marked in the line above each frame: [BOUNDARY], a motion turning point detected from the robot's own action stream (the moment one continuous motion has just ended, a gripper release, or the final frame); and [SAMPLE], a uniform time sample between boundaries. A motion segment table from the same action stream is given right before the frames. Your answer must be one of the candidate steps below.
MOTION SEGMENTS detected from the robot's commanded end-effector velocities:
steps 48-76: mean direction [x+0.86, y+0.23, z-0.45], peak speed 1.08
steps 81-92: mean direction [x+0.29, y+0.00, z+0.96], peak speed 0.75
...

CANDIDATE [SAMPLE] step 66 gripper +1.00 | speed 0.79



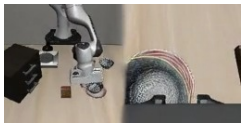
carrying the bowl

CANDIDATE [BOUNDARY] step 78 gripper −0.98 | speed 0.24



release

CANDIDATE [BOUNDARY] step 82 gripper −0.98 | speed 0.66



retreat

CANDIDATE [BOUNDARY] step 126 gripper +0.97 | speed 0.50



re-grasping (final frame)

4 of the 39 candidate frames are shown; the italic notes under them describe what happens and are ours.

How to decide

```
=== HOW TO DECIDE
STEP 1. Derive the goal state from the instruction before looking
for an answer: list every condition that must hold at the end, in the
order they must happen; the task is finished only when the last one
holds. Do not assume a sub-goal involves the gripper at all.
STEP 2. Decide when each sub-goal is satisfied. A PLACEMENT sub-goal
is satisfied only when the object rests at its target, the printed
gripper values have switched from closing to opening, and the object
has fully SETTLED; the instant of release is not the end. For a
NON-HELD sub-goal (pushing, sliding, turning) the gripper values are
irrelevant: follow the interaction motion to its TURNING POINT, the
last frame before the arm switches to a different kind of motion. A
sub-goal is satisfied at the LAST successful attempt, not the first
try.
STEP 3. Pick end_step from the evidence above: the annotated
example, so that your episode is cut at the analogous place and keeps
no frames resembling its discarded ones; the numbers, because a motion
lasts exactly as long as its velocity direction persists, so never
cut while that direction still holds; and the candidate types, since
the correct cut is usually the [BOUNDARY] at or shortly after the
completing motion finishes. Tie-break: when hesitating between two
adjacent candidates, always take the LATER one. Self-check: if the
last sub-goal is a placement and the gripper at your chosen step still
commands closing, your answer cuts before the release and is wrong.
If some sub-goal is never visibly satisfied in any offered frame, set
accomplished=false and end_step to the last candidate step.
```

Output (JSON only)

```
{"goal_state": "<what the scene must look like, <= 25 words>",
"subgoals": [{"what": "<one condition>", "satisfied_at_step":
<candidate step or null>}],
"end_step": <one of the candidate steps>, "accomplished":
<true|false>,
"gripper_open_at_end": <true|false>, "failed_grasp_attempts": <int>,
"why": "<= 20 words"}
```

Figure 12: **The prompt of the completion evaluator.** Shown for a LIBERO-Spatial task, with four frames of the in-context example and of the candidate; some non-essential content is omitted, and shaded gray text is our annotation.

A.6 Task Sequences

The task sequence of every benchmark is listed below in learning order. Within each LIBERO suite, tasks are ordered lexicographically by their language instructions, so the order is fixed by the benchmark rather than chosen by us; the five real-robot tasks follow a fixed order that is identical for all compared methods. LIBERO tasks are indexed from T0 following the benchmark and real-robot tasks from T1.

LIBERO-Spatial Task Order

0. Pick up the black bowl between the plate and the ramekin and place it on the plate.
1. Pick up the black bowl from table center and place it on the plate.
2. Pick up the black bowl in the top drawer of the wooden cabinet and place it on the plate.
3. Pick up the black bowl next to the cookie box and place it on the plate.
4. Pick up the black bowl next to the plate and place it on the plate.
5. Pick up the black bowl next to the ramekin and place it on the plate.
6. Pick up the black bowl on the cookie box and place it on the plate.
7. Pick up the black bowl on the ramekin and place it on the plate.
8. Pick up the black bowl on the stove and place it on the plate.
9. Pick up the black bowl on the wooden cabinet and place it on the plate.

LIBERO-Object Task Order

0. Pick up the alphabet soup and place it in the basket.
1. Pick up the bbq sauce and place it in the basket.
2. Pick up the butter and place it in the basket.
3. Pick up the chocolate pudding and place it in the basket.
4. Pick up the cream cheese and place it in the basket.
5. Pick up the ketchup and place it in the basket.
6. Pick up the milk and place it in the basket.
7. Pick up the orange juice and place it in the basket.
8. Pick up the salad dressing and place it in the basket.
9. Pick up the tomato sauce and place it in the basket.

LIBERO-Goal Task Order

0. Open the middle drawer of the cabinet.
1. Open the top drawer and put the bowl inside.
2. Push the plate to the front of the stove.
3. Put the bowl on the plate.
4. Put the bowl on the stove.
5. Put the bowl on top of the cabinet.
6. Put the cream cheese in the bowl.
7. Put the wine bottle on the rack.
8. Put the wine bottle on top of the cabinet.
9. Turn on the stove.

LIBERO-10 Task Order

0. Pick up the book and place it in the back compartment of the caddy.
1. Put both moka pots on the stove.
2. Put both the alphabet soup and the cream cheese box in the basket.
3. Put both the alphabet soup and the tomato sauce in the basket.
4. Put both the cream cheese box and the butter in the basket.
5. Put the black bowl in the bottom drawer of the cabinet and close it.
6. Put the white mug on the left plate and put the yellow and white mug on the right plate.
7. Put the white mug on the plate and put the chocolate pudding to the right of the plate.
8. Put the yellow and white mug in the microwave and close it.
9. Turn on the stove and put the moka pot on it.

Real-Robot Task Order

1. Make a sandwich with steak.
2. Make a sandwich with luncheon meat.
3. Make a sandwich with cheese.
4. Make a sandwich with a fried egg.
5. Make a sandwich with a meat slice.