박 사 학 위 논 문
Ph.D. Dissertation

로봇 파운데이션 모델의 실환경 배포를 위한 효율적 사후 학습 기법

Efficient Post-Training for Robot Foundation Models in Real-World Deployment

2027

김 창 연 (金 昌 淵 Kim, Changyeon)

한 국 과 학 기 술 원

Korea Advanced Institute of Science and Technology

박 사 학 위 논 문

로봇 파운데이션 모델의 실환경 배포를 위한 효율적 사후 학습 기법

2027

김 창 연

한 국 과 학 기 술 원

김재철AI대학원

로봇 파운데이션 모델의 실환경 배포를 위한 효율적 사후 학습 기법

김 창 연

위 논문은 한국과학기술원 박사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2026년 10월 26일

심사위원장 Kimin Lee (인)
심 사 위 원 Jinwoo Shin (인)
심 사 위 원 Youngwoon Lee (인)
심 사 위 원 Jaegul Choo (인)
심 사 위 원 Yuke Zhu (인)

Efficient Post-Training for Robot Foundation Models in Real-World Deployment

Changyeon Kim

Advisor: Kimin Lee

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Doctor of Philosophy in AI

Daejeon, Korea
October 26, 2026

Approved by



Kimin Lee
Professor, Kim Jaechul Graduate School of AI

The study was conducted in accordance with Code of Research Ethics[1].

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my thesis contains honest conclusions based on my own careful research under the guidance of my advisor.

DAI

김창연. 로봇 파운데이션 모델의 실환경 배포를 위한 효율적 사후 학습 기법. 김재철AI대학원 . 2027년. 52+iv 쪽. 지도교수: Kimin Lee. (영문 논문)

Changyeon Kim. Efficient Post-Training for Robot Foundation Models in Real-World Deployment. Kim Jaechul Graduate School of AI . 2027. 52+iv pages. Advisor: Kimin Lee. (Text in English)

초 록

로봇 파운데이션 모델(Robot Foundation Model, RFM)은 로봇 조작을 위한 강력한 출발점을 제공하지만, 특정 목표 환경(target scene)에 안정적으로 배포하기 위해서는 시연 데이터(demonstration)를 넘어서는 수준으로 성능을 향상시켜야 한다. 그러나 실제 환경에서 이러한 성능 향상을 달성하는 데에는 많은 비용이 따른다. 온라인 상호작용에는 상당한 로봇 운용 시간과 수동 환경 초기화(reset)가 필요하고, 후보 정책들을 비교하려면 실제 로봇에서의 반복적인 평가가 요구되며, 새로운 작업을 매번 처음부터 학습하면 이전에 축적한 경험이 버려진다. 본 학위논문은 수집된 데이터, 참조 궤적, 그리고 이전에 학습된 지식을 재사용함으로써 이러한 비용을 줄이는 효율적인 사후 학습 기법들을 제시한다. 첫째, 고정된 데이터셋으로부터 행동 시퀀스(action sequence)에 대한 가치 함수를 학습하는 오프라인 강화 학습(reinforcement learning, RL) 프레임워크인 DEAS를 제안하여, 추가적인 온라인 상호작용 없이 행동 청크(action chunk)를 예측하는 RFM의 성능을 향상시킨다. 둘째, 성공한 참조 궤적을 이용하여 정책 체크포인트의 순위를 매기는 롤아웃 없는(rollout-free) 평가 지표인 RACE를 제안하여, 모든 후보를 실제 로봇에서 평가하지 않고도 실환경 배포 성공률과 일치하는 순위를 얻는다. 마지막으로, 경량 액터(actor) 및 크리틱(critic) 네트워크를 작업 간에 계속 이어서 학습하는 지속 오프-폴리시(off-policy) 강화 학습 프레임워크인 ACCRUE를 제안하여, 이전에 습득한 행동을 보존하면서 실제 로봇에서의 새로운 작업 학습을 가속한다. 이러한 기여들은 로봇 운용 시간과 사람의 노력에 드는 비용을 크게 줄임으로써 RFM의 실환경 사후 학습을 보다 실용적으로 만든다.

핵 심 낱 말 로봇 파운데이션 모델, 사후 학습, 강화 학습, 지속 학습, 정책 평가

Abstract

Robot Foundation Models (RFMs) provide a capable starting point for robot manipulation, but deploying them in a specific target scene reliably requires improving their performance beyond demonstrations. Achieving such improvement in the real world necessitates lots of costs: online interaction demands substantial robot time and manual resets, comparing candidate policies requires repeated physical trials, and learning each new task from scratch discards prior experience. This dissertation presents efficient post-training methods that reduce these costs by reusing collected data, reference trajectories, and previously learned knowledge. First, we propose DEAS, an offline reinforcement learning (RL) framework that learns values over action sequences from fixed datasets, improving RFMs that predict action chunks without additional online interaction. Second, we introduce RACE, a rollout-free metric that ranks policy checkpoints using successful reference trajectories, yielding rankings that align with real-world deployment success without evaluating every candidate on hardware. Finally, we present ACCRUE, a continual off-policy RL framework that carries lightweight actor and critic networks across tasks, accelerating new-task learning on real robots while preserving previously acquired behavior. Together, these contributions make real-world post-training of RFMs more practical by substantially reducing the costs for robot time and human effort.

Keywords Robot Foundation Models, post-training, reinforcement learning, continual learning, policy evaluation

Contents

Contents i
List of Tables iii
List of Figures iv

Chapter 1. Introduction **1**
1.1 Robot Foundation Models in Real-World Deployment 1
1.2 Challenges of Improving Policies in the Deployment Scene 1
1.2.1 Learning without Large-Scale Online Interaction 2
1.2.2 Selecting Policies without Evaluating Every Candidate 2
1.2.3 Accumulating Improvements across Tasks 2
1.3 Dissertation Overview 3

Chapter 2. DEAS: Detached Value Learning with Action Sequences for Scalable Offline RL **4**
2.1 Introduction 4
2.2 Related Work 6
2.3 Preliminaries 6
2.4 Method 7
2.4.1 TD Learning over Action Sequences 7
2.4.2 DEAS: DEtached value learning with Action Sequence 8
2.5 Experiments 10
2.5.1 OGBench Experiments 10
2.5.2 VLA Experiments 11
2.5.3 Quantitative Analyses 13
2.5.4 Ablation Studies 14
2.6 Conclusion 16

Chapter 3. RACE: Calibrated Cumulative Scoring for Rollout-Free Checkpoint Selection **17**
3.1 Introduction 17
3.2 Related Work 18
3.3 Main Method 19
3.3.1 Problem Formulation 19
3.3.2 Pitfall of Averaged Prediction Error 20
3.3.3 RACE 20
3.4 Experiments 22
3.4.1 Simulation Experiments 23
3.4.2 Real-World Experiments 24
3.4.3 Analyses and Ablation Studies 26
3.5 Conclusion 28

Chapter 4. ACCRUE: Continual Off-Policy Reinforcement Learning for Robot Foundation Models **29**
4.1 Introduction . . . 29
4.2 Related Work . . . 31
4.2.1 RL Post-Training for RFMs . . . 31
4.2.2 Continual Learning for Robotics . . . 31
4.3 Preliminaries . . . 32
4.3.1 Robot Foundation Models . . . 32
4.3.2 Problem Setup . . . 32
4.3.3 Action-Chunk Reinforcement Learning . . . 32
4.4 Method . . . 33
4.4.1 Actor–Critic Inputs . . . 33
4.4.2 Current-Task Off-Policy RL . . . 34
4.4.3 Continual Operation . . . 34
4.5 Experiments . . . 35
4.5.1 Experimental Setup . . . 35
4.5.2 Simulation Experiments . . . 37
4.5.3 DROID Experiments . . . 39
4.5.4 Ablation Studies and Analyses . . . 40
4.6 Conclusion . . . 42

Chapter 5. Concluding Remark **43**

Bibliography **44**

List of Tables

2.1 **Offline RL results on OGBench.** Success rates (%) in six task categories, with 95% stratified bootstrap intervals over four runs. 11
2.2 **RoboCasa Kitchen evaluation.** Offline IL/RL on four selected tasks after VLA fine-tuning; success rates over 50 episodes and three seeds. 12
2.3 **Real-world evaluation.** Partial success rates (%) on three pick-and-place tasks from five initial points, with 20 trials per task. 13
2.4 **DEAS ablation studies.** Effects of sequence length, model size, training objectives, and intra-/inter-option discounts on success rates. 15

3.1 **Checkpoint ranking on DexMimicGen.** Ten references rank 12 checkpoints per run; means and 95% bootstrap intervals over eight tasks and 10 seeds. 23
3.2 **Perturbation analysis on Box Cleanup.** Deployment success and offline scores under uniform and bottleneck perturbations in 50 fixed scenes. 26
3.3 **Component and temporal ablations.** Grouping, threshold, state-conditioning, and aggregation comparisons across four tasks, with 95% bootstrap intervals. 27
3.4 **Task-specific versus shared settings.** Scoring comparisons across eight tasks and reference sources, with 95% bootstrap intervals. 27
3.5 **Real-world reference-set analyses.** Rollout augmentation on PnP Apple and successful references containing random 3–5 second pauses. 28

4.1 **Continual learning results on FORGE.** Aggregates over four task orderings and four seeds; best/second-best matched actor–critic results are marked. 38
4.2 **Continual learning results on AutoMate.** Aggregates over four asset sequences and four seeds; best/second-best matched actor–critic results are marked. 38
4.3 **Real-world retention.** Success counts over 20 trials per earlier task at the end of each later stage. 40
4.4 **ACCRUE design ablations.** FORGE sequence S1 comparisons with task order, stage budget, and shared hyperparameters fixed. 40
4.5 **Effect of retained replay size.** Varying the number of retained episodes per stage on FORGE sequence S1. 41
4.6 **Demonstration-retention ablation.** Sequential base-policy fine-tuning with different retained demonstration budgets on FORGE sequence S1. 41
4.7 **ACCRUE with different base policies.** GR00T N1.7 (VLA) and Fast-WAM (WAM) comparisons on FORGE sequence S1. 41

List of Figures

2.1 **Overview of DEAS.** Detached action-sequence value learning with distributional objectives and dual discount factors. . . . 5
2.2 **Simulation task examples.** Thirty OGBench tasks and four challenging RoboCasa Kitchen manipulation tasks. . . . 10
2.3 **Performance scaling with dataset size.** Success rates on three OGBench tasks; means and bootstrap intervals over four runs. . . . 11
2.4 **Real-world tasks.** Countertop-to-cabinet pick-and-place with peach, milka, and hichew objects. . . . 13
2.5 **Value calibration on held-out trajectories.** Predicted critic values versus Monte-Carlo returns on puzzle-4x4 and cube-quadruple. . . . 14
2.6 **Robustness to data quality.** Success rates and value calibration under different data mixtures on cube-quadruple. . . . 14

3.1 **Motivation and key idea of RACE.** Reference-calibrated cumulative scoring distinguishes deviations that averaged prediction error can obscure. . . . 19
3.2 **RACE scoring procedure.** Action-group deviations, state-conditional thresholds, and running-product aggregation produce relative checkpoint scores. . . . 21
3.3 **Diffusion policy checkpoint rankings.** Offline scores versus deployment success on three real-robot tasks, with 10 checkpoints per task. . . . 25
3.4 **VLA checkpoint rankings.** Offline scores versus deployment success for $\pi_{0.5}$-DROID on three tasks, with 10 checkpoints per task. . . . 25

4.1 **Single-task and continual off-policy RL.** Resetting the learner per task versus carrying actor–critic knowledge with limited prior-task replay. . . . 30
4.2 **Overview of ACCRUE.** Reference-conditioned actor–critic carryover, fixed visual features, and prior-task actor distillation. . . . 31
4.3 **Experimental tasks.** FORGE and AutoMate simulation sequences, the real-world NIST Assembly Task Board, and physical AutoMate assemblies. . . . 36
4.4 **Real-world continual RL learning curves.** Running success versus robot interaction time for three-stage NIST and five-stage AutoMate sequences. . . . 37

Chapter 1. Introduction

1.1 Robot Foundation Models in Real-World Deployment

Robot Foundation Models (RFMs) are pretrained on large and diverse robot data to provide reusable manipulation capabilities across tasks, embodiments, and environments. Vision-language-action models (VLAs) map visual observations, proprioception, and language instructions directly to robot actions (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024, 2025; NVIDIA, 2025b; Gemini Robotics Team et al., 2025b). More recently, world-action models (WAMs) jointly model future observations and actions, using video prediction as an additional source of pretraining signal (Zhu et al., 2025; Pai et al., 2025; Kim et al., 2026d; Ye et al., 2026; Yuan et al., 2026; Dyna Robotics, 2026); several of these build on large pretrained video world models such as NVIDIA Cosmos (Agarwal et al., 2025). Despite their differences in architecture and pretraining objective, these models share a common role: they provide a capable starting point from which a robot can begin a new task without training a policy from scratch.

For a deployed robot, however, this starting point is rarely the end goal. A robot installed in a particular factory cell, kitchen, or warehouse must complete specific tasks reliably under the cameras, lighting, objects, and contact conditions of that *target scene*. Broad pretraining does not guarantee such reliability, and the remaining failures often concentrate on the precise and contact-rich phases that matter most: a policy that reaches an assembly site successfully may still fail to insert a particular part. Fine-tuning on demonstrations collected in the target scene narrows this gap, but imitation is bounded by the quality and coverage of those demonstrations; it rarely surpasses the demonstrator and does not learn from its own failures. What matters in deployment is therefore less which RFM is used than whether its performance can be driven *beyond the demonstrations* in the target scene.

Reinforcement learning (RL) offers a natural mechanism for this improvement because it optimizes task reward through experience rather than reproducing recorded actions. Recent work has shown that RL can refine pretrained robot policies, for example by steering their action proposals with learned values or by training lightweight actor–critic learners on top of a frozen foundation policy (Nakamoto et al., 2024; Wagenmaker et al., 2025; Chen et al., 2025; Li et al., 2025c; Xu et al., 2026; Dong et al., 2026). This dissertation studies how to make such improvement *efficient* enough to be practical in real-world deployment.

1.2 Challenges of Improving Policies in the Deployment Scene

Improving an RFM in its deployment scene differs fundamentally from improving a policy in simulation. Every rollout consumes physical robot time, every failure may require a human to restore the scene, and every candidate policy must eventually be trusted on real hardware. These costs make the standard recipe of collecting more interaction, evaluating more checkpoints, and retraining from scratch for each new task burdensome. We identify three challenges that arise from this setting.

1.2.1 Learning without Large-Scale Online Interaction

Online RL typically relies on a large number of interaction episodes, and this volume is difficult to obtain on a physical robot. Each episode occupies the robot and an operator, and after every success or failure the scene must be reset: objects must be returned to their initial configurations, dropped parts must be recovered, and outcomes must often be labeled by hand. Collecting a large amount of task-specific data in the target scene, whether through teleoperated demonstrations or autonomous trials, is therefore expensive in itself. In practice, the data available for improvement is a modest, fixed collection of demonstrations and previously executed rollouts that includes both successes and failures.

Offline RL is designed to learn from such fixed datasets without further interaction (Levine et al., 2020; Kumar et al., 2020; Kostrikov et al., 2022), yet existing methods cannot simply be applied to RFMs. Expressive robot policies, including most VLAs and WAMs, predict *action chunks*: sequences of future actions produced at one policy query (Zhao et al., 2023; Chi et al., 2025). Learning values over such sequences can shorten the effective decision horizon of long manipulation tasks, but it also enlarges the action space over which the critic must generalize. When an actor is optimized against these values, it can propose sequences outside the logged data and exploit inaccurate value estimates (Seo and Abbeel, 2025; Li et al., 2025a), and offline training offers no new interaction to correct them. Leveraging fixed deployment data thus requires value learning that matches the temporally extended outputs of RFMs while remaining robust to the errors of sequence-level critics.

1.2.2 Selecting Policies without Evaluating Every Candidate

Any post-training procedure produces many candidate checkpoints whose deployment success may differ substantially, and deciding which one to deploy is itself costly. Evaluating each candidate with real-world trials multiplies the burden of robot time and resets described above. Held-out action prediction error is inexpensive to compute, but smaller deviations from recorded actions do not necessarily imply better closed-loop task completion (Li et al., 2024b; Pan et al., 2026): some deviations are easily recovered while others cause failure, and averaging errors along recorded observations ignores that an early deviation may lead execution away from the reference altogether. Simulated or learned evaluators can reduce hardware usage but require building a faithful model of the target scene. Efficient deployment therefore needs a rollout-free criterion whose checkpoint rankings agree with real-world success.

1.2.3 Accumulating Improvements across Tasks

A deployed robot rarely performs a single task for its lifetime; new requirements arrive after earlier tasks have been learned. Restarting RL for each task discards the behavioral and value knowledge acquired through costly earlier interaction, whereas continuing to train shared networks on a new task can degrade previously acquired behavior. Retaining all past interaction data becomes increasingly expensive, and the foundation policy itself may be fine-tuned on newly collected demonstrations between tasks, changing the inputs on which an inherited learner depends. Making real-world improvement cumulative requires reusing prior RL to accelerate new tasks while preserving earlier tasks under limited replay.

1.3 Dissertation Overview

This dissertation addresses these challenges with three methods, each targeting one source of deployment cost.

