ACCRUE: Continual Off-Policy Reinforcement Learning for Robot Foundation Models

Changyeon Kim[1,2] Kimin Lee[1] Yashraj Narang[2] Yijie Guo[2]
[1]KAIST [2]NVIDIA

***Abstract*— Reinforcement learning (RL) post-training improves robot foundation models (RFMs) beyond their demonstrations, but real-world deployment demands more: learning new tasks efficiently while retaining previously acquired behaviors. Existing continual RL methods on top of VLAs mitigate forgetting but rely on on-policy updates with limited forward transfer, whereas off-policy methods improve single-task sample efficiency without addressing learner persistence across tasks. We propose ACCRUE, a continual off-policy RL framework that carries lightweight actor and critic networks across tasks, allowing each stage to reuse behavioral and value knowledge acquired through prior RL. The inherited critic adapts only on current-task transitions and guides current-task actor learning, while prior behavior is preserved by distilling the actor toward cached targets on a small number of retained episodes. Fixed visual features keep the inherited learner and its replay compatible with base-policy updates. Across contact-rich assembly sequences in simulation and on real-robot hardware, ACCRUE accelerates subsequent-task learning with near-zero forgetting and significantly reduces real-robot training time by 31.0% to 68.5%.**

I. INTRODUCTION

Robot foundation models (RFMs), including vision-language-action models (VLAs) [1]–[4] and world action models (WAMs) [5]–[8], have demonstrated strong performance across a wide range of manipulation tasks. Despite these generalization capabilities, RFMs remain imperfect at deployment, where robots encounter tasks and conditions not covered by pretraining. Reinforcement learning (RL) post-training can close this gap by improving a model beyond its demonstrations through online interaction [9]–[13]. Most prior work, however, treats post-training as a one-time procedure for a single target task. A robot deployed in the real world must instead learn to perform new tasks throughout its operation, often from only a few demonstrations and without access to future tasks in advance. Robotic assembly illustrates this need: new parts and product variants require robots to adapt their contact behaviors while retaining skills for previously encountered assemblies. Restarting RL for every task repeatedly pays the full interaction cost and fails to exploit knowledge acquired during earlier RL stages. Carrying the learner across tasks introduces the risk of forgetting previously acquired behaviors. Practical continual post-training must therefore *reduce* the interaction required for each new task without sacrificing capabilities acquired from previous tasks (Fig. 1).

Recent methods use RL to adapt VLAs across sequential tasks and mitigate forgetting [14]–[16], but provide limited forward transfer. Whereas on-policy updates require fresh current-policy rollouts, off-policy learning can reuse previously collected transitions to improve interaction efficiency. Existing approaches such as RL Token [12] and EXPO-FT [13], which attach lightweight actor and critic networks to frozen RFMs, improve single-task sample efficiency but do not address learner persistence across tasks.

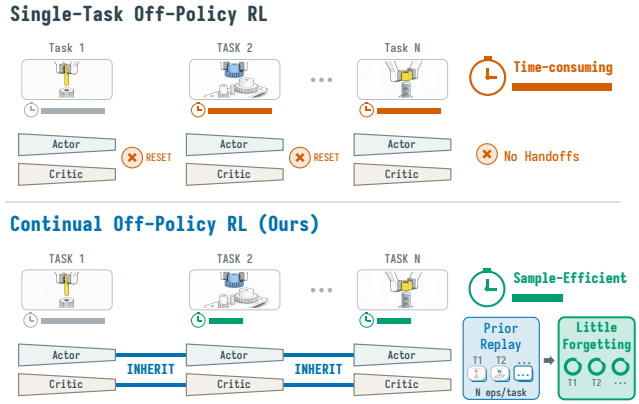


Fig. 1. **Single-task vs Continual off-policy RL.** Single-task RL resets the actor and critic for each task, requiring repeated online interactions. Our method inherits both networks and uses small prior-task replay to accelerate learning while retaining prior performance.

Directly extending these approaches across tasks introduces practical challenges. Retaining the full interaction history becomes increasingly costly as tasks accumulate, motivating a small replay memory for preserving prior behavior. With limited prior-task coverage, adapting an inherited critic to a new task can make its estimates unreliable on earlier tasks; using those estimates to drive actor updates on prior-task samples can then erode previously learned behavior. Meanwhile, updating the base RFM can change its internal representations and action proposals, potentially shifting the inputs consumed by the inherited networks.

To this end, we propose ACCRUE (*Actor–Critic Carryover for continual off-policy Reinforcement learning Using retained Experience*), a continual off-policy RL framework that inherits a lightweight actor and critic across tasks to amortize prior RL experience. The inherited actor and critic transfer behavioral and value priors, respectively, enabling subsequent tasks to be learned with fewer environment interactions. Starting from a pretrained RFM, we fine-tune the base policy at each stage on demonstrations observed so far and hold it fixed during RL. At each stage, the inherited critic is adapted only to the current-task transitions. We deliberately exclude prior-task samples from critic updates because the small number of

*Correspondence: changyeon.kim@kaist.ac.kr

retained episodes may not provide sufficient coverage for reliable value estimation on all prior tasks. The adapted critic guides actor improvement on the current task but is not used to update the actor on prior tasks. Instead, ACCRUE preserves prior behavior by distilling the actor toward cached outputs of the preceding-stage actor on the last N episodes retained from each completed task (20–100 episodes per task in main experiments). This asymmetric update retains the benefits of actor–critic inheritance without relying on the critic to preserve prior-task behavior. A fixed external visual encoder provides consistent visual features across base-policy updates, while reference-action conditioning allows the lightweight learner to refine proposals from the current base policy.

We showcase ACCRUE on contact-rich assembly sequences in simulation and on real-robot hardware, with experiments covering VLA and WAM base policies and different off-policy RL algorithms. In simulation, ACCRUE accelerates subsequent-task learning while limiting forgetting of prior tasks. Across FORGE and AutoMate, it improves continual AUC by up to 21.0% and reduces negative backward transfer by up to 70.0% compared with the strongest off-policy continual baseline. Ablations and baseline comparisons show that critic inheritance supports forward transfer and that separating current-task RL from prior-task actor preservation improves retention under limited replay. On two multi-stage real-robot sequences, ACCRUE reduces robot RL interaction time for subsequent tasks by 31.0%–68.5% compared with training a new actor–critic for each task, while maintaining approximately 97% average success on prior tasks.

II. RELATED WORK

A. RL Post-Training for RFMs

RL post-training improves RFMs beyond demonstration-only supervision using task rewards and deployment experience. One line of work directly updates the VLA or its action head through policy-gradient learning, typically requiring fresh current-policy rollouts and substantial parallel interaction [11,17]. Other approaches use offline value learning, task-specific residual policies, or RL-generated data to improve a generalist policy [9,10,18]. Most closely related to our architecture, lightweight off-policy methods refine a frozen policy through action- or latent-space actor–critic learning [12,13,19,20], but these methods focus on improving sample efficiency only on individual tasks. Our work considers the complementary problem of carrying a lightweight off-policy learner across sequentially arriving tasks while the underlying RFM is repeatedly updated with new demonstrations.