- **Chapter 2**: To learn from limited, fixed data, we propose *DEtached value learning with Action Sequence (DEAS)*, an offline RL framework that learns values over action sequences. DEAS trains the critic independently of actor-predicted actions, grounding value updates in sequences present in the dataset, and stabilizes learning with distributional objectives and dual discount factors. The learned values improve VLAs by selecting among their proposed action chunks, raising mean success on RoboCasa Kitchen tasks over a fine-tuned GR00T N1.5 baseline and improving performance in real-robot Franka kitchen tasks. This work was published at the International Conference on Learning Representations (ICLR) in 2026 (Kim et al., 2026a).

- **Chapter 3**: To select policies without evaluating every candidate, we introduce *Reference-cAlibrated cumulative scoring for rollout-free Checkpoint sElection (RACE)*, a rollout-free metric that ranks checkpoints using a fixed set of successful reference trajectories. RACE calibrates action deviations against state-conditional thresholds derived from the references and accumulates the resulting scores through a running product, so that substantial early deviations discount later predictions. Across simulated DexMimicGen tasks and real-robot experiments with diffusion policies and a VLA, its rankings agree more closely with deployment success than prediction-error criteria. This work was accepted to the Robot Learning Workshop at the Conference on Neural Information Processing Systems (NeurIPS) in 2026 (Kim et al., 2026c) and is currently under review at the IEEE International Conference on Robotics and Automation (ICRA) 2027.

- **Chapter 4**: To accumulate improvements across tasks, we present *Actor–Critic Carryover for continual off-policy Reinforcement learning Using retained Experience (ACCRUE)*, a continual off-policy RL framework that carries lightweight actor and critic networks across tasks on top of an RFM. Fixed visual features keep the inherited learner compatible with base-policy updates, and actor distillation on a small number of retained episodes preserves prior behavior. On contact-rich assembly sequences in simulation and on hardware, ACCRUE reduces real-robot training time for subsequent tasks by 31.0–68.5% while retaining near-perfect success on earlier tasks. This work was accepted to the Continually Self-Improving Robots Workshop at the Conference on Robot Learning (CoRL) in 2026 (Kim et al., 2026b) and is currently under review at ICRA 2027.

Chapter 2. DEAS: Detached Value Learning with Action Sequences for Scalable Offline RL

2.1 Introduction

Offline reinforcement learning (RL) (Lange et al., 2012; Levine et al., 2020) enables learning from static datasets without incurring online data collection risks, while circumventing the need for expensive expert demonstrations. However, existing methods primarily focus on short-horizon tasks with dense rewards (Yu et al., 2020; Fu et al., 2020; Gulcehre et al., 2020; Mandlekar et al., 2021) and struggle to scale to complex long-horizon scenarios. Recent attempts using large-scale architectures (Kumar et al., 2023a,b; Chebotar et al., 2023; Springenberg et al., 2024) show promise, but their effectiveness on complex tasks remains unexplored.

To address the need for long-horizon evaluation, recent work (Park et al., 2025a,b) has proposed challenging benchmarks for complex offline RL and demonstrated that reducing the effective planning horizon (i.e., shortening the time span over which the agent must plan) in both value and policy learning via n-step TD updates with high n values and hierarchical policies is essential. However, these approaches rely on goal-conditioned RL with explicit, expert-provided goals, which are often unavailable in practice. For instance, high n values in n-step TD updates introduce increased bias and bootstrap error in standard RL without explicit goal information (Tsitsiklis and Van Roy, 1996; Kearns and Singh, 2000; Sutton and Barto, 2018).

These limitations underscore the need for alternative approaches to horizon reduction (reducing the planning horizon) that work without explicit goal conditioning. One promising direction is leveraging action sequences, which have shown success in behavior cloning (Pomerleau, 1988) for capturing noisy, temporally-relevant distributions in expert demonstrations (Chi et al., 2025; Zhao et al., 2023). However, existing attempts to use action sequences for RL remain insufficient for achieving robust horizon reduction. Q-chunking (Li et al., 2025a) has explored the use of action sequences for RL, demonstrating their potential for temporally consistent exploration. However, introducing action sequences to standard actor-critic frameworks causes severe value overestimation (Seo and Abbeel, 2025) due to actors maximizing over potentially erroneous critic estimates with widely spanned action spaces. This problem is exacerbated in offline RL where distribution shift creates extrapolation errors (Kumar et al., 2019; Fujimoto et al., 2019; Kumar et al., 2020). While CQN-AS (Seo and Abbeel, 2025) proposes a value-only approach to avoid this issue, it introduces discretization errors that limit performance in complex tasks and cannot leverage expressive policy classes (Wang et al., 2023; Hansen-Estruch et al., 2023; Park et al., 2025c). For this reason, our research aims to develop methods that can leverage action sequences for horizon reduction while avoiding value overestimation and maintaining compatibility with expressive policy architectures.

Our approach We present *DEtached value learning with Action Sequence* (DEAS), an offline RL framework that leverages action sequences for scalable value learning in complex tasks. Our method treats consecutive action timesteps as inputs to the value function to provide more expressive information than single-step actions. This design provides principled horizon reduction analogous to n-step TD updates with temporally extended actions, while action sequences offer richer information than single-

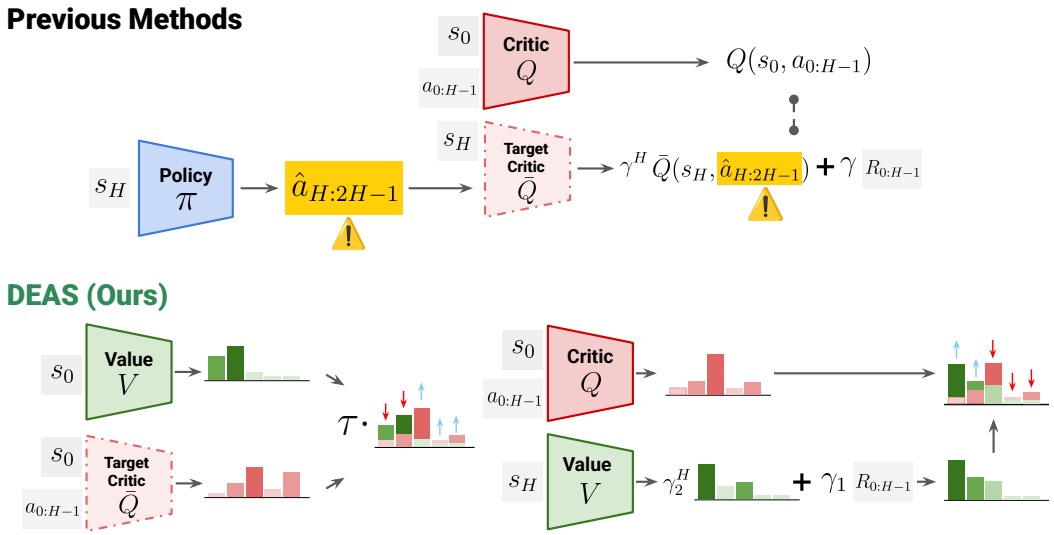


Figure 2.1: Overview. DEAS is an offline RL framework that learns from action sequences instead of single actions. Unlike previous methods that couple actor-critic training, our key insight is to train the critic separately from the policy (detached value learning) using action sequences, which enables stable learning while avoiding value overestimation. We further enhance stability by combining distributional RL objectives and using dual discount factors, which leads to additional improvement.

step actions without requiring explicit goal conditioning. To address the value overestimation challenges inherent in learning value functions with action sequences in offline RL settings, we employ detached value learning (Kostrikov et al., 2022) that decouples critic training from the actor, biasing value estimates toward high-return actions present in the offline dataset. This method is appealing as it can be applied to any expressive policy architectures including large-scale Vision-Language-Action models (VLAs) without the hazard of value overestimation. Additionally, we propose to incorporate distributional RL (Farebrother et al., 2024) in value learning to mitigate instability from accumulated bias in multi-step returns.

We validate DEAS through comprehensive experiments on challenging long-horizon tasks from OGBench (Park et al., 2025a), where standard offline RL methods struggle to achieve meaningful success rates. Our method consistently outperforms all baselines, demonstrating its effectiveness on complex tasks. Additionally, we show that DEAS can be used to improve the performance of VLAs (NVIDIA, 2025b) in hard tasks from RoboCasa Kitchen (Nasiriany et al., 2024) and real-world manipulation tasks, which significantly improves performance compared to policies trained solely on expert demonstrations. These results demonstrate DEAS's practical applicability and potential for scaling offline RL to real-world scenarios.

Contributions We highlight the key contributions of this chapter below:

- We present DEAS: *DEtached value learning with Action Sequence*, a simple yet effective offline RL method that leverages action sequences for training critics and employs detached value learning with classification loss for stable training.
- We demonstrate that DEAS significantly outperforms baselines on complex, long-horizon tasks across 30 diverse scenarios in OGBench (Park et al., 2025a).
- We demonstrate that DEAS can enhance the performance of large-scale VLAs, achieving superior

results on complex tasks from RoboCasa Kitchen (Nasiriany et al., 2024) and real-world manipulation tasks compared to policies trained solely on expert demonstrations.

2.2 Related Work

Offline reinforcement learning Offline RL focuses on learning policies from fixed datasets without further environment interaction (Levine et al., 2020). The primary challenge lies in the distributional shift between the behavior policy and the learned policy, which can result in value overestimation and suboptimal performance. Previous work has proposed various approaches including weighted regression (Peng et al., 2019; Nair et al., 2020; Wang et al., 2020), conservative regularization (Kumar et al., 2020), behavioral regularization (Fujimoto et al., 2019; Fujimoto and Gu, 2021; Tarasov et al., 2023; Park et al., 2025c), and in-sample distribution maximization (Kostrikov et al., 2022; Xu et al., 2023; Garg et al., 2023). Our method builds upon in-sample distribution maximization approaches, particularly IQL (Kostrikov et al., 2022), extending them to handle action sequences while maintaining stability by removing the critic update based on the actor's output. Furthermore, our method has the advantage of being adaptable to any policy extraction method for the final policy, making it more flexible and practical.

BC/RL with action sequence Adopting action sequence has been actively investigated in both imitation learning and RL. Behavior cloning advances show that predicting action sequences captures temporal dependencies from expert demonstrations that single-step actions miss (Chi et al., 2025; Zhao et al., 2023; Black et al., 2024; NVIDIA, 2025b; Black et al., 2025). Several works have introduced action sequences into RL (Li et al., 2024a; Tian et al., 2025), with Q-Chunking (Li et al., 2025a) demonstrating incorporation into actor-critic frameworks in offline-to-online RL without policy class constraints. However, this approach faces fundamental challenges: expanded action spaces increase value overestimation risk, particularly in offline settings with limited data coverage (Kumar et al., 2019), yet this issue remains unaddressed. CQN-AS (Seo and Abbeel, 2025) circumvents this by removing the actor entirely, but introduces accumulating discretization errors that severely limit performance in complex tasks and prevent use of expressive policy classes (Wang et al., 2023; Park et al., 2025c). Our approach uniquely combines both paradigms: we leverage horizon reduction from action sequences while addressing value overestimation through detached value learning, enabling stable training with any policy architecture.

2.3 Preliminaries

Problem formulation We consider a Markov Decision Process (MDP) (Sutton and Barto, 2018) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, R, \rho_0, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $R(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $p(s'|s,a) : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function, ρ_0 is the initial state distribution, and γ is the discount factor. In this chapter, we focus on offline reinforcement learning, where we have access only to a static dataset $\mathcal{D} = \{\tau^i\}_{i=0}^N$ containing N trajectories of fixed length H, where each trajectory $\tau^i = (s_0, a_0, r_0, \ldots, s_H, a_H, r_H)$ represents a sequence of states, actions, and rewards. The dataset is collected using a data collection policy $\pi_\mathcal{D} : \mathcal{S} \to \Delta(\mathcal{A})$, which may be unknown or suboptimal. Unlike online RL, we cannot interact with the environment during training. The objective is to learn a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the expected sum of discounted rewards $\mathbb{E}_{\rho_0,\pi,p}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$ using only this fixed dataset.

Temporal-difference (TD) learning Modern value-based RL approaches usually update a state-action value function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ to approximate the maximum discounted cumulative reward by bootstrapping from a successor state:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \Big[r_t + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a') - Q(s_t, a_t) \Big].$$

This one-step update progressively aligns Q with its Bellman target using sampled transitions. For reducing the bootstraping bias problem in long-horizon tasks, multi-step TD (Tsitsiklis and Van Roy, 1996; Kearns and Singh, 2000; Sutton and Barto, 2018) extends this idea by accumulating rewards over several steps before bootstrapping while retaining the same basic structure.

Implicit Q Learning (IQL) (Kostrikov et al., 2022) Instead of regularizing the critic with the actor output, IQL approximates the optimal critic to be maximized only in the region of action distributions present in the offline dataset with an in-sample expectile regression. Given a parameterized critic $Q(s_t, a_t; \theta)$, target critic $Q(s_t, a_t; \bar{\theta})$, and value network $V(s_t; \psi)$, the objective for value learning is defined as:

$$\mathcal{L}_V(\psi) = \mathbb{E}_{(s_t, a_t) \sim \mathcal{D}} \left[L_2^\tau (\bar{Q}(s_t, a_t; \bar{\theta}) - V(s_t; \psi)) \right] \tag{2.1}$$

$$\mathcal{L}_Q(\theta) = \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{D}} \left[(R(s_t, a_t) + \gamma V(s_{t+1}; \psi) - Q(s_t, a_t; \theta))^2 \right] \tag{2.2}$$

where $\mathcal{L}_2^\tau(u) = |\tau - \mathbb{1}(u < 0)| u^2$ is the expectile loss with expectile parameter $\tau \in [0, 1]$. By using $\tau > 0.5$, Equation (2.1) applies the higher weight on positive errors than negative errors, so that V approximates an upper expectile of the in-distribution TD targets. This objective allows V and Q to be approximated only within the in-distribution region of actions.

2.4 Method

We propose DEtached value learning with Action Sequence (DEAS), an offline RL method that models action sequences for scalable value learning. Our approach consists of two key components: (1) a critic $Q(s_t, \mathbf{a}_t; \theta)$ that estimates expected returns for H-step action sequence $\mathbf{a}_t := a_{t:t+H-1}$ from state s_t under the data collection policy $\pi_\mathcal{D}$, and (2) a flexible policy update mechanism applicable to any policy $\pi(\mathbf{a}_t; s_t, \phi)$ that outputs H-step action sequences. Section 2.4.1 describes how we extend temporal with action sequences, while Section 2.4.2 introduces how DEAS enables stable training through detached value learning, distributional RL, and dual discount factors. We provide pseudocode in Algorithm 2.1.

2.4.1 TD Learning over Action Sequences

Complex tasks require coordinated action sequences where each action's effectiveness depends on its context within the sequence. For instance, in OGBench `puzzle` or `cube` tasks, success depends on planning through multiple intermediate steps and maintaining consistent actions over extended periods. These temporal dependencies and hidden sub-tasks are not captured by current state representations, making it challenging for agents to learn effective policies. The challenge becomes even greater in goal-free settings, where agents must discover these sequential patterns from offline data without explicit goal instructions.

To address this challenge, we formalize temporally extended decisions by treating each fixed-length H-step action sequence as a single decision unit. Given the underlying MDP $\mathcal{M}$, we consider an state-

Algorithm 2.1 DEAS

Required: Offline dataset $\mathcal{D}$, Support range for return $\mathbf{v}_{\min}$, $\mathbf{v}_{\max}$, number of bins m, discount factor γ_1, γ_2
Initialize parameters $\psi, \theta, \bar{\theta}, \phi$
while not converged **do**
  Sample batch $\{(s_t, \mathbf{a}_t, R_{t:t+H-1}, s_{t+H})\}$ from $\mathcal{D}$
  Compute the discounted return of H-step action sequence $\tilde{R}(s_t, \mathbf{a}_t, \gamma)$ using Equation (2.3)
  Compute $\bar{Q}(s, \mathbf{a}; \bar{\theta})$ and $V(s; \psi)$ using Equation (2.7)
  ▷ Update V Network
  Update $V(s; \psi)$ to minimize Equation (2.8) with $\bar{Q}(s, \mathbf{a}; \bar{\theta})$ and $V(s; \psi)$
  ▷ Update Q Network
  Update $Q(s, \mathbf{a}; \theta)$ to minimize Equation (2.9)
  ▷ Update Actor Network
  Update $\pi(s; \phi)$ with any type of policy extraction algorithms (e.g., BoN, DPG, AWR, etc.)
  Update $\bar{\theta} = (1 - \beta) \cdot \bar{\theta} + \beta \cdot \theta$
return $\pi(s)$

action value function $Q(s, \mathbf{a})$ over H-step sequence of actions $\mathbf{a}_t := (a_t, a_{t+1}, \ldots, a_{t+H-1}) \in \mathcal{A}^H$. Executing $\mathbf{a}_t$ from state s_t means applying the primitive actions in order, collecting the discounted return

$$\tilde{R}(s_t, \mathbf{a}_t, \gamma) := \mathbb{E}\left[\sum_{k=0}^{H-1} \gamma^k R(s_{t+k}, a_{t+k}) \,\middle|\, s_t, \mathbf{a}_t\right], \tag{2.3}$$

and transitioning to a successor state s_{t+H}. Our TD updates are then performed directly on $Q(s_t, \mathbf{a}_t)$ using $\tilde{R}(s_t, \mathbf{a}_t, \gamma)$ as the multi-step target, i.e., standard TD learning applied to sequence actions in $\mathcal{A}^H$. This formulation is equivalent to considering decision-making at every H-th time step with temporally extended actions.

For updating $Q(s, \mathbf{a})$, we can use the following TD learning objective that extends standard Q-learning (Bradtke and Duff, 1994; Sutton et al., 1999):

$$Q(s_t, \mathbf{a}_t) \leftarrow Q(s_t, \mathbf{a}_t) + \alpha \left[\tilde{R}(s_t, \mathbf{a}_t, \gamma_1) + \gamma_2^H \max_{o' \in \mathcal{A}^H} Q(s_{t+H}, o') - Q(s_t, \mathbf{a}_t)\right] \tag{2.4}$$

where γ_1 is used to construct the H-step return within a sequence, while γ_2 discounts across sequence-level decision points. This formulation aggregates rewards over H steps and propagates value estimates across temporally extended transitions, achieving horizon reduction similar to n-step TD learning (Park et al., 2025b) while guaranteeing unbiased value estimates for the H-step rewards (Li et al., 2025a).

2.4.2 DEAS: DEtached value learning with Action Sequence

Detached value learning for handling action sequence Action sequences introduce challenges for value function approximation, as the expanded action space makes it harder for the critic to estimate Q-values accurately. Meanwhile, the actor can exploit regions where the critic makes prediction errors, leading to value overestimation and unstable learning (Seo and Abbeel, 2025). To address this, we adopt detached value learning (Kostrikov et al., 2022; Xu et al., 2023; Garg et al., 2023) that decouples actor and critic training, introducing a critic $Q(s_t, \mathbf{a}_t; \theta)$ and a value $V(s_t; \psi)$ networks following IQL (Kostrikov et al., 2022):

$$\mathcal{L}_V(\psi) = \mathbb{E}_{(s_t, \mathbf{a}_t) \sim \mathcal{D}}\left[L_2^\tau(\bar{Q}(s_t, \mathbf{a}_t; \bar{\theta}) - V(s_t; \psi))\right] \tag{2.5}$$

$$\mathcal{L}_Q(\theta) = \mathbb{E}_{(s_t, \mathbf{a}_t) \sim \mathcal{D}}\left[(\tilde{R}(s_t, \mathbf{a}_t, \gamma_1) + \gamma_2^H V(s_{t+H}; \psi) - Q(s_t, \mathbf{a}_t; \theta))^2\right], \tag{2.6}$$

This approach steers the critic toward high-return action sequences in the offline dataset without the potential of exploiting critic approximation errors, preventing value overestimation and enabling stable value learning even with longer action sequences.

Distributional RL for enhanced stability Even with detached value learning, the cumulative reward term $\hat{R}_{t:t+H-1}$ could introduce significant variance when H is large. To enhance stability, we extend our framework with distributional RL (Bellemare et al., 2017; Farebrother et al., 2024), modeling both critic and value networks as categorical distributions over fixed support $[\mathbf{v}_{\min}, \mathbf{v}_{\max}]$ discretized into m bins:

$$Q(s, \mathbf{a}; \theta) = \mathbb{E}\left[Z(s, \mathbf{a}; \theta)\right] \quad Z(s, \mathbf{a}; \theta) = \sum_{i=1}^{m} \hat{p}_i(s, \mathbf{a}; \theta) \cdot \delta_{z_i} \quad \hat{p}_i(s, \mathbf{a}; \theta) = \frac{e^{l_i(s, \mathbf{a}; \theta)}}{\sum_{i=1}^{m} e^{l_i(s, \mathbf{a}; \theta)}}, \tag{2.7}$$

where $\hat{p}_i$ denotes the predicted probabilities for location z_i. $V(s; \psi)$ is computed similarly, by conditioning only on the state s. To address scale differences between regression and classification objectives, we maintain IQL's weighting scheme but replace regression with classification-based learning:

$$\begin{aligned} \mathcal{L}_V(\psi) &= \mathbb{E}_{(s_t, \mathbf{a}_t) \sim \mathcal{D}} \left[\alpha_t \cdot \sum_{i=1}^{m} \hat{p}_i(s_t; \psi) \log \hat{p}_i(s_t, \mathbf{a}_t; \bar{\theta}) \right] \\ \alpha_t &= \begin{cases} \tau & \text{if } \bar{Q}(s_t, \mathbf{a}_t; \bar{\theta}) \geq V(s_t; \psi) \\ 1 - \tau & \text{otherwise,} \end{cases} \end{aligned} \tag{2.8}$$

$$\mathcal{L}_Q(\theta) = \mathbb{E}_{(s_t, \mathbf{a}_t, s_{t+H}) \sim \mathcal{D}} \left[\sum_{i=1}^{m} p_i(s_t; \psi) \log \hat{p}_i(s_t, \mathbf{a}_t; \theta) \right]. \tag{2.9}$$

For target probabilities p_i, we adopt the truncated normal distribution with mean as Bellman target $(\hat{\mathcal{T}}\mathcal{V})(s, \mathbf{a}_t) = \sum_{k=0}^{H-1} \gamma_1^k r_{t+k} + \gamma_2^H V(s_{t+H}; \psi)$ and standard deviation $\sigma = 0.75 \cdot (v_{max} - v_{min}/m)$, inspired by Farebrother et al. (2024).

Dual discount factors To further enhance stability and expressiveness in value estimation, we employ two separate discount factors: γ_1 for rewards within H-step action sequences and γ_2 for the summation across sequence-level decision points. This dual-discounting scheme enables the value function to appropriately weigh immediate and future returns, mitigating issues such as value explosion or collapse that can arise from improper return scaling. In our experiments, we observe that decreasing γ_1 and increasing γ_2 lead to more stable training and are critical for performance improvement, especially as the action sequence lengthens (see Section 2.5.4 for supporting results).

Compatible policy methods For obtaining final policy $\pi(s; \phi)$, our framework is compatible with a variety of policy extraction strategies (Park et al., 2024), including weighted behavior cloning (Peng et al., 2019), deterministic policy gradient (DPG) (Fujimoto and Gu, 2021), best-of-N sampling (Chen et al., 2023), and flow-matching approaches (Park et al., 2025c). Since value function training does not require querying the policy, it can be performed independently, and the policy can be updated separately. To demonstrate this, we illustrate the effectiveness of our method using various policy extraction methods in our experiments.



Figure 2.2: Simulation task examples. We study DEAS on 30 different tasks from OGBench (Park et al., 2025a) and 4 challenging manipulation tasks from RoboCasa Kitchen (Nasiriany et al., 2024).

2.5 Experiments

We first validate the effectiveness of DEAS through extensive experiments on various complex, long-horizon tasks in OGBench (Park et al., 2025a). Additionally, to prove that DEAS can be naturally plugged into large-scale VLAs for practical applications, we evaluate DEAS by fine-tuning GR00T N1.5 (NVIDIA, 2025a) using offline RL methods on 4 hard tasks from RoboCasa Kitchen (Nasiriany et al., 2024) and also conduct real-world experiments with Franka Emika Research 3 Robot Arm. See Figure 2.2 and Figure 2.4 for task examples used in our experiments.

2.5.1 OGBench Experiments

Setup We evaluate on 6 manipulation environments from OGBench (Park et al., 2025a), each with 5 subtasks. We use datasets ranging from 1M to 100M transitions based on task difficulty. While OGBench is originally designed for offline goal-conditioned RL, we use its single-task variants (‘`singletask`′) for reward-maximizing offline RL. For fair comparison, all methods use identical MLP architectures for actor networks and adopt the same policy extraction approach as FQL (Park et al., 2025c), except for CQN-AS, which uses value function networks as the actor itself through discretization. Action sequence length H is set to 8 for `scene` and `puzzle` tasks, and 4 for `cube` tasks, with $n = H$ is used for n-step FQL.

Baselines We compare against FQL (Park et al., 2025c), a state-of-the-art offline RL method using one-step distillation between flow matching models with different denoising steps, and n-step FQL (Sutton and Barto, 2018), which extends FQL with n-step TD updates for horizon reduction (Park et al., 2025b). While increasing n increases bias in standard offline RL, DEAS explicitly models action sequences while maintaining horizon reduction benefits. We also consider Q-Chunking (QC) (Li et al., 2025a), which uses action chunking for actor-critic training while keeping the interaction between actor and critic, while DEAS uses detached value learning. For fair comparison with ours, we extensively tune QC-FQL hyperparameters to achieve higher performance than the original paper. Lastly, CQN-AS (Seo and Abbeel, 2025), a value-based RL method with action sequence utilizing multi-level critics with iterative discretization, is included as a baseline.

Quantitative results As shown in Table 2.1, DEAS consistently achieves the best performance across all 6 task categories with various dataset sizes. Comparing FQL and N-step FQL, we observe that simply increasing the n-step mostly leads to performance degradation due to bias in standard offline RL, while our detached value learning approach enables stable training with action sequences. Notably, DEAS matches or outperforms QC-FQL across all tasks, demonstrating the effectiveness of our stable value learning in addressing offline RL instability. The method shows particularly strong performance on tasks requiring long-horizon reasoning like `puzzle` and the most challenging tasks (i.e., `cube` – `quadruple`),

Table 2.1: Offline RL results in 6 task categories from OGBench (Park et al., 2025a). We report the success rate (%) and 95% stratified bootstrap confidence interval over 4 runs. **Bold** indicates the values at or above 95% of the best performance.

Task Category	#Data	FQL	N-step FQL	QC-FQL	CQN-AS	**DEAS**
`scene-play-singletask (5 tasks)`	1M	50 ±3	36 ±2	**73** ±2	1 ±1	**76** ±2
`cube-double-play-singletask (5 tasks)`		14 ±2	4 ±2	41 ±3	2 ±1	**48** ±2
`puzzle-3x3-play-singletask (5 tasks)`		44 ±3	36 ±3	62 ±7	0 ±0	**91** ±3
`cube-triple-play-singletask (5 tasks)`	10M	10 ±3	23 ±2	**83** ±4	0 ±0	**82** ±5
`puzzle-4x4-play-singletask (5 tasks)`		32 ±4	19 ±5	69 ±8	0 ±0	**82** ±6
`cube-quadruple-play-singletask (5 tasks)`	100M	17 ±8	36 ±10	45 ±7	0 ±0	**64** ±8

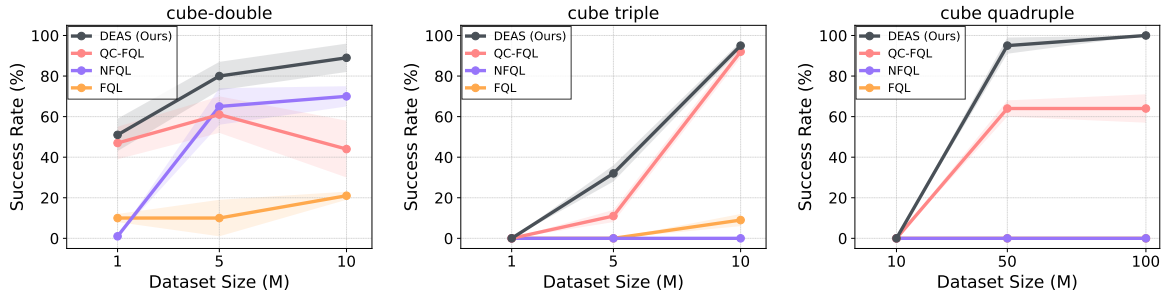


Figure 2.3: Agent performance across varying dataset sizes on three representative OGBench (Park et al., 2025a) tasks, evaluated by success rate (%). Solid lines indicate the mean, while shaded areas denote the stratified bootstrap confidence intervals over 4 independent runs.

where the benefits of using action sequences are most pronounced. CQN-AS shows significantly lower performance, likely due to its direct application of strong BC regularization on the value function in the presence of predominantly suboptimal data, along with cumulative errors from iterative discretizations that reduce action precision.

Scaling analysis To further validate the scalability of DEAS, we conduct a scaling analysis on three representative OGBench tasks with varying dataset sizes. As shown in Figure 2.3, DEAS consistently outperforms all baselines across all dataset sizes, achieving the highest success rates in every environment. The method demonstrates robust scaling across different dataset sizes, maintaining consistent performance gains even with larger datasets. This superior performance validates our approach of explicitly modeling action sequences while effectively leveraging suboptimal data through our detached value learning and stable multi-step training.

2.5.2 VLA Experiments

To validate the practical applicability of DEAS, we demonstrate its effectiveness with large-scale VLAs (Black et al., 2024; NVIDIA, 2025b,a). These models, trained on internet-scale diverse datasets with billion-scale parameters, predict much longer action sequences and are widely used in robotics applications. However, deploying these models typically requires fine-tuning on task-specific data, which often necessitates collecting expensive expert demonstrations. We design our experiments to validate whether DEAS can improve VLA performance by effectively utilizing suboptimal demonstrations along-

Table 2.2: RoboCasa Kitchen evaluation results. We fine-tune VLAs on 24 RoboCasa Kitchen tasks using 100 expert demonstrations per task. For 4 selected tasks, we collect 300 rollouts and apply offline IL/RL algorithms. Success rates (%) on 50 episodes, aggregated with 3 seeds. `PnPC2M` denotes 'PnPCounterToMicrowave' and `PnPM2C` denotes 'PnPMicrowaveToCounter'. **Bold** and underline indicate best and runner-up results, respectively.

* Result from NVIDIA (2025b) † Reproduced performance

Models	`CoffeeSetupMug`	`PnPC2M`	`PnPM2C`	`TurnOffStove`	Avg.
Reference models					
GR00T N1*	2.0	0.0	0.0	15.7	4.4
π_0†	20.0	11.3	10.0	8.0	12.3
Base model					
GR00T N1.5†	4.7	21.3	7.3	14.7	12.0
Imitation learning					
+ Filtered BC	14.7	25.3	<u>14.7</u>	**19.3**	18.5
Offline RL					
+ IQL	<u>23.3</u>	<u>30.0</u>	<u>14.7</u>	12.7	<u>20.2</u>
+ QC	16.0	28.7	<u>14.7</u>	10.7	17.5
+ **DEAS (Ours)**	**28.7**	**36.0**	**18.0**	<u>18.0</u>	**25.2**

side limited expert data, potentially reducing the required amount of costly expert demonstrations.

2.5.2.1 RoboCasa Kitchen Experiments

Setup We employ GR00T N1.5 (NVIDIA, 2025a) as the backbone VLA. First, we fine-tune the VLA using 100 expert demonstrations from all 24 RoboCasa Kitchen tasks to verify that we achieve performance similar to the original GR00T N1 (NVIDIA, 2025b). From these tasks, we select 4 tasks with the lowest success rates in their respective categories for our offline IL/RL experiments. We then collect 300 rollouts for each task from the resulting policy and apply various offline IL/RL methods. For RL methods, we fine-tune the base policy using behavior cloning on both expert demonstrations and the rollout dataset and use the model as an actor for training critic functions when necessary. For policy extraction, we adopt best-of-N sampling (Chen et al., 2023; Nakamoto et al., 2024), where we sample multiple outputs from the policy and select the action sequence with the highest Q-value. We set $H = 16$ for all methods, matching GR00T N1.5's action chunk size.

Baselines We compare against several baselines across both imitation learning and reinforcement learning paradigms. For imitation learning, we consider Filtered BC, which fine-tunes the base policy using both expert demonstrations and successful episodes from the rollout data (Oh et al., 2018). For reinforcement learning, we evaluate IQL, a value-based method that operates on single actions without requiring policy outputs. For determining action sequence in IQL, we use the very first action in the sequence for value estimation. Lastly, we consider QC, which employs action chunking for critic training but relies on predicted action sequences from VLA for the critic update.

Results As shown in Table 2.2, DEAS achieves the highest success rates in 3 out of 4 tasks, with the remaining task also showing improved performance compared to the base model. While filtered BC

Figure 2.4: Real-world tasks. We conduct pick-and-place tasks from the countertop to the bottom cabinet with `peach`, `milka`, and `hichew`.



Table 2.3: Real-world evaluation results. We report the partial success rate (%, over 20 trials per task) on 3 tasks from 5 initial points. **Bold** and underline indicate best and runner-up results, respectively.

Models	`peach`	`milka`	`hichew`	Avg.
Base model				
GR00T N1.5	62.0	<u>45.0</u>	85.0	64.0
Imitation learning				
+ Filtered BC	76.3	25.0	<u>92.5</u>	64.6
Offline RL				
+ IQL	<u>82.5</u>	37.5	78.8	<u>66.3</u>
+ QC	58.8	15.0	45.0	39.6
+ DEAS (Ours)	**86.3**	**53.8**	**95.0**	**78.4**

improves performance with simple approaches, but our approach exhibits additional performance gains by effectively utilizing suboptimal data. While single-step IQL also demonstrates effectiveness, it shows smaller performance gains across all tasks compared to our approach, due to its lack of understanding of action sequences. QC shows limited improvement compared to BC-based approaches, highlighting the advantage of our detached value learning with action sequences.