B. Continual Learning for Robotics

Continual robot learning [21,22] has largely focused on preserving demonstrated behavior through experience replay [23,24], hierarchical skill libraries [22], task-specific adapters [25], or parameter merging [26]. Beyond imitation learning, recent continual VLA methods carry policy parameters across tasks using on-policy RL [14,16] or demonstration-derived reinforcement fine-tuning [15]. The former require fresh interaction at each stage, whereas the latter do not transfer value knowledge acquired through environment interaction. Concurrent work CIDER learns a shared visuomotor policy from task-specific data without a pretrained RFM, combining current-task off-policy human-in-the-loop RL with prior-task actor distillation [27]. In contrast, ACCRUE builds on manipulation skills already learned by a pretrained VLA or WAM and uses RL to improve its proposed actions. By retaining the actor and critic across tasks, ACCRUE reuses earlier RL experience even as the base model is further fine-tuned with new demonstrations.

III. PRELIMINARIES

A. Robot Foundation Models

We build on a base policy π_{base}, a robot foundation model (RFM) pretrained on large-scale robot data, such as VLA [1]–[4] or WAM [5]–[8]. Given multi-view images $\mathbf{I}_t$, proprioception $\mathbf{q}_t$, and a language instruction ℓ, the policy outputs an H-step action chunk $\tilde{\mathbf{a}}_t = (\tilde{a}_t, \ldots, \tilde{a}_{t+H-1}) \in \mathbb{R}^{H\times d}$, where d denotes the dimension of the action space [28,29]. We use the same horizon H for prediction and execution in all experiments. We access π_{base} only through this input–output interface, so our framework is agnostic to the choice of internal architecture in π_{base}.

B. Problem Setup

We consider a robot that acquires K tasks sequentially, one task per stage. Each task T^k is a finite-horizon Markov decision process that shares its state space, action space, and discount γ with the other tasks, but may differ in its transition function, reward function, initial-state distribution, and horizon. Reward function r_t is sparse and emitted upon task success. We follow a task-incremental continual-learning setting [30], where task boundaries and identities are known.

At stage k, a small expert demonstration set $\mathcal{D}^k$ (1–50 trajectories in our experiments) becomes available for T^k. The base policy is fine-tuned on all demonstrations observed up to that stage, $\mathcal{D}^{1:k} = \bigcup_{i=1}^{k} \mathcal{D}^i$, yielding π^k_{base}. Starting from π^k_{base}, the agent performs RL using new interactions collected exclusively from the current task T^k and stored in the current-task replay buffer $\mathcal{B}^k$. The learner additionally uses a bounded prior-task replay buffer $\mathcal{B}^{<k}$, which contains a subset of online experience retained from earlier RL stages. Training on $\mathcal{B}^k$ and $\mathcal{B}^{<k}$ produces the adapted policy π^k. Unlike protocols [14,15] that fine-tune the base policy on the complete task suite before continual RL begins, our setup assumes neither access to future demonstrations nor a fixed base policy throughout the sequence. It therefore captures repeated post-training as new task requirements and demonstrations arrive over time.

After stage k, π^k is evaluated on all tasks encountered so far, with the corresponding language instruction provided for each task. The objective is to maximize the average return across these tasks:

$$\max_{\pi^k} J^k(\pi^k) = \frac{1}{k}\sum_{i=1}^{k} \mathbb{E}_{\tau\sim(\pi^k,T^i)}\left[\sum_{t=0}^{L_\tau}\gamma^t r_t^i\right],$$

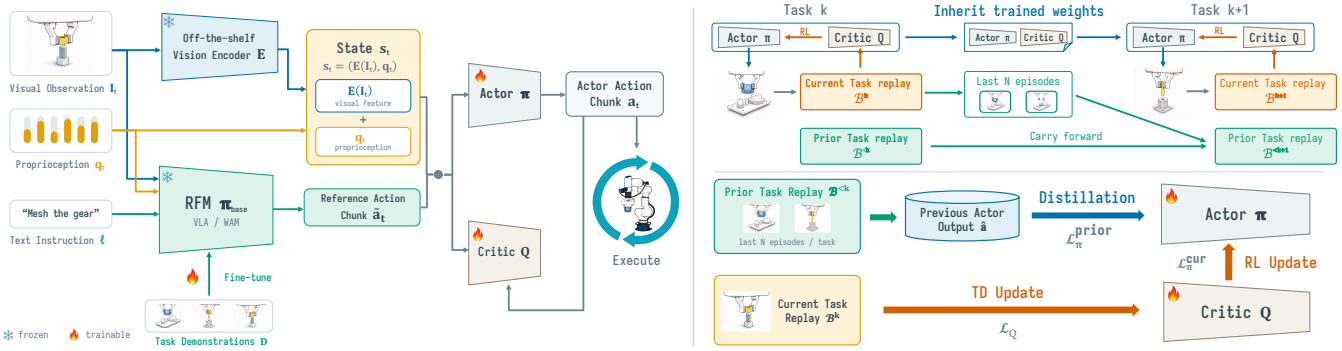


Fig. 2. **Overview of ACCRUE.** (Left) Base policy is fine-tuned on demonstrations observed up to current stage, held fixed, and proposes reference action chunks that actor and critic networks refine through off-policy RL on frozen encoder features. (Right Top) Both networks are carried across stages through a handoff, independent of the base policy, with small fraction of online replay buffer. (Right Bottom) While updating the current task with RL objectives, earlier tasks are preserved by distilling the actor toward cached targets on a small retained memory.

where L^τ denotes the length of trajectory τ. Although new training interactions can be collected only from T^k, the learner should exploit knowledge acquired during earlier RL stages to adapt efficiently to the current task while preserving performance on previous tasks.

C. Action-Chunk Reinforcement Learning

Within stage k, we improve performance on the current task T^k using an off-policy actor–critic algorithm [31,32] trained on the current-task replay buffer $\mathcal{B}^k$. Let s_t denote the state available at timestep t. The actor $\pi_\theta(\cdot \mid s_t)$ outputs an H-step action chunk $\mathbf{a}_t = (a_t, \ldots, a_{t+H-1})$, which is executed in full before the next policy query. The critic $Q_\psi(s_t, \mathbf{a}_t)$ takes the state and the full action chunk as inputs to estimate its value [12,13,18,33]. We define $R_t^{(H)} = \sum_{h=0}^{H-1} \gamma^h r_{t+h}$ as the H-step discounted reward sum collected during chunk execution. The chunk value is estimated by the H-step backup

$$Q_\psi(s_t, \mathbf{a}_t) \approx R_t^{(H)} + \gamma^H \, \mathbb{E}_{\mathbf{a}' \sim \pi_\theta(\cdot \mid s_{t+H})}[Q_\psi(s_{t+H}, \mathbf{a}')] \, .$$

Relative to single-action backups, chunk-level backups reduce the nominal credit-assignment path by a factor of H.