2.5.2.2 Real-world Experiments

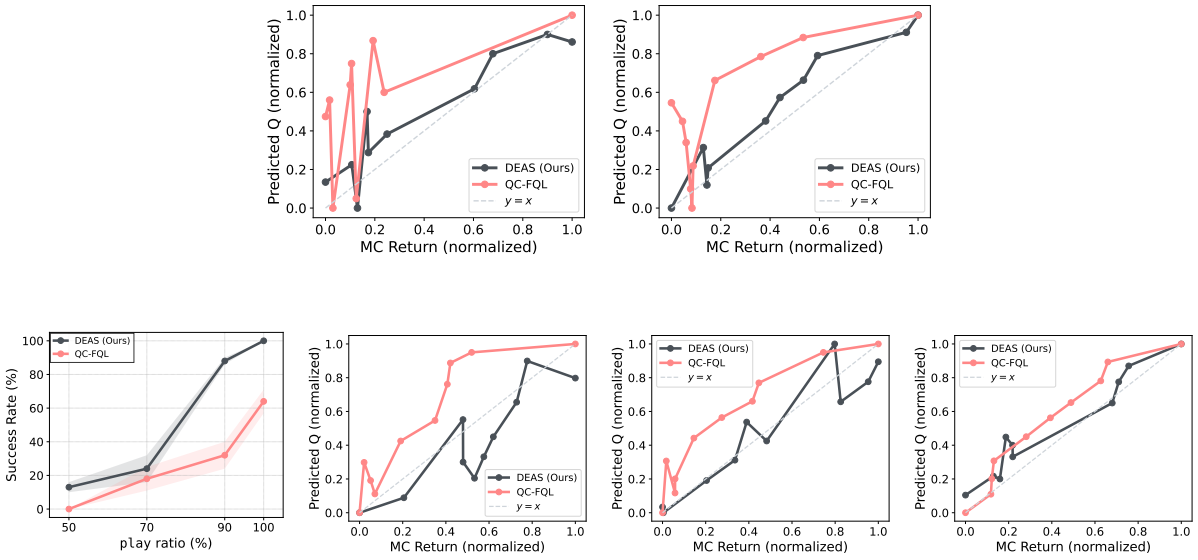
Setup We further investigate the effectiveness of DEAS in real-world tasks using Franka Emika Research 3 Robot Arm. Inspired by RoboCasa Kitchen, we design pick-and-place tasks from the countertop to the bottom cabinet, with three different objects: `peach`, `milka`, and `hichew` (see Figure 2.4). For each task, we collect 5 demonstrations, fine-tune GR00T N1.5, collect 25 rollouts, and apply various offline IL/RL methods. We evaluate using 20 rollouts per task from 5 different initial points and use the same baselines as in the RoboCasa Kitchen experiments. Success rates are calculated based on partial success scoring (0-1 scale) that considers subtask completion.

Results In Table 2.3, DEAS achieves the highest success rates across all three pick-and-place tasks compared to baselines. The method shows consistent improvements, particularly on challenging objects like `milka` (a deformable object) where other approaches struggle. Notably, QC shows degraded performance compared to the base model, likely due to its instability when using action sequences with relatively small datasets, while our method shows stable improvement even with limited data. These results demonstrate that our detached value learning approach can be effectively applied to real-world robotic tasks and remains stable regardless of the dataset size.

2.5.3 Quantitative Analyses

Alignment to actual returns To evaluate whether a critic with action sequences is properly aligned with actual returns, we analyze value calibration between the critic values and the true discounted returns. The main intuition of this experiment is that a properly trained critic should maintain a monotonic relationship between predicted Q values and actual returns on unseen state-action distributions. First, we compute critic predictions $\hat{Q}(s_t, \mathbf{a}_t)$ and Monte-Carlo returns $\hat{G}(s_t, \mathbf{a}_)$ from transitions of 5000 unseen

Figure 2.5: Value calibration curve comparing predicted critic values to Monte-Carlo returns on held-out trajectories in puzzle-4x4 (left) and cube-quadruple (right).



(a) Success Rate (%) **(b)** `play` 50% + `noisy` 50% **(c)** `play` 70% + `noisy` 30% **(d)** `play` 90% + `noisy` 10%

Figure 2.6: Success rate (%) (a) and value calibration curve (b,c,d) across different mixtures of on cube-quadruple task. DEAS exhibits not only higher task performance but also an improved value calibration closer to the diagonal across all data regimes.

trajectories. We then partition transitions into bins by $\hat{Q}$ values and plot the average $\hat{G}$ within each bin. The deviation from the diagonal $y = x$ quantifies overestimation (above) or underestimation (below). As shown in Figure 2.5, DEAS exhibits a calibration curve consistently closer to the diagonal than QC-FQL, indicating reduced overestimation and improved value alignment. These findings support the claim that DEAS's detached value learning framework with distributional objectives provides more reliable value estimation than baselines that incorporate actor-critic interactions in extended, high-dimensional action spaces.

Robustness to data qualities To validate the robustness of DEAS across dataset qualities, we construct mixed datasets by combining `play` and `noisy` datasets provided by OGBench, where `noisy` datasets contain more suboptimal transitions, at different ratios. We measure success rates and analyze calibration curves of DEAS and QC-FQL in Figure 2.6. DEAS outperforms QC-FQL across all data regimes, demonstrating robust performance under varying data quality. Additionally, DEAS's calibration curve remains consistently closer to the diagonal across all regimes, which further supports the effectiveness of our method.

2.5.4 Ablation Studies

We investigate the effect of hyperparameters and various components of DEAS by running experiments on OGBench puzzle-4x4 task.

H	Actor	SR
1	512 × 4	21 ±3
2	512 × 4	25 ±5
4	512 × 4	75 ±8
8	512 × 4	**88** ±4
16	512 × 4	51 ±4
16	1024 × 4	84 ±4

(a) Action sequence

IQL	HLG	SR
✗	✓	75 ±5
✓	✗	63 ±6
✓	✓	**88** ±4

(c) Objectives

Critic	Value	SR
256 × 4	256 × 4	69 ±7
512 × 4	256 × 4	**88** ±4
1024 × 4	256 × 4	**91** ±4
512 × 4	512 × 4	50 ±4

(b) Critic size

γ_1	γ_2	SR
0.8	0.999	**87** ±4
0.9	0.999	**88** ±4
0.99	0.999	81 ±5
0.999	0.999	80 ±8

(d) γ_1 and γ_2

Table 2.4: Ablation studies. We investigate the effect of (a) action sequence length H, (b) critic and value model size, (c) training objectives, and (d) separate discount factors γ_1 and γ_2 for intra-option and inter-option rewards. SR denotes success rate (%) and default settings are highlighted in gray. **Bold** indicates values at or above 95% of the best performance.

Effect of action sequence length Table 2.4a investigates the impact of action sequence length on performance. When using single-step or two-step action ($H = 1, 2$), DEAS fails to achieve meaningful performance, confirming the necessity of action sequences for long-horizon tasks. Performance improves with longer sequences, but when the sequence length becomes longer than 8, it requires proportionally larger actor networks to handle the increased action dimensions, suggesting a trade-off between sequence length and computational efficiency.

Effect of network size Table 2.4b analyzes the sensitivity to network sizes. For the critic network, we observe that increasing capacity initially improves performance by better approximating the value function. For the value function, we find that the network needs sufficient capacity to capture the complexity of action sequence values, but excessive capacity without proper regularization causes instability in value estimation, leading to performance degradation.

Effect of training objective In Table 2.4c, we compare different training objectives for value estimation. We found that using only distributional RL (HLG) (Farebrother et al., 2024) or only standard regression (IQL) shows limited performance. However, combining detached value learning with distributional estimation significantly improves results, suggesting both components are crucial for stable training with action sequences.

Effect of dual discount factors Lastly, we examine the effect of dual discount factors on learning dynamics in Table 2.4d. Proper tuning of γ_1 (the discount factor for action sequences) is essential for performance, as it controls the temporal horizon for value estimation within sequences. In this chapter, we use $\gamma_1 = 0.9$ for all experiments.

2.6 Conclusion

We introduced DEAS, a simple but effective offline RL method that exploits action sequences to learn efficiently on complex tasks. DEAS provides a practical recipe for modeling temporally extended actions while avoiding value overestimation via detached value learning and distributional networks, yielding a principled reduction of the effective planning horizon that is critical in long-horizon settings. Empirically, it consistently outperforms strong baselines on challenging OGBench tasks and scales to large VLAs across simulation and real-world experiments, demonstrating its potential to bring offline RL closer to real-world applications.

Chapter 3. RACE: Calibrated Cumulative Scoring for Rollout-Free Checkpoint Selection

3.1 Introduction

Recent advances in robot learning have enabled manipulation policies to perform an increasingly broad range of tasks across different embodiments (Black et al., 2024, 2025; NVIDIA, 2025b; Gemini Robotics Team et al., 2025b,a; Generalist AI Team, 2026). As these policies are trained and adapted to new settings, it remains difficult to assess how their ability to complete tasks evolves during training. A decreasing training loss does not necessarily identify the checkpoint that will perform best in deployment. Directly measuring task success requires repeated trials on a real robot, and evaluating the many checkpoints produced during training therefore incurs substantial robot time and human effort. This motivates validation criteria that can identify promising checkpoints and focus hardware evaluation on a small set of candidates.

To reduce hardware evaluation costs, surrogate-based approaches evaluate policies in calibrated simulators (Li et al., 2024b; Jain et al., 2025) or learned world models (Li et al., 2025b; Quevedo et al., 2025; Wang et al., 2026; Zhang et al., 2026). However, their usefulness depends on how accurately the surrogate represents the target environment, and constructing or adapting it can require additional data and computation. Rollout-free evaluation offers a complementary approach: a fixed validation set is collected once and reused across checkpoints, whereas deployment trials must be repeated for each evaluated candidate. A common criterion is prediction error on held-out demonstrations, although it can correlate poorly with closed-loop success (Li et al., 2024b; Pan et al., 2026).

This mismatch reflects two limitations. First, smaller action errors do not necessarily imply higher task success: a closed-loop policy can correct some deviations, whereas others prevent task completion. Reducing a recoverable deviation can therefore lower validation loss without increasing success. Second, held-out evaluation continues along the reference trajectory regardless of the policy's previous predictions. After a substantial deviation, later reference observations may no longer reflect the states the policy would encounter (Ross et al., 2011; Spencer et al., 2021; Simchowitz et al., 2025), yet a temporal mean gives their scores equal weight. Off-manifold norm (Pan et al., 2026), a recent alternative, measures deviation from a local expert action manifold but likewise scores predictions pointwise, without accounting for how earlier deviations affect the availability of later execution.

In this chapter, we propose RACE (*Reference-cAlibrated cumulative scoring for rollout-free Checkpoint SElection*), a novel rollout-free metric for ranking policy checkpoints highly aligned with deployment performance using a fixed set of held-out successful trajectories. RACE calibrates action deviations against state-conditional thresholds derived from action differences at similar states in these trajectories. To compare deviations across action groups with different physical units, such as translation and rotation, we measure each group's deviation separately and normalize it by its threshold. Deviations within the threshold receive a group score of one, while larger relative deviations receive lower scores, avoiding rewards for closer imitation within the chosen range. We combine the group scores into timestep scores and form a cumulative score curve through their running product, so that a substantial deviation reduces the contribution of subsequent reference predictions. Averaging this curve over time preserves the contribution of the trajectory prefix before a deviation while giving earlier reductions a

greater effect on the score. Finally, we average the trajectory scores across the reference set, and rank checkpoints in descending order of the resulting score.

Throughout extensive experiments, we demonstrate the effectiveness of RACE in both simulation and real-world settings. In simulation, RACE consistently outperforms the baselines across validation sets drawn from demonstrations and policy rollouts, while several baselines deteriorate sharply when the validation source changes. On real robots, its rankings remain positively correlated with deployment success across the evaluated policy backbones and tasks of varying difficulty. Perturbation analyses further show that RACE distinguishes perturbations with markedly different effects on task success, even when baselines respond similarly.

We highlight our key contributions below:

- We identify and analyze two limitations of aAveraged validation loss for checkpoint selection: it penalizes action deviations without considering whether the policy can recover from them, and temporal averaging ignores how earlier deviations affect the relevance of later reference predictions.
- We propose RACE, a rollout-free checkpoint scoring method that calibrates action deviations using successful reference trajectories, forms cumulative scores through running products, and averages them over time and across trajectories.
- We demonstrate improved checkpoint ranking in standard held-out demonstration evaluation for both simulation and real-world manipulation tasks.

3.2 Related Work

Evaluating robot policies Prior work has sought to reduce the cost of evaluating robot policies on real hardware (Bauer et al., 2022; Atreya et al., 2025; Zhou et al., 2025). Sequential statistical procedures (Snyder et al., 2025, 2026) reduce the number of trials by stopping policy comparisons once sufficient evidence has accumulated, while still relying on rollout outcomes. Simulation-based (Li et al., 2024b; Jain et al., 2025; Zhang et al., 2026) and world-model-based (Li et al., 2025b; Quevedo et al., 2025; Li et al., 2026c) evaluators assess candidates through interactions with a surrogate environment. Hybrid real–sim approaches combine paired hardware and simulation evaluations through control variates (Luo et al., 2025) or prediction-powered inference (Badithela et al., 2025) to obtain confidence intervals on mean real-world performance with fewer hardware trials. These methods still rely on rollout outcomes from the policies being evaluated. RACE ranks checkpoints from their predictions on a fixed set of held-out successful trajectories, requiring no additional hardware or surrogate rollouts during scoring. The same reference set is reused across candidates to support shortlisting before deployment evaluation.

Rollout-free policy evaluation Off-policy evaluation (OPE) estimates a policy's expected return from logged data, using learned value functions (Le et al., 2019; Paine et al., 2020) or importance-sampling corrections (Precup et al., 2000; Jiang and Li, 2016). These approaches use rewards in the logged data to estimate return. Our setting targets relative checkpoint ranking from successful reference trajectories, without per-step reward annotations or a learned return estimator.

Reward-free model selection has also been studied for imitation learning (Hussenot et al., 2021). Validation action MSE is a common criterion, yet it can correlate poorly with deployment performance (Li et al., 2024b; Pan et al., 2026). It penalizes action discrepancies without considering whether the policy can recover from them. Off-manifold norm (Pan et al., 2026) measures the component of a predicted

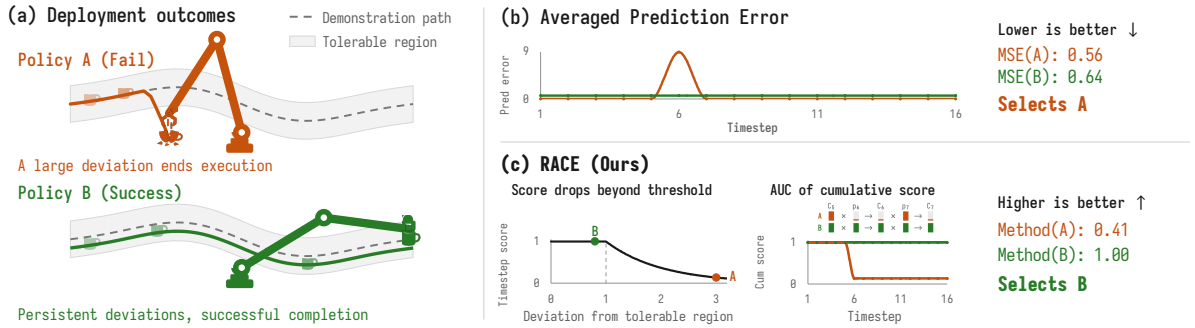


Figure 3.1: Motivation and key idea of RACE. (a) Both policies are evaluated on the held-out demonstration. Policy π_A deviates from the demonstrated route and breaks the cup, whereas π_B reaches the destination despite persistent smaller deviations. (b) Averaged prediction error dilutes the large deviation of π_A with smaller errors elsewhere, giving it a lower mean squared error than π_B. (c) Deviations within a reference-derived threshold receive a timestep score of one; larger deviations receive lower scores. Running products reduce the contribution of later predictions, as illustrated by two successive updates in the small bars. Averaging the cumulative scores over time correctly selects π_B. Higher scores are preferred.

action outside a local subspace formed by expert actions at nearby states. Both criteria score predictions at recorded observations without adjusting the contribution of later predictions for earlier deviations. RACE derives thresholds from action differences at similar states in successful reference trajectories and converts deviations beyond these thresholds into lower timestep scores. The running product of these scores forms a cumulative score curve, reducing the contribution of later reference predictions after substantial earlier deviations. We rank checkpoints by averaging cumulative scores over time within each trajectory and then across reference trajectories.

3.3 Main Method

In this section, we introduce RACE, a novel rollout-free validation metric for policy checkpoint selection highly correlated with deployment performance. We first formulate the checkpoint selection problem (Section 3.3.1), then illustrate the limitations of averaged prediction error through a motivating example (Section 3.3.2), and finally present the scoring method (Section 3.3.3).

3.3.1 Problem Formulation

Training a policy π typically yields a sequence of N_{ckpt} checkpoints $\Pi = \{\pi_1, \ldots, \pi_{N_{\text{ckpt}}}\}$ saved at successive gradient steps. Each checkpoint's quality is typically measured by its *closed-loop success rate*:

$$J(\pi) = \Pr_{\rho_0, \pi}\left[\exists t \le T : \; s_t \in \mathcal{S}_{\text{success}}\right],$$

where ρ_0 is the initial-state distribution, T is the episode horizon, and S_{success} indicates the set of successful states. Because $J(\pi)$ can only be estimated through deployment at environments, evaluating every π scales poorly with $|\Pi|$ and may be unsafe on real hardware. We therefore study *rollout-free checkpoint selection*: given a held-out set of successful reference trajectories $\mathcal{D}_{\text{val}} = \{\tau^{(j)}\}_{j=1}^{N_{\text{val}}}$, we seek a scoring function $S(\pi; \mathcal{D}_{\text{val}})$ that ranks candidates without executing rollouts, using only each policy's predictions on $\mathcal{D}_{\text{val}}$, such that its induced ranking is highly correlated with the deployment ranking under J. The reference trajectories may be demonstrations or previously collected successful policy rollouts.

3.3.2 Pitfall of Averaged Prediction Error

A common rollout-free criterion for checkpoint selection is the averaged prediction error on held-out demonstrations (Li et al., 2024b; Pan et al., 2026). Given a policy π and validation set $\mathcal{D}_{\text{val}}$, let $\ell_t = \mathcal{L}(\pi(o_t), a_t)$ denote the per-step prediction error at timestep t, where $\mathcal{L}$ is a distance metric matched to the policy's output space (*e.g.* MSE for deterministic policies, flow-matching or negative log-likelihood for generative ones). it averages errors within each trajectory and then across trajectories, giving each demonstration equal weight. The criterion selects the checkpoint maximizing:

$$S_{\text{avg}}(\pi; \mathcal{D}_{\text{val}}) = -\frac{1}{N_{\text{val}}} \sum_{j=1}^{N_{\text{val}}} \left[\frac{1}{T_j} \sum_{t=1}^{T_j} \ell_t^{(j)} \right],$$

where T_j denotes the length of trajectory j.

To illustrate the limitation, we provide two hypothetical policies evaluated on the same held-out demonstration (Figure 3.1). Policy π_A closely reproduces the reference actions at most timesteps, but one of its actions causes task failure during execution (*e.g.*, dropping the object beyond the robot's reach). Policy π_B differs more frequently from the reference actions at many timesteps, but it resides in tolerable region and completes the task. Despite these contrasting outcomes, π_A can obtain a lower average prediction error because its many small errors dilute the isolated large deviation. The mean-error criterion would then misleadingly select π_A over π_B.

These observations suggest two considerations for designing rollout-free evaluation criteria. First, action prediction errors need not translate directly into task failure. A closed-loop policy may correct some deviations through subsequent actions, whereas other deviations can prevent task completion. Reducing an error from which the policy can recover therefore need not improve its success rate, even though it lowers the validation loss. This motivates calibrating prediction errors before aggregating them, rather than treating every numerical reduction as an improvement in checkpoint quality. Second, averaged prediction error scores each prediction at a recorded demonstration observation, regardless of the policy's earlier predictions. If an earlier deviation would cause execution to depart from the reference, predictions at subsequent timesteps may no longer describe the policy's closed-loop behavior. This motivates reducing the contribution of later validation steps following a substantial deviation.

3.3.3 RACE

RACE evaluates checkpoints in two stages (Figure 3.2). First, state-conditional calibration compares action deviations against thresholds derived from reference actions at similar states, yielding timestep scores in $[0, 1]$ (Section 3.3.3.1). Second, temporal aggregation forms running products of these scores, so that a low score reduces the contribution of later reference predictions. Averaging the resulting curves over time and across trajectories gives the checkpoint score (Section 3.3.3.2).

3.3.3.1 State-Conditional Calibration

We first describe how to assign a timestep score $p_t \in [0, 1]$ to a single predicted action chunk. At observation o_t, let $\mathbf{a}_t = (a_t, \ldots, a_{t+H-1})$ denote the reference action chunk of length H, and let $\hat{\mathbf{a}}_t$ denote a chunk predicted by the policy.

Group distances As action components differ in physical units and geometry, computing a single Euclidean distance over the full action vector can therefore overemphasize components with larger numerical scales and misrepresent rotational differences. To resolve this, we first restore actions to their

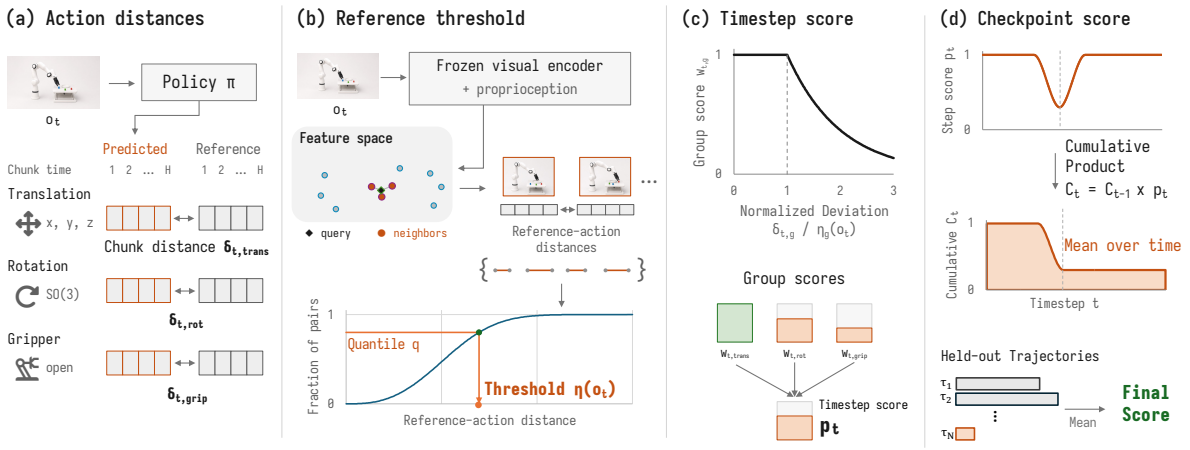


Figure 3.2: RACE scoring procedure. (a) We compare predicted and reference action chunks within each action group to obtain $\delta_{t,g}$. (b) Distances between eligible pairs of reference chunks define a distribution whose quantile at level q gives the state-conditional threshold $\eta_g(o_t)$. The representation, neighborhoods, and thresholds are shared across candidate checkpoints. Both comparisons use the same group distance D_g; the illustrated configuration averages the largest 50% of within-chunk timestep distances in physical units. (c) Threshold-normalized deviations yield group scores, which a soft minimum combines into $p_t \in [0, 1]$. (d) We form running products of the timestep scores, then average over time and across trajectories to rank checkpoints. Higher scores are preferred.

physical units and partition them into G groups, such as translation and rotation. We compute distances separately for each group, using Euclidean distance for Euclidean components and the geodesic angle on SO(3) for rotations. For chunked predictions, we aggregate these distances over the prediction horizon. Averaging all timestep distances can dilute a brief but large deviation. We therefore define D_g as the mean of the largest 50% of the timestep distances within group g, giving greater weight to larger deviations. The predicted chunk's deviation in group g is

$$\delta_{t,g} = D_g(\hat{\mathbf{a}}_t, \mathbf{a}_t)\,.$$

Reference thresholds To determine which deviations to penalize, we use differences between actions taken at similar states in successful reference trajectories. Our design treats deviations within a chosen range of these differences as acceptable and penalizes larger deviations. Because this range can vary across states, we introduce state-conditioned threshold: for each observation o_t, we retrieve the k nearest reference observations in $\mathcal{D}_{\text{val}}$ using frozen visual features and proprioception, then compute pairwise distances between their action chunks using the same D_g.[1] We set $\eta_g(o_t)$ to the q-quantile of these distances. The quantile level q controls the coverage of this reference range: larger values yield higher thresholds and penalize only larger deviations. The resulting threshold is tighter where nearby reference actions are more consistent (*e.g.* fine-grained placement) and looser where they differ more (*e.g.,* free-space movement).

Timestep scores To compare deviations across action groups on a common relative scale, we divide each group deviation by its own threshold. Deviations within the threshold receive a score of one, so closer imitation within this range does not improve the score. Beyond the threshold, an exponential

[1] For joint-velocity control, pairs are restricted to the same reference trajectory; for task-space control, pairs may also span trajectories.

decay penalizes the relative excess:

$$w_{t,g} = \exp\left(-\left[\frac{\delta_{t,g}}{\eta_g(o_t)} - 1\right]_+\right),$$

where $[x]_+ = \max(x, 0)$.

We combine the group scores using a normalized soft minimum. This aggregation is symmetric across groups while giving greater influence to lower scores from any group:

$$p_t = -\log\left(\frac{1}{G}\sum_{g=1}^{G}\exp(-w_{t,g})\right).$$

We can also evaluate M action chunks predicted at the same observation. We score each chunk using the same reference and thresholds, then average the resulting scores as $p_t = \frac{1}{M}\sum_{m=1}^{M} p_t^{(m)}$, where $p_t^{(m)}$ is the score of the m-th chunk. Scoring each chunk before averaging preserves the effect of deviations above the threshold that averaging raw distances may conceal. Empirically, averaging over multiple chunks yields more stable scores and improves checkpoint-ranking accuracy compared with a single sample.

3.3.3.2 Temporal Aggregation

To reduce the contribution of later predictions after substantial deviations, we form the running product of the timestep scores. For trajectory $\tau^{(j)}$, we initialize $C_0^{(j)} = 1$ and define

$$C_t^{(j)} = C_{t-1}^{(j)} p_t^{(j)} = \prod_{u=1}^{t} p_u^{(j)}.$$

A timestep score of one leaves the cumulative value unchanged, whereas a lower score reduces it from that point onward. Repeated low scores compound this reduction. We score each trajectory by its mean cumulative score over the T_j evaluated timesteps, equivalently the normalized area under the cumulative score curve (AUC):

$$S_\tau^{(j)} = \frac{1}{T_j}\sum_{t=1}^{T_j} C_t^{(j)}.$$

An earlier decline lowers more terms in this mean, while the preceding trajectory prefix retains its contribution.

Finally, we get the final score $S(\pi; \mathcal{D}_{\text{val}})$ by averaging the trajectory scores over the held-out set:

$$S(\pi; \mathcal{D}_{\text{val}}) = \frac{1}{N_{\text{val}}}\sum_{j=1}^{N_{\text{val}}} S_\tau^{(j)}.$$

For the candidate set Π, we compute $\{S(\pi; \mathcal{D}_{\text{val}})\}_{\pi\in\Pi}$ and rank checkpoints in descending order of their scores.

3.4 Experiments

We design our experiments to answer three questions:

- Can RACE rank checkpoints more reliably from held-out demonstrations, and retain useful rankings with other successful reference sources? (Section 3.4.1)
- Does it support real-robot checkpoint selection across policy families and task demands? (Section 3.4.2)

Table 3.1: Rollout-free checkpoint ranking on DexMimicGen. All criteria utilizes the same 10 successful trajectories per source to rank 12 checkpoints per run. Entries are means with 95% hierarchical bootstrap intervals, averaged over 10 training seeds per task and then eight tasks. Arrows indicate better values; bold marks the best mean per metric within each source. See Section 3.4 for the evaluation protocol.

Validation source	Criterion	Hit@1↑	Hit@3↑	Hit@5↑	ρ ↑	MMRV↓	nRegret↓
Held-out Demos	RACE	**0.44** [0.23,0.65]	**0.75** [0.59,0.90]	**0.95** [0.88,1.00]	**0.83** [0.72,0.92]	**0.030** [0.012,0.053]	**0.082** [0.035,0.142]
	Validation Loss	0.31 [0.12,0.53]	0.59 [0.33,0.82]	0.75 [0.49,0.97]	0.52 [0.01,0.93]	0.042 [0.011,0.088]	0.238 [0.030,0.504]
	Validation MSE	0.00 [0.00,0.00]	0.08 [0.00,0.17]	0.20 [0.04,0.40]	−0.22 [−0.62,0.20]	0.122 [0.068,0.180]	0.495 [0.325,0.660]
	Off-Manifold Norm	0.25 [0.11,0.40]	0.54 [0.33,0.74]	0.72 [0.46,0.94]	0.60 [0.22,0.87]	0.042 [0.022,0.067]	0.146 [0.052,0.288]
Rollouts	RACE	**0.27** [0.12,0.45]	**0.62** [0.38,0.83]	**0.90** [0.77,1.00]	**0.79** [0.69,0.88]	**0.034** [0.021,0.047]	**0.113** [0.049,0.186]
	Validation Loss	0.03 [0.00,0.10]	0.12 [0.02,0.27]	0.15 [0.03,0.30]	−0.10 [−0.45,0.25]	0.136 [0.086,0.190]	0.402 [0.298,0.514]
	Validation MSE	0.08 [0.02,0.18]	0.22 [0.05,0.40]	0.30 [0.12,0.50]	−0.06 [−0.27,0.16]	0.129 [0.084,0.177]	0.480 [0.343,0.633]
	Off-Manifold Norm	0.17 [0.07,0.27]	0.37 [0.23,0.50]	0.55 [0.42,0.68]	0.53 [0.39,0.66]	0.068 [0.047,0.094]	0.164 [0.123,0.207]
Noised Rollouts	RACE	**0.32** [0.19,0.49]	**0.69** [0.52,0.85]	**0.91** [0.81,0.99]	**0.78** [0.68,0.86]	**0.035** [0.021,0.053]	**0.081** [0.053,0.116]
	Validation Loss	0.00 [0.00,0.00]	0.04 [0.00,0.11]	0.09 [0.01,0.19]	−0.47 [−0.68,−0.15]	0.154 [0.105,0.202]	0.613 [0.462,0.773]
	Validation MSE	0.09 [0.00,0.21]	0.29 [0.11,0.49]	0.39 [0.17,0.61]	0.30 [−0.05,0.61]	0.083 [0.041,0.130]	0.288 [0.148,0.443]
	Off-Manifold Norm	0.06 [0.01,0.14]	0.31 [0.16,0.47]	0.42 [0.25,0.59]	0.30 [0.05,0.52]	0.085 [0.055,0.122]	0.276 [0.185,0.405]

- Which scoring choices contribute to ranking quality, and how do changes in successful references affect it? (Section 3.4.3)

Baselines We compare three criteria on held-out demonstrations: **Validation Loss**, the behavior-cloning training objective, **Validation MSE**, the mean squared action prediction error independent of the training objectives, and **Off-Manifold Norm** (Pan et al., 2026), the residual norm after projection onto a local basis formed from the k nearest neighbors. For visibility, we negate all baseline values so higher scores are preferred like our proposed score.

Metrics Let y_i denote the empirical success rate of checkpoint i and s_i its offline criterion score (higher is better). **Spearman correlation** ρ measures agreement between the rankings induced by s_i and y_i. **Hit@K** **(**$K \in \{1, 3, 5\}$**)** is the fraction of runs in which the empirically best checkpoint is among the criterion's top-K candidates. **nRegret** measures the success-rate gap between the empirically best checkpoint and the criterion's highest-scoring checkpoint, normalized by the best-minus-worst success-rate range. **Mean Maximum Rank Violation (MMRV)** averages each checkpoint's largest success-rate gap among its pairwise ranking disagreements, assigning zero when none occur. Thus, ranking errors involving larger performance gaps incur greater penalties. We refer to Li et al. (2024b) for the formal definition.

3.4.1 Simulation Experiments

Setup We evaluate eight DexMimicGen (Jiang et al., 2025) tasks, excluding Transport. The action partition follows the robot embodiment and its control interface. For bimanual Franka with parallel grippers (*Threading* and *Three Piece Assembly*), each arm contributes separate end-effector translation

and rotation groups, giving $G = 4$, and binary gripper commands are excluded. For bimanual Franka with dexterous hands (*Box Cleanup*, *Lift Tray*, and *Drawer Cleanup*), each hand's absolute joint-position vector forms one additional group, giving $G = 6$. For GR-1 (*Pouring*, *Coffee*, and *Can Sort*), we group the absolute joint positions of each arm and hand separately, giving $G = 4$. Translation and joint-position vectors use Euclidean distance, while rotations use the geodesic angle on SO(3).

For each task, we train flow-matching `Chi-UNet` policies (Chi et al., 2025; Pan et al., 2026) using behavior cloning on 500 demonstrations following RoboMimic (Mandlekar et al., 2021), aggregated with 10 independent training seeds. Each run yields 12 checkpoints saved every 50 epochs over 600 epochs. Each checkpoint's empirical success rate is the mean of its own 200 deployment rollouts. To test each metric under varying degrees of distribution shift, we construct a separate validation set of 10 successful trajectories for each of the following three sources: (i) **Held-out Demonstrations** are from expert trajectories matching the training distribution exactly but not used for training. (ii) **Rollouts** denote successful closed-loop trajectories from the separately trained policy, introducing mild shift through the policy's own state visitation, and (iii) **Noised Rollouts** are collected using successful rollouts from the same policy perturbed with Gaussian action noise of standard deviation 0.02, clipped to ± 0.05, representing the largest shift.

For each reference source, we compute metrics per training run, average the 10 seeds within each task, and then average the task means with equal weights. We draw 10,000 hierarchical bootstrap resamples: each resample draws tasks with replacement, then training seeds with replacement within each sampled task, and recomputes the same aggregate. The notation $[L, U]$ denotes the lower and upper bounds of the resulting 95% bootstrap interval.

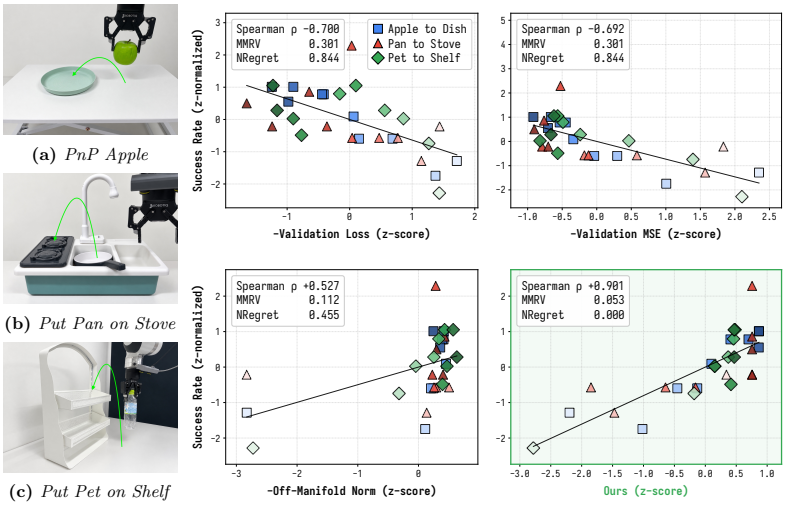
Results As shown in Table 3.1, RACE achieves the strongest ranking quality across all metrics and configurations. Crucially, RACE maintains stable scores as the validation distribution shifts from clean demonstrations to noisy rollouts, whereas baselines like Validation Loss and Off Manifold Norm collapse significantly under distribution shift and exhibit negative correlation values. These results confirm that RACE provides a reliable ranking signal regardless of how the validation trajectories are collected, a critical property for real-robot workflows where clean demonstrations are scarce.