IV. METHOD

We introduce ACCRUE, a simple but effective continual off-policy RL framework for RFMs (Fig. 2). As new task demonstrations arrive at each stage, the base policy is fine-tuned on demonstrations from all tasks encountered so far and then held fixed throughout the RL phase of that stage. The lightweight actor and critic are conditioned on visual features from a fixed external encoder and reference action chunks proposed by the current base policy (Sec. IV-A). The fixed encoder keeps the visual representation unchanged across base-policy updates. For stage $k > 1$, the actor and critic are initialized from the preceding checkpoint (Sec. IV-B). With limited replay, the critic may no longer accurately evaluate actions on earlier tasks. We therefore train the critic only on current-task transitions and preserve prior behavior by matching the current actor's outputs to those of the preceding-stage actor on retained samples (Sec. IV-C).

A. Actor–Critic Inputs

To carry the actor and critic across stages, their visual inputs must keep a consistent representation as the base policy is updated. We therefore use an off-the-shelf visual encoder E that is independent of the base policy and remains frozen across stages. We form the actor–critic input from the encoder output $E(\mathbf{I}_t)$ and robot proprioception $\mathbf{q}_t$ as $s_t = \big(E(\mathbf{I}_t), \, \mathbf{q}_t\big)$.

The actor and the critic apply their own lightweight trainable projector to s_t. The projectors are parameterized as part of θ and ψ, respectively. The replay buffer B^k stores s_t rather than the outputs from these trainable projectors. Freezing E keeps the visual features of every stored transition in the same representation across stages, allowing the inherited projectors to process inputs s_t collected at earlier stages.

The input s_t does not explicitly encode the language instruction. The known task identity determines the language instruction ℓ^k provided to the base policy, which produces the reference action chunk $\tilde{\mathbf{a}}_t$. Rather than encoding the instruction again in the actor and critic, we condition both networks on the instruction-conditioned reference to obtain $\mathbf{a}_t \sim \pi_\theta(s_t, \tilde{\mathbf{a}}_t)$ and $Q_\psi\big(s_t, \mathbf{a}_t, \tilde{\mathbf{a}}_t\big)$.

Because the base policy already encodes task semantics and provides a task-level action proposal, the lightweight actor learns a local refinement conditioned on $\tilde{\mathbf{a}}$ rather than reconstructing the full skill. Continual learning therefore needs to retain only the RL-acquired refinement, reducing the amount of behavior that must remain stable in the shared actor. Moreover, related tasks can share refinement and value structure, allowing the inherited actor and critic to provide useful initializations for subsequent tasks. Removing reference conditioning increases forgetting at a similar final success rate (Table IV). Unlike RL Token [12], which conditions only the actor on $\tilde{\mathbf{a}}$, we condition the critic as well, because the action value is task-dependent and the reference provides task contexts. To keep the actor from merely copying the reference, we apply reference action dropout [12] during training, replacing $\tilde{\mathbf{a}}$ with zeros for a random fraction p of each batch for both networks.

B. Current-Task Off-Policy RL

Actor and critic network parameters are randomly initialized only at the first stage. For each stage $k > 1$, we initialize the actor–critic pair from the checkpoint of stage $k-1$, transferring a behavioral initialization through the

actor and a value initialization through the critic. The current-task buffer $\mathcal{B}^k$ is seeded with warmup episodes collected by executing the base policy π^k_{base}. The actor then interacts with the environment, and the actor–critic pair is updated on transitions sampled from $\mathcal{B}^k$.

The actor objective is to maximize the value estimated by the critic and remain close to the reference chunk,

$$\mathcal{L}^{\text{cur}}_{\pi}(\theta;\mathcal{B}^k) = \mathbb{E}_{\substack{(s,\tilde{\mathbf{a}})\sim\mathcal{B}^k \\ \mathbf{a}\sim\pi_\theta(s,\tilde{\mathbf{a}})}}\left[-Q_\psi(s,\mathbf{a},\tilde{\mathbf{a}}) + \beta\|\mathbf{a}-\tilde{\mathbf{a}}\|_2^2\right], \quad (1)$$

where β penalizes deviation from the reference action [12], reducing the learning to local refinement of the base policy's action distribution rather than exhaustive search over Hd-dimensional action spaces.

The critic is trained with off-policy temporal-difference learning. For a transition $(s_t, \mathbf{a}_t, \tilde{\mathbf{a}}_t, R_t^{(H)}, s_{t+H}, \tilde{\mathbf{a}}_{t+H})$ sampled from $\mathcal{B}^k$, we write the common chunk-level target for value bootstrapping, omitting learner-specific terms:

$$\hat{Q} = R_t^{(H)} + \gamma^H Q_{\psi'}(s_{t+H}, \mathbf{a}_{t+H}, \tilde{\mathbf{a}}_{t+H}), \quad (2)$$

where ψ' denotes the target-critic parameters. The next action chunk $\mathbf{a}_{t+H}$ is sampled from the actor for stochastic learners or formed using the target-action rule for deterministic learners. In implementation, the critic predicts a categorical return distribution over a fixed support, and Q_ψ denotes its expected value. We train the critic with cross-entropy loss [34], which we found more stable under sparse reward.

C. Continual Operation

At the end of stage $k-1$, ACCRUE transfers the actor–critic parameters $(\theta^{k-1}, \psi^{k-1})$ and $\mathcal{B}^{<k}$, a memory of retained experience, to stage k. The base policy is separately fine-tuned on $\mathcal{D}^{1:k}$ to obtain π^k_{base} and then held fixed during stage-k RL. Fine-tuning on accumulated demonstrations is intended to retain the base policy's earlier-task competence. Our continual mechanism aims to preserve the actor's RL-acquired refinements while the actor and critic adapt to the current task.

ACCRUE retains the last N training episodes of each completed task in a pooled memory $\mathcal{B}^{<k}$, giving $(k-1)N$ stored episodes at stage k. For a transition j collected at stage $i < k$, $\tilde{\mathbf{a}}_j$ is the action chunk generated by π^i_{base} at collection time and is retained unchanged. At each stage $k > 1$, each actor update uses a 1:1 ratio of current-task and prior-task samples, drawn in separate minibatches from $\mathcal{B}^k$ and $\mathcal{B}^{<k}$ for their respective actor losses. Before training stage k, we evaluate a frozen copy of the preceding-stage actor (the teacher $\pi_{\theta^{k-1}}$) on each retained transition to provide targets for the current actor (the student π_{θ^k}). The actor may be deterministic or stochastic, depending on the off-policy learner. For distillation, we denote its deterministic evaluation output, obtained without action sampling or exploration noise, by $\bar{\pi}_\theta(s,\tilde{\mathbf{a}})$ and cache

$$\hat{\mathbf{a}}_j^{k-1} = \bar{\pi}_{\theta^{k-1}}(s_j, \tilde{\mathbf{a}}_j), \qquad j \in \mathcal{B}^{<k}. \quad (3)$$

These targets remain fixed throughout stage k, and the teacher copy is discarded after caching. Behavioral targets are refreshed from the preceding-stage actor at each stage, while reference chunks retain their original values, keeping teacher and student at the same previously observed inputs $(s_j, \tilde{\mathbf{a}}_j)$.