3.4.2 Real-World Experiments

To validate that RACE generalizes beyond simulation, we evaluate it on real-robot manipulation tasks using two policy architectures of different scales: a task-specific diffusion policy trained from scratch and a large pretrained VLA fine-tuned on in-domain data. All experiments use a Franka Research 3 arm following DROID (Khazatsky et al., 2024) hardware and software setup, with task-specific demonstrations collected via teleoperation. In both settings, held-out demonstrations unseen during training serve as the validation set, and baselines and ranking metrics follow the same protocol as in Section 3.4.1. To support reliable empirical comparison, we evaluate each of 10 checkpoints with 30 deployment rollouts on each of three tasks, yielding $30 \times 10 \times 3 = 900$ real-robot evaluations per policy family and 1,800 in total.

Diffusion Policy Experiments

Setup We fine-tune DROID Diffusion Policy (Khazatsky et al., 2024) with pre-trained ResNet-50 encoder using two exterior left camera views. The policy is co-trained with the DROID dataset for

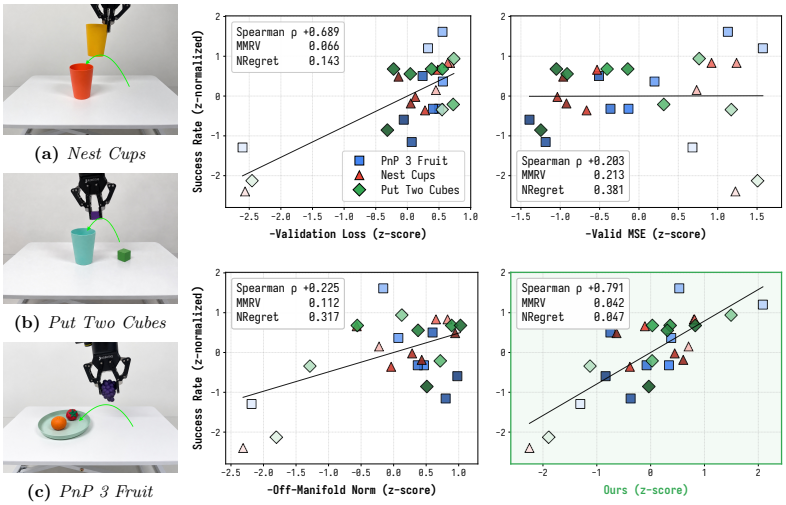


(a) *PnP Apple*

(b) *Put Pan on Stove*

(c) *Put Pet on Shelf*

Figure 3.3: Diffusion Policy experiments. (Left) task setups. (Right) z-normalized criterion scores versus z-normalized empirical success rates, with one marker per checkpoint; marker shape and color distinguish tasks. Each task uses 20 training and 30 held-out validation demonstrations, 10 candidate checkpoints, and 30 deployment rollouts per checkpoint (900 rollouts across the three tasks). RACE uses task-specific scoring settings. Panel annotations report point estimates of Spearman $\rho \uparrow$, MMRV$\downarrow$, and NRegret$\downarrow$.



(a) *Nest Cups*

(b) *Put Two Cubes*

(c) *PnP 3 Fruit*

Figure 3.4: $\pi_{0.5}$-DROID experiments. (Left) task setups. (Right) z-normalized criterion scores versus z-normalized empirical success rates, with one marker per checkpoint; marker shape and color distinguish tasks. Policies use joint-velocity control and are fine-tuned on 20 target-task training demonstrations per task. Each task uses 30 held-out validation demonstrations, 10 candidate checkpoints, and 30 deployment rollouts per checkpoint (900 rollouts across the three tasks). RACE uses task-specific scoring settings. Panel annotations report point estimates of Spearman $\rho \uparrow$, MMRV$\downarrow$, and NRegret$\downarrow$.

50 epochs on 20 task-specific demonstrations and checkpointed every 5 epochs. We evaluate on three tabletop tasks—*PnP Apple*, *Put Pan on Stove*, and *Put Pet on Shelf*—each validated against 30 held-out demonstrations. The policy predicts relative end-effector translation and rotation, with rotations represented in 6D (Zhou et al., 2019), together with a binary gripper command. For scoring, translation and rotation form two separate groups ($G = 2$), and the gripper is excluded. We use Euclidean distance for translation and convert the 6D rotation representation to SO(3) before computing the geodesic angle.

Results As shown in Figure 3.3, standard baselines struggle in this setting: Validation Loss and Validation MSE produces highly negative correlation values and reverses the true checkpoint ranking. While Off-Manifold Norm recovers moderate correlation, this result is mainly due to the separation of early checkpoints, and all other checkpoints are indistinguishable. In contrast, RACE consistently ranks checkpoints in alignment with deployment success across all three tasks, demonstrating its robustness and reliability for real-robot checkpoint selection.

VLA Experiments

Setup To test RACE on longer-horizon tasks that require precise sequential manipulation, we fine-tune $\pi_{0.5}$-DROID (Black et al., 2025) using wrist camera and a single exterior left camera view. The policy uses 7-DoF joint-velocity control. We train and evaluate on 3 tasks with varying difficulties: *Nest Cups*, *Put Two Cubes*, and *PnP Three Fruit*. All tasks are trained with 20 demonstrations and validated against 30 held-out demonstrations. For scoring, each arm joint forms a separate scalar group ($G = 7$), with the absolute difference between predicted and reference joint velocities as its distance. The binary gripper command is excluded.

Results As shown in Figure 3.4, the baselines recover moderate correlation in this setting, yet RACE leads by a clear margin. The gap is most pronounced on *Put Two Cubes* and *PnP Three Fruit*, where longer horizons amplify compounding errors that one-step losses fail to capture. We hypothesize that baselines perform better here than in the Diffusion Policy setting because $\pi_{0.5}$'s large-scale pretraining provides well-structured representations, ensuring that validation loss tracks genuine policy improvement rather than overfitting. Nevertheless, RACE still outperforms all baselines by a consistent margin and, across both architectures, reliably selects near-optimal checkpoints on real hardware with irreducible variables from human operators and sensor noise.

3.4.3 Analyses and Ablation Studies

Perturbations with similar prediction error The motivating example in Section 3.3.2 raises a practical question: can similar prediction errors accompany different deployment consequences? On *Box Cleanup*, we perturb one trained policy uniformly ($\sigma = 0.05$) or at bottlenecks (*i.e.,* robot aligning the box lid to the box) ($\sigma = 0.2$), matching average prediction error and evaluating 50 paired rollouts from identical initial states. Despite similar MSE and Off-Manifold Norm changes, bottleneck perturbations degrade success and RACE scores more strongly (Table 3.2). This supports the complete score's sensitivity to consequential deviations. Success-difference ranges use paired resampling; offline metric changes are point estimates.

Table 3.2: Perturbation Analysis We apply different scale of perturbations on trained policies for *Box Cleanup* tasks, and compared changes relative to the original policy using 50 fixed scenes.

Variant	ΔSuccess	ΔMSE	ΔOff-Man.	ΔRACE
Uniform (σ=0.05)	-4.0%P	-2.46%	-2.51%	-1.42%
Bottleneck (σ=0.2)	-40.0%P	-2.47%	-2.00%	-52.69%

Component ablation To identify which design choices contribute to ranking, we remove action grouping (Grp), threshold scoring (Thr), or state-conditional adaptation (SC) on *Coffee*, *Drawer Cleanup*,

Table 3.3: Component and temporal ablations across four tasks. (a) Removing calibration threshold uses $q = 1.0$ for all groups, and removing state conditioning fixes the threshold across states. (b) Both reductions use identical timestep scores. Means are shown above 95% hierarchical bootstrap intervals. Bold marks the best value per column within each panel.

(a) Component ablation

Setting	$\rho \uparrow$	MMRV↓	nRegret↓
Full	**0.76** [0.66,0.86]	**0.040** [0.02,0.07]	**0.096** [0.05,0.15]
w/o action group	0.69 [0.56,0.81]	0.048 [0.03,0.07]	0.148 [0.09,0.21]
w/o calib. threshold	0.31 [0.11,0.52]	0.093 [0.05,0.13]	0.363 [0.29,0.44]
w/o state cond.	0.42 [0.20,0.68]	0.083 [0.03,0.14]	0.334 [0.19,0.47]

(b) Temporal aggregation

Setting	$\rho \uparrow$	MMRV↓	nRegret↓
Mean	0.63 [0.40,0.78]	0.052 [0.02,0.10]	0.180 [0.09,0.33]
Cumulative scoring	**0.76** [0.66,0.86]	**0.040** [0.02,0.07]	**0.096** [0.05,0.15]

Pouring, and *Three Piece Assembly*. Each removal lowers aggregate ranking quality (Table 3.3(a)), supporting comparisons that account for heterogeneous action units and calibrate deviations against state-relative reference variation.

Temporal aggregation The perturbation comparison evaluates the complete score; to separate temporal aggregation from calibration, we compare the per-trajectory mean of identical stored p_t values with cumulative scoring (Section 3.3.3.2), which averages their running products over time. Both methods then average trajectory scores with equal weights. Candidates, references, thresholds, sampled predictions, and deployment outcomes are fixed. Across four DexMimicGen tasks, cumulative scoring improves ranking correlation and lowers selection regret compared with the mean (Table 3.3(b)). This supports reducing the contribution of later reference predictions after a substantial deviation, beyond calibrating individual timestep scores.

Table 3.4: Task-specific versus shared settings. Means over 24 task/reference-source combinations across eight tasks, with 95% hierarchical bootstrap intervals shown below. Shared (task k) varies only neighborhood size k. Bold marks the best value per column.

Setting	$\rho \uparrow$	MMRV↓	nRegret↓
Tuned	**0.80** [0.74,0.86]	**0.033** [0.02,0.05]	**0.090** [0.06,0.12]
Shared (task k)	0.75 [0.67,0.81]	0.037 [0.03,0.05]	0.110 [0.08,0.15]
Shared	0.69 [0.57,0.78]	0.038 [0.03,0.05]	0.135 [0.10,0.17]

Shared scoring settings To assess tuning burden, we compare task-specific (Tuned), fully Shared, and Shared settings with task-specific neighborhood size k across the 22 task/reference-source combinations spanning the eight DexMimicGen tasks. Fully Shared settings reduce ranking quality relative to Tuned (Table 3.4). Allowing only k to vary by task recovers part of the gap, indicating that adapting neighborhood size provides some of the benefit of task-specific tuning while sharing the remaining settings.

Augmenting with rollouts On *PnP Apple*, supplementing held-out demonstrations with successful final-policy rollouts improves checkpoint ranking (Table 3.5(a)). This suggests that successful policy rollouts can complement held-out demonstrations as reference data, improving selection without changing the scoring rule.

Table 3.5: Real-world reference-set analyses. (a) Adding successful final-policy rollouts to held-out demonstrations on *PnP Apple*. (b) Using successful policy rollouts containing a random 3–5 second pause. ρ denotes Spearman correlation in (a) and Pearson correlation in (b). Values are point estimates.

(a) Rollout augmentation

Reference set	$\rho \uparrow$	MMRV$\downarrow$
Held-out	0.834	0.230
+ Rollouts	**0.920**	**0.070**

(b) Temporal artifacts

Reference set	$\rho \uparrow$	MMRV$\downarrow$
Held-out	0.834	0.230
Paused Rollouts	0.801	0.194

Robustness to temporal artifacts To examine reference trajectories containing idle segments, we collect successful policy rollouts with a random 3–5 second pause at a randomly chosen point in each rollout (Table 3.5(b)). Checkpoint rankings remain comparable to those obtained with held-out demonstrations, with slightly lower correlation but lower MMRV. These results suggest that successful references containing idle segments can still provide useful information for checkpoint selection.

3.5 Conclusion

We present RACE, a novel rollout-free checkpoint selection metric that converts action deviations from successful reference trajectories into a normalized score. RACE provides a practical way to compare robot policy checkpoints before deployment using a reusable set of successful trajectories. Our simulation and real-robot results show that RACE rankings align more closely with task success than those based on averaged prediction error. This can support more informed decisions during policy development, from tracking progress across checkpoints to focusing limited robot evaluation on promising candidates.

Despite of its advantages, RACE has two limitations. First, even though action variation at nearby successful states is a proxy for tolerable deviation, but it cannot establish whether a candidate policy would recover from an error or predict its behavior at unseen states induced by its own actions. Second, RACE provides a relative ranking, not a calibrated success-rate estimate, and selected candidates still require deployment validation.

Chapter 4. ACCRUE: Continual Off-Policy Reinforcement Learning for Robot Foundation Models

4.1 Introduction

Robot Foundation Models (RFMs), including vision-language-action models (VLAs) (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024; NVIDIA, 2025b) and world action models (WAMs) (Ye et al., 2026; Dyna Robotics, 2026; Zhu et al., 2025; Pai et al., 2025), 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 (Li et al., 2025c; Xiao et al., 2026; Li et al., 2026a; Xu et al., 2026; Dong et al., 2026). 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 (Figure 4.1).

Recent methods use RL to adapt VLAs across sequential tasks and mitigate forgetting (Hu et al., 2026; Liu et al., 2026b; Zeng et al., 2026), 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 (Xu et al., 2026) and EXPO-FT (Dong et al., 2026), which attach lightweight actor and critic networks to frozen RFMs, improve single-task sample efficiency but do not address learner persistence across tasks.

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 retained episodes may not provide sufficient coverage for reliable value estimation

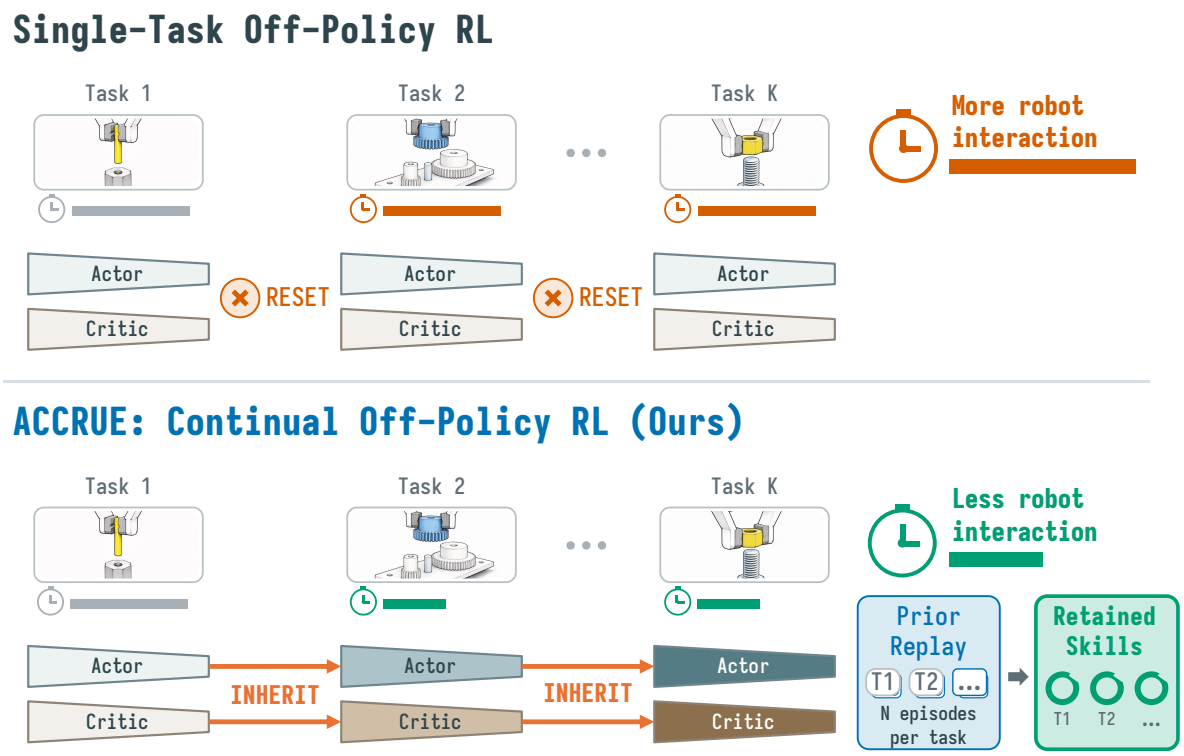


Figure 4.1: Single-task vs Continual off-policy RL. Single-task RL resets the actor and critic for each task, requiring repeated online interactions. Our method inherits both networks and uses small prior-task replay to accelerate learning while retaining prior performance.

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.