Algorithm 1 One stage of ACCRUE

Require: base policy π^k_{base}, frozen visual encoder E, chunk length H, handoff $(\theta^{k-1}, \psi^{k-1}, \mathcal{B}^{<k})$

Initialize π_θ, Q_ψ from θ^{k-1}, ψ^{k-1}
Cache $\hat{\mathbf{a}}_j^{k-1} = \bar{\pi}_{\theta^{k-1}}(s_j, \tilde{\mathbf{a}}_j)$ for $j \in \mathcal{B}^{<k}$ ▷ (3)
Collect warmup rollouts from π^k_{base}
for $t = 0, H, 2H, \ldots$ **do**
  $s_t \leftarrow (E(\mathbf{I}_t), \mathbf{q}_t)$, $\tilde{\mathbf{a}}_t \sim \pi^k_{\text{base}}(\cdot \mid \mathbf{I}_t, \mathbf{q}_t, \ell^k)$
  Execute $\mathbf{a}_t \sim \pi_\theta(\cdot \mid s_t, \tilde{\mathbf{a}}_t)$ and observe $R_t^{(H)}$, s_{t+H}
  Store $(s_t, \mathbf{a}_t, \tilde{\mathbf{a}}_t, R_t^{(H)}, s_{t+H}, \tilde{\mathbf{a}}_{t+H})$ in $\mathcal{B}^k$
  Update ψ on $\mathcal{B}^k$ ▷ (2)
  Update θ on $\mathcal{B}^k \cup \mathcal{B}^{<k}$ ▷ (5)
end for
return last N episodes from $\mathcal{B}^k$, handoff (θ^k, ψ^k)

At stage k, TD updates on $\mathcal{B}^k$ adapt the inherited critic to T^k. Although we could also replay $\mathcal{B}^{<k}$ for critic learning, only N retained episodes per prior task may not reliably preserve its prior-task value estimates. Applying the actor RL objective to these $\mathcal{B}^{<k}$ samples could therefore turn inaccurate critic estimates into drift in previously learned behavior. To avoid this risk, we apply critic TD and the actor RL objective only to $\mathcal{B}^k$, while using $\mathcal{B}^{<k}$ only to match the current actor to the preceding-stage actor at retained inputs,

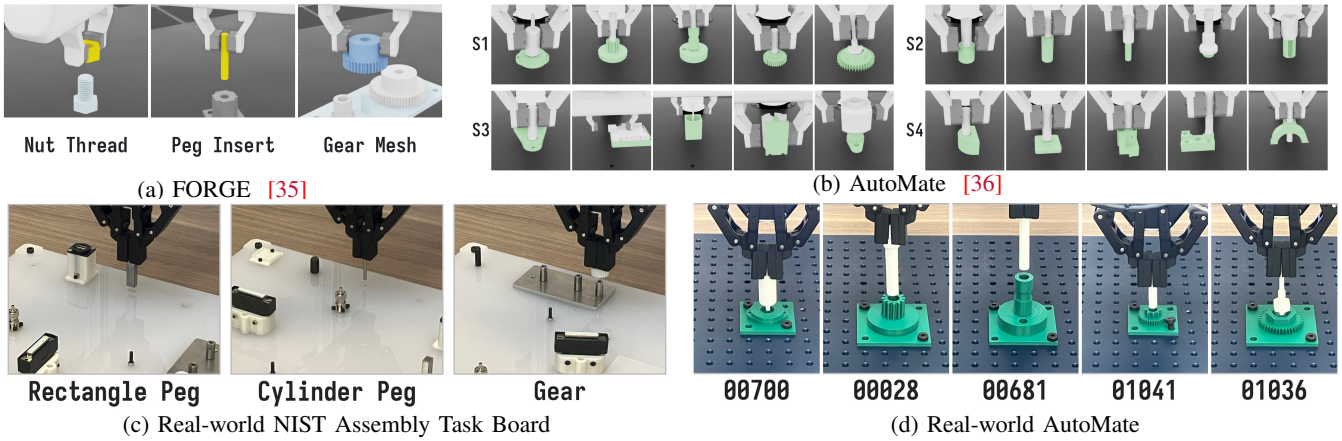
$$\mathcal{L}^{\text{prior}}_{\pi}(\theta;\mathcal{B}^{<k}) = \mathop{\mathbb{E}}_{j\sim\mathcal{B}^{<k}}\left[\left\|\pi_\theta(s_j,\tilde{\mathbf{a}}_j) - \hat{\mathbf{a}}_j^{k-1}\right\|_2^2\right], \quad (4)$$

$$\mathcal{L}_\pi(\theta;k) = \mathcal{L}^{\text{cur}}_{\pi}(\theta;\mathcal{B}^k) + \lambda\,\mathcal{L}^{\text{prior}}_{\pi}(\theta;\mathcal{B}^{<k}), \quad (5)$$

where λ weights prior-task distillation. Thus, we do not use prior-task replay for critic updates or for the actor objective in Eq. (1). Distillation constrains actor outputs at the stored $(s_j, \tilde{\mathbf{a}}_j)$ inputs. Current-task interaction and deployment use fresh references from π^k_{base}, which may differ even when the base policy retains earlier-task competence. We therefore evaluate prior-task retention after each stage with the updated base policy and actor together, testing whether the preserved behavior remains effective with current references. Algorithm 1 summarizes one stage of ACCRUE.

V. EXPERIMENTS

Our experiments ask three questions. 1) Does ACCRUE retain earlier assembly skills once the base policy has been updated for a later stage? 2) Does inheriting the lightweight actor and critic accelerate a new stage? And 3) Which of ACCRUE's components is responsible for each effect? Sec. V-A describes what is held common across platforms. Sec. V-B and Sec. V-C address the first two questions in simulation and on real-world DROID [37]. The two settings use different base policies, reinforcement learning algorithms, and interaction budgets, which lets us verify that the findings do not depend on a particular setup. Sec. V-D addresses the third question in simulation, where the stage budget and the task sequence can be held fixed across variants.