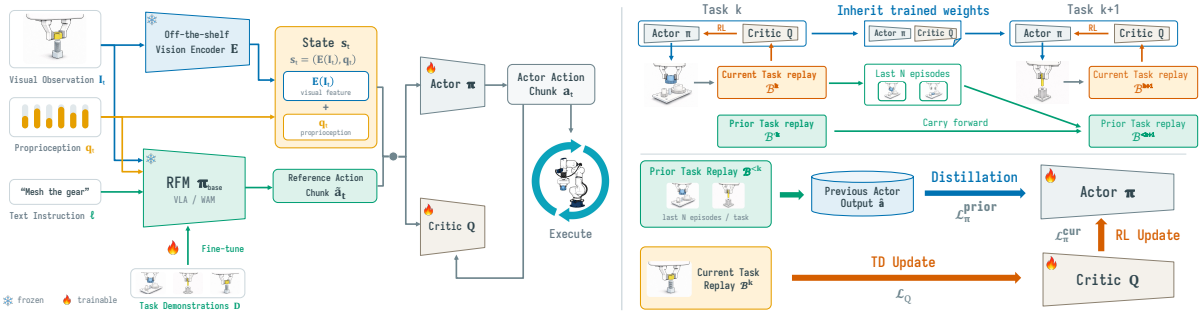


Figure 4.2: Overview of ACCRUE. (Left) Base policy is fine-tuned on demonstrations observed up to current stage, held fixed, and proposes reference action chunks that actor and critic networks refine through off-policy RL on frozen encoder features. (Right Top) Both networks are carried across stages 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.

4.2 Related Work

4.2.1 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 (Guo et al., 2025; Li et al., 2026a). Other approaches use offline value learning, task-specific residual policies, or RL-generated data to improve a generalist policy (Kim et al., 2026a; Li et al., 2025c; Xiao et al., 2026). Most closely related to our architecture, lightweight off-policy methods refine a frozen policy through action- or latent-space actor–critic learning (Yuan et al., 2025; Wagenmaker et al., 2025; Xu et al., 2026; Dong et al., 2026), 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.

4.2.2 Continual Learning for Robotics

Continual robot learning (Liu et al., 2023; Wan et al., 2024) has largely focused on preserving demonstrated behavior through experience replay (Liu et al., 2026a; Du et al., 2026), hierarchical skill libraries (Wan et al., 2024), task-specific adapters (Liu et al., 2024), or parameter merging (Yadav et al., 2026). Beyond imitation learning, recent continual VLA methods carry policy parameters across tasks using on-policy RL (Hu et al., 2026; Zeng et al., 2026) or demonstration-derived reinforcement fine-tuning (Liu et al., 2026b). 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 (Li et al., 2026b). 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.

4.3 Preliminaries

4.3.1 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 (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024; NVIDIA, 2025b) or WAM (Ye et al., 2026; Dyna Robotics, 2026; Zhu et al., 2025; Pai et al., 2025). 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 (Zhao et al., 2023; Chi et al., 2025). 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}.

4.3.2 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 (Van de Ven et al., 2022), 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 (Hu et al., 2026; Liu et al., 2026b) 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:

$$\max_{\pi^k} \; J^k(\pi^k) = \frac{1}{k}\sum_{i=1}^{k} \mathbb{E}_{\tau\sim(\pi^k,T^i)}\left[\sum_{t=0}^{L_\tau} \gamma^t r_t^i\right], \tag{4.1}$$

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.

4.3.3 Action-Chunk Reinforcement Learning

Within stage k, we improve performance on the current task T^k using an off-policy actor–critic algorithm (Seo et al., 2025b,a) 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 (Li et al., 2025a; Kim et al., 2026a; Xu et al., 2026; Dong

et al., 2026). 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

$$Q_\psi(s_t, \mathbf{a}_t) \approx R_t^{(H)} + \gamma^H \, \mathbb{E}_{\mathbf{a}' \sim \pi_\theta(\cdot|s_{t+H})}[Q_\psi(s_{t+H}, \mathbf{a}')] \,. \tag{4.2}$$

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

4.4 Method

We introduce ACCRUE, a simple but effective continual off-policy RL framework for RFMs (Fig. 4.2). 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. 4.4.1). 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. 4.4.2). 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. 4.4.3).

4.4.1 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 4.4). Unlike RL Token (Xu et al., 2026), 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 (Xu et al., 2026) during training, replacing $\tilde{\mathbf{a}}$ with zeros for a random fraction p of each batch for both networks.

4.4.2 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,

$$\mathcal{L}^{\text{cur}}_{\pi}(\theta; \mathcal{B}^k) = \mathbb{E}_{\substack{(s,\tilde{\mathbf{a}})\sim\mathcal{B}^k \\ \mathbf{a}\sim\pi_\theta(s,\tilde{\mathbf{a}})}} \big[- Q_\psi\big(s, \mathbf{a}, \tilde{\mathbf{a}}\big) + \beta \|\mathbf{a} - \tilde{\mathbf{a}}\|_2^2 \big], \tag{4.3}$$

where β penalizes deviation from the reference action (Xu et al., 2026), 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 $(s_t, \mathbf{a}_t, \tilde{\mathbf{a}}_t, R^{(H)}_t, s_{t+H}, \tilde{\mathbf{a}}_{t+H})$ sampled from $\mathcal{B}^k$, we write the common chunk-level target for value bootstrapping, omitting learner-specific terms:

$$\hat{Q} = R^{(H)}_t + \gamma^H Q_{\psi'}(s_{t+H}, \mathbf{a}_{t+H}, \tilde{\mathbf{a}}_{t+H}), \tag{4.4}$$

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 (Bellemare et al., 2017), which we found more stable under sparse reward.

4.4.3 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

$$\hat{\mathbf{a}}^{k-1}_j = \bar{\pi}_{\theta^{k-1}}(s_j, \tilde{\mathbf{a}}_j), \qquad j \in \mathcal{B}^{<k}. \tag{4.5}$$

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 4.1 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 $j \in \mathcal{B}^{<k}$ ▷ (4.5)

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 $\mathcal{B}^k$ ▷ (4.4)

  Update θ on $\mathcal{B}^k \cup \mathcal{B}^{<k}$ ▷ (4.7)

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,

$$\mathcal{L}_\pi^{\text{prior}}(\theta; \mathcal{B}^{<k}) = \mathop{\mathbb{E}}_{j \sim \mathcal{B}^{<k}} \left[\left\| \pi_\theta(s_j, \tilde{\mathbf{a}}_j) - \hat{\mathbf{a}}_j^{k-1} \right\|_2^2 \right], \tag{4.6}$$

$$\mathcal{L}_\pi(\theta; k) = \mathcal{L}_\pi^{\text{cur}}(\theta; \mathcal{B}^k) + \lambda \, \mathcal{L}_\pi^{\text{prior}}(\theta; \mathcal{B}^{<k}), \tag{4.7}$$

where λ weights prior-task distillation. Thus, we do not use prior-task replay for critic updates or for the actor objective in Eq. equation 4.3. 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 4.1 summarizes one stage of ACCRUE.

4.5 Experiments

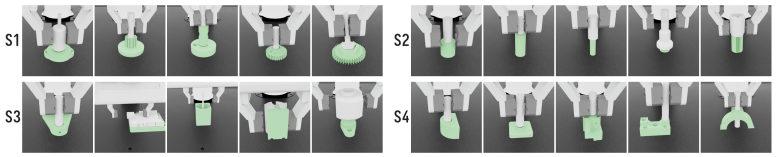
Our experiments ask three questions. 1) Does ACCRUE retain earlier assembly skills once the base policy has been updated for a later stage? 2) Does inheriting the lightweight actor and critic accelerate a new stage? And 3) Which of ACCRUE's components is responsible for each effect? Sec. 4.5.1 describes what is held common across platforms. Sec. 4.5.2 and Sec. 4.5.3 address the first two questions in simulation and on real-world DROID (Khazatsky et al., 2024). 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. 4.5.4 addresses the third question in simulation, where the stage budget and the task sequence can be held fixed across variants.

4.5.1 Experimental Setup

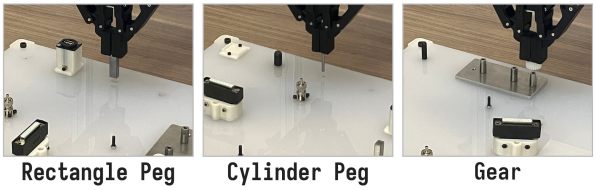
Baselines We compare continual learners within a shared off-policy actor–critic architecture, alongside reference methods with different learning or storage requirements. In the main simulation com-



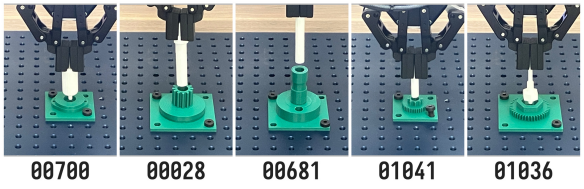
(a) FORGE (Noseworthy et al., 2025)



(b) AutoMate (Tang et al., 2024)



(c) Real-world NIST Assembly Task Board



(d) Real-world AutoMate

Figure 4.3: Overview of experimental tasks. (a) Three contact-rich manipulation tasks in FORGE (Noseworthy et al., 2025). 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.

parisons, 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. 4.3.2 and the off-policy RL algorithm, and inherit the actor–critic pair across tasks. **Sequential** trains it using only current-task data. **ER** (Chaudhry et al., 2019) 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** (Kirkpatrick et al., 2017) penalizes actor drift from the previous-stage parameters, while **RETAIN** (Yadav et al., 2026) merges the current and previous actor parameters through discounted interpolation. As an external reference, **Simple Recipe** (Hu et al., 2026) 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 (Liu et al., 2023; Hu et al., 2026; Liu et al., 2026a), 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^*.

$$\begin{aligned}
\mathrm{FWT}_k &= \tfrac{1}{|\mathcal{E}|} \textstyle\sum_{e\in\mathcal{E}} c_{k,k,e}, \\
\mathrm{NBT}_k &= \tfrac{1}{K-k} \textstyle\sum_{m=k+1}^{K} \big(c_{k,k} - c_{m,k}\big), \\
\mathrm{AUC}_k &= \tfrac{1}{K-k+1} \big(\mathrm{FWT}_k + \textstyle\sum_{m=k+1}^{K} c_{m,k}\big).
\end{aligned}$$

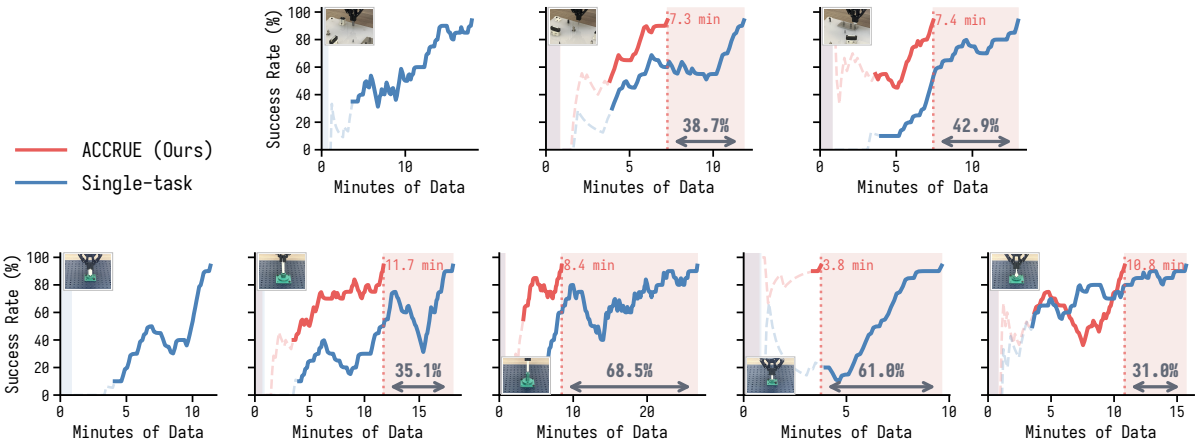


Figure 4.4: Real-world continual RL on DROID setup. (Top) Three-stage NIST Assembly Task Board sequence: **Rectangle Peg** → **Cylinder Peg** → **Gear**. (Bottom) Five-stage AutoMate sequence: **00700** → **00028** → **00681** → **01041** → **01036**. Each panel reports the running success rate over 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.

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. 4.7 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. 4.3 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.

4.5.2 Simulation Experiments

Benchmarks We use AutoMate (Tang et al., 2024) and FORGE (Noseworthy et al., 2025), both built on Isaac Lab (Mittal et al., 2025). 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 (Liu et al., 2023), 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. Figures 4.3a

Table 4.1: 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 ↑	$\text{ACC}_{\text{final}}$ ↑	F_{final} ↓
Shared actor–critic methods					
Sequential	0.427	0.819	<u>0.718</u>	<u>0.685</u>	0.409
EWC (Kirkpatrick et al., 2017)	<u>0.217</u>	0.593	0.582	0.603	<u>0.224</u>
RETAIN (Yadav et al., 2026)	0.514	0.802	0.682	0.644	0.474
ER (Chaudhry et al., 2019)	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 (Hu et al., 2026)	0.038	0.586	0.583	0.594	0.061
Isolated†	–	0.816	0.868	0.919	–

Table 4.2: 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 ↑	$\text{ACC}_{\text{final}}$ ↑	F_{final} ↓
Shared actor–critic methods					
Sequential	0.178	0.824	0.792	0.765	0.175
EWC (Kirkpatrick et al., 2017)	<u>0.075</u>	0.704	0.704	0.699	<u>0.067</u>
RETAIN (Yadav et al., 2026)	0.191	0.822	0.784	0.750	0.192
ER (Chaudhry et al., 2019)	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 (Hu et al., 2026)	0.016	0.730	0.726	0.730	0.019
Isolated†	–	0.826	0.858	0.879	–

and 4.3b 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 chunk carries a learning signal, and we use a discount factor of $\gamma = 0.997$. We use GR00T N1.7 (NVIDIA, 2025b) 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 4.5.4. For RL, we adopt FastSAC (Seo et al., 2025b) over chunks of length $H = 16$ on frozen DINOv3 features (Siméoni et al., 2026), 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 4.1 and 4.2). 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 4.1 and 4.2). These results indicate the efficacy of ACCRUE in setup with limited interaction budget.

4.5.3 DROID Experiments

Tasks To validate ACCRUE beyond simulation, We assess it on the DROID platform (Khazatsky et al., 2024) 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. Figures 4.3c and 4.3d visualizes the tasks.

Setup For the base policy π_{base}, we utilize $\pi_{0.5}$ (Black et al., 2025) 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 robot's seven joint positions, without additional object- or task-state information. For RL backbone, we employ FastTD3 (Seo et al., 2025b) 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 Dong et al. (2026), 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. 4.4), and retention remains essentially complete, with every earlier task succeeding on at least 18 of 20 trials and 19.4 on average (Table 4.3). 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.

Table 4.3: 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

4.5.4 Ablation Studies and Analyses

Table 4.4: 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

Actor–critic inputs Replacing the fixed DINOv3 representation with features extracted by the stage-updated GR00T VLM yields worse retention and forward transfer (Table 4.4). 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.

Reference guidance and critic inheritance Removing the reference-deviation penalty by setting $\beta = 0$ in Eq. 4.3 has little effect on backward retention but substantially weakens forward transfer and task learning (Table 4.4). 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 4.5). Memories of 10–20 episodes are insufficient to preserve prior behavior, while increasing the budget from 100 to 200 episodes

Table 4.5: 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>

degrades performance. This non-monotonic trend shows that effective continual post-training does not follow directly from replaying more prior data.

Table 4.6: 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

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 4.6). 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.

Table 4.7: 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 (NVIDIA, 2025b) (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 (Yuan et al., 2026) (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**

Base-policy dependence In Table 4.7, we apply ACCRUE on Fast-WAM (Yuan et al., 2026) 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.

4.6 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 Robot Foundation Models.

Chapter 5. Concluding Remark

In this thesis, we introduced efficient post-training methods that improve Robot Foundation Models beyond their demonstrations in real-world deployment, where online interaction, policy evaluation, and retraining for new tasks all consume robot time and human effort. Specifically, in Chapter 2, we proposed DEAS, an offline RL framework that learns values over action sequences through detached value learning, allowing RFMs to improve from fixed datasets without additional interaction and doubling the mean success of a fine-tuned GR00T N1.5 policy on RoboCasa Kitchen tasks. In Chapter 3, we introduced RACE, a rollout-free checkpoint selection metric that calibrates action deviations against thresholds derived from successful reference trajectories and accumulates them through a running product, yielding rankings that align with deployment success in simulation and on real robots. In Chapter 4, we presented ACCRUE, a continual off-policy RL framework that carries lightweight actor and critic networks across tasks and preserves earlier behavior through actor distillation, reducing real-robot training time on subsequent tasks by 31.0–68.5% with near-zero forgetting. Together, these methods show that reusing collected data, reference trajectories, and previously learned knowledge can substantially reduce the physical cost of adapting RFMs to their target scenes.

Future work. Our long-term goal is efficient and effective post-training of RFMs in the real world: given a target task, a robot improves its policy through interaction, identifies the best resulting checkpoint, and deploys it, with minimal human effort throughout. Toward this goal, we envision scaling real-world post-training along three axes: tasks, efficiency, and embodiment. First, post-training should extend to broader and more diverse tasks. Given a set of tasks, rather than learning them all at once or in an arbitrary sequence, one can consider selecting the order in which they are learned so that each task benefits most from earlier ones, aiming to achieve higher success with less training than sequential or even multi-task fine-tuning, even when the cost of selecting the order is included in the training budget. Next, post-training should require minimal real-world data and compute. A promising setting is real-world refinement after sim-to-real training: dexterous policies trained in simulation acquire coordinated skills such as grasping and approaching, yet still fail during contact-rich phases in the real world because of discrepancies such as fixtures that comply under contact (Lum et al., 2026). Treating such a policy as a skill prior, off-policy RL could reuse real-world experience to explore useful contact corrections, such as insertion pose alignment and coordinated finger motion, while preserving the pretrained coordination. Lastly, post-training should scale to complex, high-degree-of-freedom embodiments such as humanoids. A humanoid could continually improve its manipulation across progressively harder tasks, from simple pick-and-place to contact-rich manipulation while reusing balance and skill primitives acquired in earlier stages.

Bibliography

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical AI. *arXiv preprint arXiv:2501.03575*, 2025.

Pranav Atreya, Karl Pertsch, Tony Lee, Moo Jin Kim, Arhan Jain, Artur Kuramshin, Cyrus Neary, Edward S. Hu, Kanav Arora, Kirsty Ellis, Luca Macesanu, Matthew Leonard, Meedeum Cho, Ozgur Aslan, Shivin Dass, Tony Wang, Xingfang Yuan, Abhishek Gupta, Dinesh Jayaraman, Glen Berseth, Kostas Daniilidis, Roberto Martín-Martín, Youngwoon Lee, Percy Liang, Chelsea Finn, and Sergey Levine. Roboarena: Distributed real-world evaluation of generalist robot policies. In *Conference on Robot Learning*, 2025.

Apurva Badithela, David Snyder, Lihan Zha, Joseph Mikhail, Matthew O'Kelly, Anushri Dixit, and Anirudha Majumdar. Reliable and scalable robot policy evaluation with imperfect simulators. *arXiv preprint arXiv:2510.04354*, 2025.

Stefan Bauer, Manuel Wüthrich, Felix Widmaier, Annika Buchholz, Sebastian Stark, Anirudh Goyal, Thomas Steinbrenner, Joel Akpo, Shruti Joshi, Vincent Berenz, et al. Real robot challenge: A robotics competition in the cloud. In *NeurIPS 2021 Competitions and Demonstrations Track*, 2022.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, 2017.

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

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Conference on Robot Learning*, 2025.

Steven Bradtke and Michael Duff. Reinforcement learning methods for continuous-time markov decision problems. In *Conference on Neural Information Processing Systems*, 1994.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, 2023.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. In *arXiv preprint arXiv:1902.10486*, 2019.

Yevgen Chebotar, Quan Vuong, Karol Hausman, Fei Xia, Yao Lu, Alex Irpan, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, et al. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *Conference on Robot Learning*, 2023.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *International Conference on Learning Representations*, 2023.

Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. Conrft: A reinforced fine-tuning method for vla models via consistency policy. *arXiv preprint arXiv:2502.05450*, 2025.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *International Journal of Robotics Research*, 2025.

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

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

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

Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taiga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, Aviral Kumar, and Rishabh Agarwal. Stop regressing: Training value functions via classification for scalable deep RL. In *International Conference on Machine Learning*, 2024.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Conference on Neural Information Processing Systems*, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, 2019.

Divyansh Garg, Joey Hejna, Matthieu Geist, and Stefano Ermon. Extreme q-learning: Maxent RL without entropy. In *International Conference on Learning Representations*, 2023.

Gemini Robotics Team, Abbas Abdolmaleki, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Ashwin Balakrishna, Nathan Batchelor, Alex Bewley, Jeff Bingham, et al. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025a.

Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025b.

Generalist AI Team. Gen-1: Scaling embodied foundation models to mastery. *Generalist AI Blog*, 2026. https://generalistai.com/blog/apr-02-2026-GEN-1.

Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Thomas Paine, Sergio Gómez, Konrad Zolna, Rishabh Agarwal, Josh S Merel, Daniel J Mankowitz, Cosmin Paduraru, et al. Rl unplugged: A suite of

benchmarks for offline reinforcement learning. *Conference on Neural Information Processing Systems*, 2020.

Yanjiang Guo, Jianke Zhang, Xiaoyu Chen, Xiang Ji, Yen-Jen Wang, Yucheng Hu, and Jianyu Chen. Improving vision-language-action model with online reinforcement learning. In *IEEE International Conference on Robotics and Automation*, 2025.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

Jiaheng Hu, Jay Shim, Chen Tang, Yoonchang Sung, Bo Liu, Peter Stone, and Roberto Martín-Martín. Simple recipe works: Vision-language-action models are natural continual learners with reinforcement learning. *arXiv preprint arXiv:2603.11653*, 2026.

Leonard Hussenot, Marcin Andrychowicz, Damien Vincent, Robert Dadashi, Anton Raichuk, Lukasz Stafiniak, Sertan Girgin, Raphael Marinier, Nikola Momchev, Sabela Ramos, Manu Orsini, Olivier Bachem, Matthieu Geist, and Olivier Pietquin. Hyperparameter selection for imitation learning. In *International Conference on Machine Learning*, 2021.

Arhan Jain, Mingtong Zhang, Kanav Arora, William Chen, Marcel Torne, Muhammad Zubair Irshad, Sergey Zakharov, Yue Wang, Sergey Levine, Chelsea Finn, Wei-Chiu Ma, Dhruv Shah, Abhishek Gupta, and Karl Pertsch. Polaris: Scalable real-to-sim evaluations for generalist robot policies. *arXiv preprint arXiv:2512.16881*, 2025.

Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *International Conference on Machine Learning*, 2016.

Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Jim Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *IEEE International Conference on Robotics and Automation*, 2025.

Michael J Kearns and Satinder Singh. Bias-variance error bounds for temporal difference updates. In *Conference on Learning Theory*, 2000.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems*, 2024.

Changyeon Kim, Haeone Lee, Younggyo Seo, Kimin Lee, and Yuke Zhu. DEAS: DEtached value learning with action sequence for scalable offline RL. In *International Conference on Learning Representations*, 2026a.

Changyeon Kim, Kimin Lee, Yashraj Narang, and Yijie Guo. ACCRUE: Continual off-policy reinforcement learning for robot foundation models. In *Continually Self-Improving Robots Workshop at Conference on Robot Learning*, 2026b.

Changyeon Kim, Yekyung Nah, and Kimin Lee. RACE: Calibrated cumulative scoring for rollout-free checkpoint selection. In *Robot Learning Workshop at Conference on Neural Information Processing Systems*, 2026c.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning*, 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*, 2026d.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 2017.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Conference on Neural Information Processing Systems*, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Conference on Neural Information Processing Systems*, 2020.

Aviral Kumar, Rishabh Agarwal, Xinyang Geng, George Tucker, and Sergey Levine. Offline q-learning on diverse multi-task data both scales and generalizes. In *International Conference on Learning Representations*, 2023a.

Aviral Kumar, Anikait Singh, Frederik Ebert, Mitsuhiko Nakamoto, Yanlai Yang, Chelsea Finn, and Sergey Levine. Pre-training for robots: Offline rl enables learning new tasks from a handful of trials. In *Robotics: Science and Systems*, 2023b.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning: State-of-the-art*, 2012.

Hoang M. Le, Cameron Voloshin, and Yisong Yue. Batch policy learning under constraints. *arXiv preprint arXiv:1903.08738*, 2019.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Ge Li, Dong Tian, Hongyi Zhou, Xinkai Jiang, Rudolf Lioutikov, and Gerhard Neumann. Top-erl: Transformer-based off-policy episodic reinforcement learning. *arXiv preprint arXiv:2410.09536*, 2024a.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Yang Zhaohui, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, et al. Simplevla-rl: Scaling vla training via reinforcement learning. In *International Conference on Learning Representations*, 2026a.

Houlin Li, Minghui Xu, Guo Xu, Xuan Du, Xiaohan Yan, Chun Wang, Yuxiang Yan, Shukai Yang, Yongcheng Liu, Wei Shan, and Maoqing Yao. CIDER: Continual interactive distillation for embodied reinforcement learning. *arXiv preprint arXiv:2608.21899*, 2026b.

Qiyang Li, Zhiyuan Paul Zhou, and Sergey Levine. Reinforcement learning with action chunking. In *Conference on Neural Information Processing Systems*, 2025a.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Oier Mees, Karl Pertsch, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. In *8th Annual Conference on Robot Learning*, 2024b.

Yaxuan Li, Yichen Zhu, Junjie Wen, Chaomin Shen, and Yi Xu. Worldeval: World model as real-world robot policies evaluator. *arXiv preprint arXiv:2505.19017*, 2025b.

Yaxuan Li, Zhongyi Zhou, Yefei Chen, Yaokai Xue, and Yichen Zhu. dworldeval: Scalable robotic policy evaluation via discrete diffusion world model. In *International Conference on Machine Learning*, 2026c.

Yunfei Li, Xiao Ma, Jiafeng Xu, Yu Cui, Zhongren Cui, Zhigang Han, Liqun Huang, Tao Kong, Yuxiao Liu, Hao Niu, et al. Gr-rl: Going dexterous and precise for long-horizon robotic manipulation. *arXiv preprint arXiv:2512.01801*, 2025c.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In *Conference on Neural Information Processing Systems*, 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*, 2026a.

Yuan Liu, Haoran Li, Shuai Tian, Yuxing Qin, Yuhui Chen, Yupeng Zheng, Yongzhen Huang, and Dongbin Zhao. Towards long-lived robots: Continual learning vla models via reinforcement fine-tuning. In *Robotics: Science and Systems*, 2026b.

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.

Tyler Ga Wei Lum, Kushal Kedia, C. Karen Liu, and Jeannette Bohg. Play2Perfect: What matters in dexterous play pretraining for precise assembly? In *Conference on Robot Learning*, 2026.

Rachel Luo, Heng Yang, Michael Watson, Apoorva Sharma, Sushant Veer, Edward Schmerling, and Marco Pavone. Leveraging correlation across test platforms for variance-reduced metric estimation. In *Conference on Robot Learning*, 2025.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning*, 2021.

Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Munoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. In *Conference on Robot Learning*, 2024.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems*, 2024.

Michael Noseworthy, Bingjie Tang, Bowen Wen, Ankur Handa, Chad Kessens, Nicholas Roy, Dieter Fox, Fabio Ramos, Yashraj Narang, and Iretiayo Akinola. Forge: Force-guided exploration for robust contact-rich manipulation under uncertainty. In *IEEE Robotics and Automation Letters*, 2025.

NVIDIA. Gr00t n1.5: An improved open foundation model for generalist humanoid robots. `https://research.nvidia.com/labs/gear/gr00t-n1_5/`, June 2025a. Accessed: 2025-09-09.

NVIDIA. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025b.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International Conference on Machine Learning*, 2018.

Ajay Pai et al. mimic-video: Video-action models for generalizable robot control beyond VLAs. *arXiv preprint arXiv:2512.15692*, 2025.

Tom Le Paine, Cosmin Paduraru, Andrea Michi, Caglar Gulcehre, Konrad Zolna, Alexander Novikov, Ziyu Wang, and Nando de Freitas. Hyperparameter selection for offline reinforcement learning. *arXiv preprint arXiv:2007.09055*, 2020.

Chaoyi Pan, Giri Anantharaman, Nai-Chieh Huang, Claire Jin, Daniel Pfrommer, Chenyang Yuan, Frank Permenter, Guannan Qu, Nicholas Boffi, Guanya Shi, and Max Simchowitz. Much ado about noising: Dispelling the myths of generative robotic control. In *International Conference on Learning Representations*, 2026.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is value learning really the main bottleneck in offline rl? In *Conference on Neural Information Processing Systems*, 2024.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations*, 2025a.

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes rl scalable. In *Conference on Neural Information Processing Systems*, 2025b.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *International Conference on Machine Learning*, 2025c.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Conference on Neural Information Processing Systems*, 1988.

Doina Precup, Richard S. Sutton, and Satinder P. Singh. Eligibility traces for off-policy policy evaluation. In *International Conference on Machine Learning*, 2000.

Julian Quevedo, Ansh Kumar Sharma, Yixiang Sun, Varad Suryavanshi, Percy Liang, and Sherry Yang. Evaluating robot policies in a world model. *arXiv preprint arXiv:2506.00613*, 2025.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, 2011.

Younggyo Seo and Pieter Abbeel. Coarse-to-fine q-network with action sequence for data-efficient robot learning. In *Conference on Neural Information Processing Systems*, 2025.

Younggyo Seo, Carmelo Sferrazza, Juyue Chen, Guanya Shi, Rocky Duan, and Pieter Abbeel. Learning sim-to-real humanoid locomotion in 15 minutes. *arXiv preprint arXiv:2512.01996*, 2025a.

Younggyo Seo, Carmelo Sferrazza, Haoran Geng, Michal Nauman, Zhao-Heng Yin, and Pieter Abbeel. Fasttd3: Simple, fast, and capable reinforcement learning for humanoid control. *arXiv preprint arXiv:2505.22642*, 2025b.

Max Simchowitz, Daniel Pfrommer, and Ali Jadbabaie. The pitfalls of imitation learning when actions are continuous. In *Conference on Learning Theory*, 2025.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seung Eun Yi, Michael Ramamonjisoa, Francisco Massa, Daniel HAZIZA, Luca Wehrstedt, Jianyuan Wang, et al. DINOv3. *Transactions on Machine Learning Research*, 2026.

David Snyder, Asher James Hancock, Apurva Badithela, Emma Dixon, Patrick Miller, Rares Andrei Ambrus, Anirudha Majumdar, Masha Itkina, and Haruki Nishimura. Is your imitation learning policy better than mine? policy comparison with near-optimal stopping. *arXiv preprint arXiv:2503.10966*, 2025.

David Snyder, Apurva Badithela, Nikolai Matni, George Pappas, Anirudha Majumdar, Masha Itkina, and Haruki Nishimura. Beyond binary success: Sample-efficient and statistically rigorous robot policy comparison. *arXiv preprint arXiv:2603.13616*, 2026.

Jonathan Spencer, Sanjiban Choudhury, Arun Venkatraman, Brian Ziebart, and J. Andrew Bagnell. Feedback in imitation learning: The three regimes of covariate shift. *arXiv preprint arXiv:2102.02872*, 2021.

Jost Tobias Springenberg, Abbas Abdolmaleki, Jingwei Zhang, Oliver Groth, Michael Bloesch, Thomas Lampe, Philemon Brakel, Sarah Maria Elisabeth Bechtle, Steven Kapturowski, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Offline actor-critic reinforcement learning scales to large models. In *International Conference on Machine Learning*, 2024.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 1999.

Bingjie Tang, Iretiayo Akinola, Jie Xu, Bowen Wen, Ankur Handa, Karl Van Wyk, Dieter Fox, Gaurav S Sukhatme, Fabio Ramos, and Yashraj Narang. Automate: Specialist and generalist assembly policies over diverse geometries. In *Robotics: Science and Systems*, 2024.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. In *Conference on Neural Information Processing Systems*, 2023.

Dong Tian, Ge Li, Hongyi Zhou, Onur Celik, and Gerhard Neumann. Chunking the critic: A transformer-based soft actor-critic with n-step returns. *arXiv preprint arXiv:2503.03660*, 2025.

John Tsitsiklis and Benjamin Van Roy. Analysis of temporal-diffference learning with function approximation. In *Conference on Neural Information Processing Systems*, 1996.

Gido M Van de Ven, Tinne Tuytelaars, and Andreas S Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 2022.

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

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

Yixuan Wang, Rhythm Syed, Fangyu Wu, Mengchao Zhang, Aykut Onol, Jose Barreiros, Hooshang Nayyeri, Tony Dear, Huan Zhang, and Yunzhu Li. Interactive world simulator for robot policy training and evaluation. *arXiv preprint arXiv:2603.08546*, 2026.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *International Conference on Learning Representations*, 2023.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. In *Conference on Neural Information Processing Systems*, 2020.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Zhengyi Luo, Yuqi Xie, Fengyuan Hu, Jim Fan, Guanya Shi, et al. Self-improving vision-language-action models with data generation via residual rl. In *International Conference on Learning Representations*, 2026.

Charles Xu, Jost Tobias Springenberg, Michael Equi, Ali Amin, Adnan Esmail, Sergey Levine, and Liyiming Ke. Rl token: Bootstrapping online rl with vision-language-action models. *arXiv preprint arXiv:2604.23073*, 2026.

Haoran Xu, Li Jiang, Jianxiong Li, Zhuoran Yang, Zhaoran Wang, Victor Wai Kin Chan, and Xianyuan Zhan. Offline RL with no OOD actions: In-sample learning via implicit value regularization. In *International Conference on Learning Representations*, 2023.

Yajat Yadav, Zhiyuan Zhou, Andrew Wagenmaker, Karl Pertsch, and Sergey Levine. Robust fine-tuning of vision-language-action robot policies via parameter merging. In *International Conference on Learning Representations*, 2026.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, 2020.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Zhang Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. In *International Conference on Learning Representations*, 2025.

Qixin Zeng, Shuo Zhang, Hongyin Zhang, Renjie Wang, Han Zhao, Libang Zhao, Runze Li, Donglin Wang, and Chao Huang. Crl-vla: Continual vision-language-action learning. *arXiv preprint arXiv:2602.03445*, 2026.

Kaifeng Zhang, Shuo Sha, Hanxiao Jiang, Matthew Loper, Hyunjong Song, Guangyan Cai, Zhuo Xu, Xiaochen Hu, Changxi Zheng, and Yunzhu Li. Real-to-sim robot policy evaluation with gaussian splatting simulation of soft-body interactions. In *IEEE International Conference on Robotics and Automation*, 2026.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems*, 2023.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Zhiyuan Zhou, Pranav Atreya, You Liang Tan, Karl Pertsch, and Sergey Levine. Autoeval: Autonomous evaluation of generalist robot manipulation policies in the real world. In *Conference on Robot Learning*, 2025.

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