(a) FORGE [35] (b) AutoMate [36]

(c) Real-world NIST Assembly Task Board (d) Real-world AutoMate

Fig. 3. **Overview of experimental tasks.** (a) Three contact-rich manipulation tasks in FORGE [35]. Four orderings are evaluated: S1: **Nut Thread** → **Peg Insert** → **Gear Mesh**, S2: **Gear Mesh** → **Peg Insert** → **Nut Thread**, S3: **Peg Insert** → **Gear Mesh** → **Nut Thread**, and S4: **Peg Insert** → **Nut Thread** → **Gear Mesh**. (b) Four five-stage AutoMate sequences in simulation: S1: **00700** → **00028** → **00681** → **01041** → **01036**, S2: **01092** → **00141** → **00255** → **00486** → **00615**, S3: **00731** → **00470** → **00117** → **00726** → **00296**, and S4: **00417** → **00083** → **01079** → **00514** → **01029**. (c) Three insertion task sequence on the NIST Assembly Task Board and (d) Five-stage AutoMate sequence executed on real 3D-printed assets.

A. Experimental Setup

Baselines: We compare continual learners within a shared off-policy actor–critic architecture, alongside reference methods with different learning or storage requirements. In the main simulation comparisons, all RL methods use the same GR00T N1.7 backbone and per-stage environment-step budget. The matched shared actor–critic methods additionally share the cumulative-demonstration fine-tuning protocol of Sec. III-B and the off-policy RL algorithm, and inherit the actor–critic pair across tasks. **Sequential** trains it using only current-task data. **ER** [38] retains the same number of episodes as ACCRUE per task, but applies the RL objective to the retained memory rather than using actor distillation. **EWC** [39] penalizes actor drift from the previous-stage parameters, while **RETAIN** [26] merges the current and previous actor parameters through discounted interpolation. As an external reference, **Simple Recipe** [14] sequentially fine-tunes the VLA directly with on-policy RL. **Single-task RL** initializes and trains a new actor–critic pair for each task. Finally, **Isolated** initializes each new pair from the preceding stage but retains a separate checkpoint per task, avoiding shared-parameter interference at the cost of storing one learner per task.

Metrics: Following prior work [14,21,23], forward transfer (FWT) averages training success on new tasks, and gains over Single-task RL indicate transfer between tasks. Negative backward transfer (NBT) measures the performance on preceding tasks from the current stage, where lower values indicate better backward retention. Continual AUC combines learning efficiency and retention. Final accuracy and forgetting describe end-of-sequence success and prior-task loss.

Stages share an environment-step budget and uniform evaluation grid $\mathcal{E}$. Let $c_{i,j,e}$ be task j's success rate after e steps in stage i. Set $c_{i,i} = \max_e c_{i,i,e}$ and $c_{i,j} = c_{i,j,e_i^*}$, using the earliest maximizing checkpoint e_i^*.

$$\mathrm{FWT}_k = \frac{1}{|\mathcal{E}|}\sum_{e\in\mathcal{E}} c_{k,k,e},$$
$$\mathrm{NBT}_k = \frac{1}{K-k}\sum_{m=k+1}^{K}\left(c_{k,k} - c_{m,k}\right),$$
$$\mathrm{AUC}_k = \frac{1}{K-k+1}\left(\mathrm{FWT}_k + \sum_{m=k+1}^{K} c_{m,k}\right).$$

We average FWT and continual AUC over $k \in [K]$, and NBT over $k \in [K-1]$. Final deployment metrics are $\mathrm{ACC}_{\mathrm{final}} = \frac{1}{K}\sum_k c_{K,k}$ and $\mathrm{F}_{\mathrm{final}} = \frac{1}{K-1}\sum_{k<K}(c_{k,k} - c_{K,k})$. Unlike NBT, $\mathrm{F}_{\mathrm{final}}$ ignores transient performance drops and measures forgetting only at the end of the sequence. These two metrics describe the end-of-sequence deployment performance.

Hyperparameter selection in ACCRUE*:* We choose λ in Eq. 5 to balance prior-task preservation against current-task adaptation, favoring stronger distillation when successive tasks share fewer behaviors. Accordingly, we use $\lambda = 1.0$ for the more diverse real-world NIST sequence and $\lambda = 0.1$ for the more closely related real-world AutoMate assemblies. The coefficient β in Eq. 1 controls the penalty on deviation from the base policy's action proposals. We use $\beta = 0.1$ in simulation and increase it to 10.0 on hardware, where keeping refinements close to the VLA's proposals is particularly important during physical contact.

B. Simulation Experiments

Benchmarks: We use AutoMate [36] and FORGE [35], both built on Isaac Lab [40]. AutoMate provides assembly tasks in which plugs of different shapes must be inserted into matching sockets. We organize these tasks into four five-stage sequences. FORGE provides three contact-rich tasks (peg insertion, gear meshing, and nut threading), which we run as a three-stage sequence. For both tasks, the policy operates in a 6D end-effector delta action space. As task ordering is known to affect continual learning outcomes [21], we report the aggregated results over multiple sequences. Specifically, we aggregate over all four asset sequences for AutoMate, and we sample four task orderings randomly for FORGE; every result is aggregated over 4 seeds. Figs. 3a and 3b visualizes the tasks.

Setup: Each stage runs 64 parallel environments for a budget of 500K environment steps, is evaluated every 50K steps, and terminates an episode on success. The reward is sparse and emitted only on task success. Since an episode ends at the success step, we label the final three steps of a successful episode with a reward of 1 so that the terminal

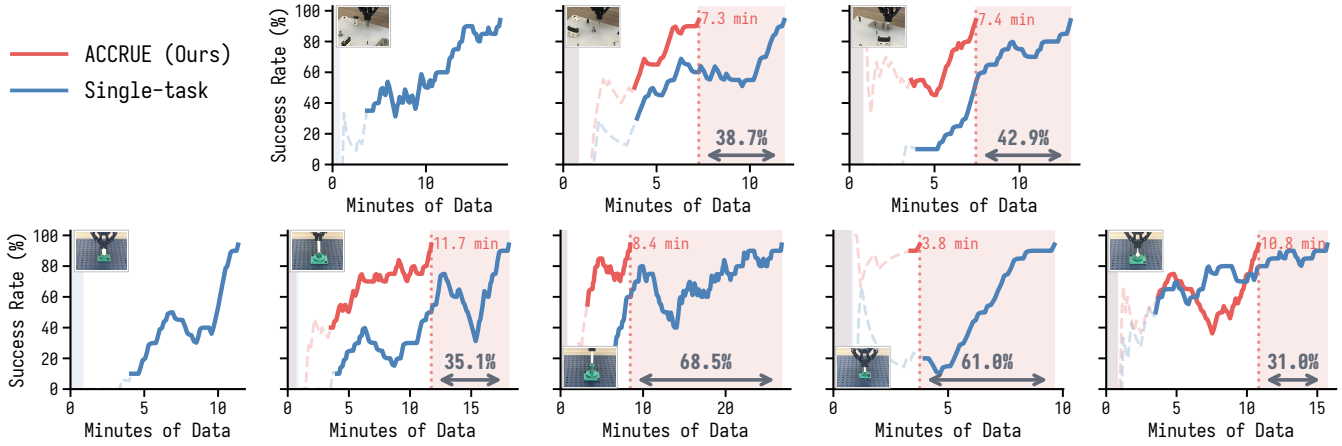


Fig. 4. **Real-world continual RL on DROID setup.** (Top) Three-stage NIST Assembly Task Board sequence: **Rectangle Peg** → **Cylinder Peg** → **Gear**. (Bottom) Five-stage AutoMate sequence: **00700** → **00028** → **00681** → **01041** → **01036**. Each panel reports the running success rate over a 20-episode window as a function of cumulative robot interaction time for RL, excluding human demonstration collection. Single-task trains the actor and critic from scratch, whereas ACCRUE inherits them from the previous stage. Gray shading marks warmup rollouts collected by the base policy to initialize the replay buffer, and dashed segments at the left indicate that fewer than 20 episodes have been collected. The dotted line marks when Continual reaches the stopping threshold of 95% success (19/20 episodes), and the shaded span reports robot interaction time saved relative to Single-task reaching the same threshold. The first stage of each sequence is shared by both conditions.

TABLE I. Results on FORGE, aggregated over 4 task orderings and 4 seeds. Bold and underlined values denote the best and second-best results among the matched shared actor–critic methods.

Method	NBT ↓	FWT ↑	AUC ↑	ACC_{final} ↑	F_{final} ↓
Shared actor–critic methods					
Sequential	0.427	0.819	<u>0.718</u>	<u>0.685</u>	0.409
EWC [39]	<u>0.217</u>	0.593	0.582	0.603	<u>0.224</u>
RETAIN [26]	0.514	0.802	0.682	0.644	0.474
ER [38]	0.518	**0.833**	0.697	0.626	0.516
ACCRUE (Ours)	**0.065**	<u>0.823</u>	**0.869**	**0.917**	**0.068**
Reference methods					
π_{base}	–	–	–	0.687	–
Single-task RL	–	0.741	–	–	–
Simple Recipe [14]	0.038	0.586	0.583	0.594	0.061
Isolated†	–	0.816	0.868	0.919	–

TABLE II. Results on AutoMate, aggregated over 4 asset sequences and 4 seeds. Bold and underlined values denote the best and second-best results among the matched shared actor–critic methods.

Method	NBT ↓	FWT ↑	AUC ↑	ACC_{final} ↑	F_{final} ↓
Shared actor–critic methods					
Sequential	0.178	0.824	0.792	0.765	0.175
EWC [39]	<u>0.075</u>	0.704	0.704	0.699	<u>0.067</u>
RETAIN [26]	0.191	0.822	0.784	0.750	0.192
ER [38]	0.167	<u>0.834</u>	<u>0.799</u>	<u>0.779</u>	0.176
ACCRUE (Ours)	**0.047**	**0.848**	**0.858**	**0.863**	**0.054**
Reference methods					
π_{base}	–	–	–	0.757	–
Single-task RL	–	0.807	–	–	–
Simple Recipe [14]	0.016	0.730	0.726	0.730	0.019
Isolated†	–	0.826	0.858	0.879	–

chunk carries a learning signal, and we use a discount factor of $\gamma = 0.997$. We use GR00T N1.7 [4] for π_{base}, which is fully fine-tuned at each stage on 1 demonstration for 1000 gradient steps on AutoMate, and on 10, 20, and 50 demonstrations for gear meshing, nut threading, and peg insertion, respectively, for 2000 steps on FORGE. These task-specific demonstration budgets aim to provide a reasonably competent base policy while minimizing human demonstration collection, reserving further improvement for online interactions in RL. We further examine the effect of fewer task demonstrations in Section V-D. For RL, we adopt FastSAC [31] over chunks of length $H = 16$ on frozen DINOv3 features [41], with 50 (AutoMate) / 100 (FORGE) retained episodes per stage.

Results: Among the matched shared actor–critic methods, ACCRUE achieves the lowest NBT and final forgetting and the highest AUC and final accuracy on both benchmarks (Tables I and II). It also exceeds Single-task RL in FWT under the same stage-wise interaction budget. EWC trades new-task learning for retention, whereas Sequential, ER, and RETAIN learn new tasks efficiently but forget more. On AutoMate, ACCRUE approaches the privileged Isolated reference while attaining higher FWT and maintaining a single shared actor–critic pair. As an external reference, Simple Recipe attains lower FWT and continual AUC than ACCRUE on both benchmarks under the same VLA backbone and interaction budget (Tables I and II). These results indicate the efficacy of ACCRUE in setup with limited interaction budget.

C. DROID Experiments

Tasks: To validate ACCRUE beyond simulation, We assess it on the DROID platform [37] with two task families. The first sequences three insertions on the NIST Assembly Task Board (rectangular peg, cylindrical peg, and gear mesh). The second runs a five-stage sequence of AutoMate assemblies on the same hardware, which tests whether the simulation task family transfers to a physical setup. For both tasks, the policy operates in a 6D end-effector delta action space, and each episode begins with the target object held by an object-specific 3D-printed gripper. Figs. 3c and 3d visualizes the tasks.

Setup: For the base policy π_{base}, we utilize $\pi_{0.5}$ [42] rather than GR00T N1.7 owing to its stronger performance on DROID. Before RL at each stage, we fine-tuned the model with LoRA on 5 (AutoMate) and 10 (NIST) demonstrations collected by SpaceMouse teleoperation per each task. Unlike in simulation, the low-dimensional input contains only the

TABLE III. Real-world retention reporting successes over 20 evaluation trials on each earlier task at the end of each later stage.

	After stage			
	2	3	4	5
NIST board				
Stage 1 (Rectangle Peg)	20	19		
Stage 2 (Cylinder Peg)	–	20		
AutoMate				
Stage 1 (00700)	20	19	18	19
Stage 2 (00028)	–	19	19	20
Stage 3 (00681)	–	–	19	20
Stage 4 (01041)	–	–	–	20

TABLE IV. Ablation of the design choices on FORGE sequence S1, holding the task sequence, stage budget, and all shared hyperparameters fixed. Best values are in bold and second best are underlined.

Method	NBT ↓	FWT ↑	AUC ↑	$\text{ACC}_{\text{final}}$ ↑	F_{final} ↓
ACCRUE	**0.028**	**0.854**	**0.906**	**0.961**	**0.037**
DINOv3 → GR00T VLM	0.071	0.814	0.846	0.883	0.113
$\pi_\theta(s, \tilde{\mathbf{a}}) \rightarrow \pi_\theta(s)$	<u>0.035</u>	0.807	0.869	0.936	0.050
$\beta = 0.0$	**0.029**	0.763	0.821	0.942	**0.037**
Reset Critic	0.042	0.841	<u>0.890</u>	<u>0.942</u>	<u>0.054</u>
Update Critic w/ $\mathcal{B}^{<k} \cup \mathcal{B}^k$	0.037	<u>0.842</u>	0.888	0.925	0.067

robot's seven joint positions, without additional object- or task-state information. For RL backbone, we employ FastTD3 [31] rather than FastSAC, because its explicit control of exploration noise avoids erratic contact behavior of the arm. We use actor and critic over chunks of length $H = 10$, with 20 retained episodes per stage. The reward is sparse on episode success, labeled by an operator. An asynchronous learner and actor process, similar to [13], keeps data collection running while the learner is busy. We report a running success rate over a window of 20 episodes and terminate a stage once it reaches 95% (19/20). Retention is measured by 20 evaluation trials on each earlier task at the end of each later stage. We compare against single-task RL within the same framework, since a full baseline sweep is not feasible at this scale.

Results: Inheriting the actor and critic reduces the robot time needed to reach the stopping criterion by 31.0 to 68.5 percent across the stages of the two sequences (Fig. 4), and retention remains essentially complete, with every earlier task succeeding on at least 18 of 20 trials and 19.4 on average (Table III). Both effects are larger than in simulation, which we attribute to the far smaller interaction budget: a better starting point is worth more when a stage is measured in tens of minutes of robot time, and the saving is the difference between a stage that fits in a single session and one that does not. Notably, these results are obtained with a different base policy family and a different RL algorithm than in simulation, under real sensing and contact noise, indicating that ACCRUE depends on neither a particular base policy nor a particular RL algorithm.

D. Ablation Studies and Analyses

Actor–critic inputs: Replacing the fixed DINOv3 representation with features extracted by the stage-updated GR00T VLM yields worse retention and forward transfer (Table IV). Although this comparison also changes the encoder itself and therefore does not isolate representation drift, it supports the practical advantage of a fixed visual interface for inheriting the actor and critic across base-policy updates. Removing the reference-action input likewise lowers retention and transfer, proving that reference action provides useful conditioning beyond the visual state alone.

TABLE V. Effect of the number of retained episodes per stage N in FORGE sequence S1. Best values are in bold and second best are underlined.

N	NBT ↓	FWT ↑	AUC ↑	$\text{ACC}_{\text{final}}$ ↑	F_{final} ↓
10	0.107	<u>0.853</u>	0.854	0.847	0.192
20	0.093	0.844	0.859	0.869	0.167
50	0.060	0.838	0.871	0.900	0.104
100	**0.040**	**0.854**	**0.896**	**0.925**	**0.062**
200	<u>0.043</u>	0.842	<u>0.879</u>	<u>0.903</u>	<u>0.083</u>

TABLE VI. Ablation of demo retention during sequential fine-tuning of π_{base} on FORGE sequence S1. Best values are shown in bold and second-best values are underlined.

	π_{base} (GR00T N1.7)		ACCRUE				
Demo Retention	SR ↑	NBT ↓	NBT ↓	FWT ↑	AUC ↑	$\text{ACC}_{\text{final}}$ ↑	F_{final} ↓
100%	**0.710**	**0.013**	**0.028**	**0.854**	**0.906**	**0.961**	**0.037**
50%	0.580	<u>0.035</u>	<u>0.056</u>	0.817	<u>0.860</u>	<u>0.908</u>	<u>0.092</u>
10%	<u>0.637</u>	0.090	0.078	0.811	0.844	0.875	0.133
5%	0.577	0.163	0.071	<u>0.836</u>	0.859	0.894	0.113

TABLE VII. ACCRUE with different π_{base} in FORGE sequence S1.

Method	Base SR	NBT ↓	FWT ↑	AUC ↑	$\text{ACC}_{\text{final}}$ ↑	F_{final} ↓
π_{base}: GR00T N1.7 [4] (VLA)						
ER	0.710	0.339	0.840	0.705	0.611	0.529
ACCRUE		**0.028**	**0.854**	**0.906**	**0.961**	**0.037**
π_{base}: Fast-WAM [43] (WAM)						
ER	0.347	0.419	0.742	0.588	0.517	0.625
ACCRUE		**0.035**	**0.768**	**0.847**	**0.905**	**0.042**

Reference guidance and critic inheritance: Removing the reference-deviation penalty by setting $\beta = 0$ in Eq. 1 has little effect on backward retention but substantially weakens forward transfer and task learning (Table IV). Explicitly biasing the actor toward the reference action therefore supports efficient local refinement rather than preservation of prior behavior. Resetting the critic at each stage produces modestly more forgetting and lower task-learning efficiency. Extending critic updates to prior-task replay increases forgetting and lowers continual AUC and final success, supporting our choice to train the critic only on current-task transitions.

Replay budget: Retaining $N = 100$–200 episodes per completed task (approximately 3% of the total replay buffer) provides the strongest joint retention–learning trade-off (Table V). Memories of 10–20 episodes are insufficient to preserve prior behavior, while increasing the budget from 100 to 200 episodes degrades performance. This non-monotonic trend shows that effective continual post-training does not follow directly from replaying more prior data.

Robustness to base-policy forgetting: We vary the fraction of prior-task demonstrations retained for sequential VLA fine-tuning from 100% to 5%, keeping the continual RL procedure and its replay budget fixed (Table VI). Reduced demonstration replay degrades both base-policy and post-RL performance relative to using all demonstrations. Nevertheless, ACCRUE mitigates forgetting and maintains new-task learning efficiency even when the base policy itself forgets.

Base-policy dependence: In Table VII, we apply ACCRUE on Fast-WAM [43] to prove the effectiveness regardless of model architecture. ACCRUE maintains strong

retention and final performance even with Fast-WAM despite their different architecture and pre-RL success rates. ER similarly degrades under both policy families.

VI. CONCLUSION

We presented ACCRUE, a continual off-policy RL framework for robot foundation models. Our approach is to inherit lightweight actor–critic guided by RFM's reference actions and to apply prior-task distillation. Through this, ACCRUE enables efficient RL on new tasks while preserving earlier behavior with limited replay. Experiments in simulation and on real robots demonstrate faster subsequent-task learning with significantly reduced forgetting. These results show that continual RL can support more sample-efficient post-training of robot foundation models as new tasks arrive, while retaining previously learned behaviors.

Our work establishes advances on task sequences of up to five stages in contact-rich assembly, with replay memory bounded per task. We leave it for future work to scale ACCRUE to longer task sequences under a fixed total memory budget. Additionally, a natural next step is to evaluate ACCRUE on a broader range of manipulation tasks, including less closely related tasks. Finally, it is promising to learn shared action representation and transfer RL post-training across varied embodiments or action spaces. Overall, ACCRUE shows that RL-acquired knowledge can accumulate across sequential deployment stages, providing a practical path to continually learn skills on robotic foundation models.

REFERENCES

[1] A. Brohan, *et al.*, "RT-2: Vision-language-action models transfer web knowledge to robotic control," in *Conference on Robot Learning*, 2023.

[2] M. J. Kim, *et al.*, "OpenVLA: An open-source vision-language-action model," in *Conference on Robot Learning*, 2024.

[3] K. Black, *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[4] NVIDIA, "GR00T N1: An open foundation model for generalist humanoid robots," *arXiv preprint arXiv:2503.14734*, 2025.

[5] S. Ye, Y. Ge, K. Zheng, S. Gao, S. Yu, *et al.*, "World action models are zero-shot policies," *arXiv preprint arXiv:2602.15922*, 2026.

[6] Dyna Robotics, "Dyna-2: A 1-million-hour scaling law for world-action models," August 2026.

[7] C. Zhu *et al.*, "Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets," *arXiv preprint arXiv:2504.02792*, 2025.

[8] A. Pai *et al.*, "mimic-video: Video-action models for generalizable robot control beyond VLAs," *arXiv preprint arXiv:2512.15692*, 2025.

[9] Y. Li, *et al.*, "Gr-rl: Going dexterous and precise for long-horizon robotic manipulation," *arXiv preprint arXiv:2512.01801*, 2025.

[10] W. Xiao, *et al.*, "Self-improving vision-language-action models with data generation via residual rl," in *International Conference on Learning Representations*, 2026.

[11] H. Li, *et al.*, "Simplevla-rl: Scaling vla training via reinforcement learning," in *International Conference on Learning Representations*, 2026.

[12] C. Xu, *et al.*, "Rl token: Bootstrapping online rl with vision-language-action models," *arXiv preprint arXiv:2604.23073*, 2026.

[13] P. Dong, K.-H. Hung, T. Gao, D. Sadigh, and C. Finn, "Expo-ft: Sample-efficient reinforcement learning finetuning for vision-language-action models," *arXiv preprint arXiv:2605.25477*, 2026.

[14] J. Hu, *et al.*, "Simple recipe works: Vision-language-action models are natural continual learners with reinforcement learning," *arXiv preprint arXiv:2603.11653*, 2026.

[15] Y. Liu, *et al.*, "Towards long-lived robots: Continual learning vla models via reinforcement fine-tuning," in *Robotics: Science and Systems*, 2026.

[16] Q. Zeng, *et al.*, "Crl-vla: Continual vision-language-action learning," *arXiv preprint arXiv:2602.03445*, 2026.

[17] Y. Guo, *et al.*, "Improving vision-language-action model with online reinforcement learning," in *IEEE International Conference on Robotics and Automation*, 2025.

[18] C. Kim, H. Lee, Y. Seo, K. Lee, and Y. Zhu, "DEAS: DEtached value learning with action sequence for scalable offline RL," in *International Conference on Learning Representations*, 2026.

[19] X. Yuan, T. Mu, S. Tao, Y. Fang, Z. Zhang, and H. Su, "Policy decorator: Model-agnostic online refinement for large policy model," in *International Conference on Learning Representations*, 2025.

[20] A. Wagenmaker, *et al.*, "Steering your diffusion policy with latent space reinforcement learning," in *Conference on Robot Learning*, 2025.

[21] B. Liu, *et al.*, "Libero: Benchmarking knowledge transfer for lifelong robot learning," in *Conference on Neural Information Processing Systems*, 2023.

[22] W. Wan, Y. Zhu, R. Shah, and Y. Zhu, "Lotus: Continual imitation learning for robot manipulation through unsupervised skill discovery," in *IEEE International Conference on Robotics and Automation*, 2024.

[23] H. Liu, C. Kim, B. Liu, M. Liu, and Y. Zhu, "Pretrained vision-language-action models are surprisingly resistant to forgetting in continual learning," in *International Conference on Machine Learning*, 2026.

[24] M. Du, Z. Sun, C. Xu, P. Shah, M. Itkina, and S. Song, "Memory anchors for continual robot learning," *arXiv preprint arXiv:2608.26545*, 2026.

[25] Z. Liu, *et al.*, "Tail: Task-specific adapters for imitation learning with large pretrained models," in *International Conference on Learning Representations*, 2024.

[26] Y. Yadav, Z. Zhou, A. Wagenmaker, K. Pertsch, and S. Levine, "Robust fine-tuning of vision-language-action robot policies via parameter merging," in *International Conference on Learning Representations*, 2026.

[27] H. Li, *et al.*, "CIDER: Continual interactive distillation for embodied reinforcement learning," *arXiv preprint arXiv:2608.21899*, 2026.

[28] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," in *Robotics: Science and Systems*, 2023.

[29] C. Chi, *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," *International Journal of Robotics Research*, 2025.

[30] G. M. Van de Ven, T. Tuytelaars, and A. S. Tolias, "Three types of incremental learning," *Nature Machine Intelligence*, 2022.

[31] Y. Seo, C. Sferrazza, H. Geng, M. Nauman, Z.-H. Yin, and P. Abbeel, "Fasttd3: Simple, fast, and capable reinforcement learning for humanoid control," *arXiv preprint arXiv:2505.22642*, 2025.

[32] Y. Seo, C. Sferrazza, J. Chen, G. Shi, R. Duan, and P. Abbeel, "Learning sim-to-real humanoid locomotion in 15 minutes," *arXiv preprint arXiv:2512.01996*, 2025.

[33] Q. Li, Z. P. Zhou, and S. Levine, "Reinforcement learning with action chunking," in *Conference on Neural Information Processing Systems*, 2025.

[34] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *International Conference on Machine Learning*, 2017.

[35] M. Noseworthy, *et al.*, "Forge: Force-guided exploration for robust contact-rich manipulation under uncertainty," in *IEEE Robotics and Automation Letters*, 2025.

[36] B. Tang, *et al.*, "Automate: Specialist and generalist assembly policies over diverse geometries." in *Robotics: Science and Systems*, 2024.

[37] A. Khazatsky, *et al.*, "Droid: A large-scale in-the-wild robot manipulation dataset," in *Robotics: Science and Systems*, 2024.

[38] A. Chaudhry, *et al.*, "On tiny episodic memories in continual learning," in *arXiv preprint arXiv:1902.10486*, 2019.

[39] J. Kirkpatrick, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, 2017.

[40] M. Mittal, *et al.*, "Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning," *arXiv preprint arXiv:2511.04831*, 2025.

[41] O. Siméoni, *et al.*, "DINOv3," *Transactions on Machine Learning Research*, 2026.

[42] K. Black, *et al.*, "$\pi_{0.5}$: a vision-language-action model with open-world generalization," in *Conference on Robot Learning*, 2025.

[43] T. Yuan, Z. Dong, Y. Liu, and H. Zhao, "Fast-wam: Do world action models need test-time future imagination?" *arXiv preprint arXiv:2603.16666*, 2026.