

Actor-Critic for Continuous Action Chunks: A Reinforcement Learning Framework for Long-Horizon Robotic Manipulation with Sparse Reward

Jiarui Yang¹, Bin Zhu², Jingjing Chen^{3*}, Yu-Gang Jiang³

¹College of Computer Science and Artificial Intelligence, Fudan University

²Singapore Management University

³Institute of Trustworthy Embodied AI, Fudan University

jryang24@m.fudan.edu.cn, binzhu@smu.edu.sg, {chenjingjing,ygj}@fudan.edu.cn

Abstract

Existing reinforcement learning (RL) methods struggle with long-horizon robotic manipulation tasks, particularly those involving sparse rewards. While action chunking is a promising paradigm for robotic manipulation, using RL to directly learn continuous action chunks in a stable and data-efficient manner remains a critical challenge. This paper introduces AC3 (Actor-Critic for Continuous Chunks), a novel RL framework that learns to generate high-dimensional, continuous action sequences. To make this learning process stable and data-efficient, AC3 incorporates targeted stabilization mechanisms for both the actor and the critic. First, to ensure reliable policy improvement, the actor is trained with an asymmetric update rule, learning exclusively from successful trajectories. Second, to enable effective value learning despite sparse rewards, the critic’s update is stabilized using intra-chunk n -step returns and further enriched by a self-supervised module providing intrinsic rewards at anchor points aligned with each action chunk. We conducted extensive experiments on 25 tasks from the BiGym and RL Bench benchmarks. Results show that by using only a few demonstrations and a simple model architecture, AC3 achieves superior success rates on most tasks, validating its effective design.

Code — <https://github.com/flyfaerss/ac3>

Introduction

Recent reinforcement learning (RL) algorithms (Haarnoja et al. 2018; Kalashnikov et al. 2018, 2021; Yarats et al. 2021; Herzog et al. 2023) have demonstrated significant advances in learning continuous-action control from online experiences. Nevertheless, these methods excel mostly at short-horizon tasks such as *closing drawers* or *doors*, and consistently struggle to learn effective policies for long-horizon problems, i.e., those comprising multiple sub-tasks and requiring precise manipulation, such as *moving plates* or *sandwich flipping*.

To complete long-horizon tasks, the agent must execute an extended sequence of coherent actions, vastly expanding the state-action exploration space. In most cases, the failure of any single sub-task directly leads to failure in exploring the overall task. As a result, this learning challenge is further

*Corresponding author.

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

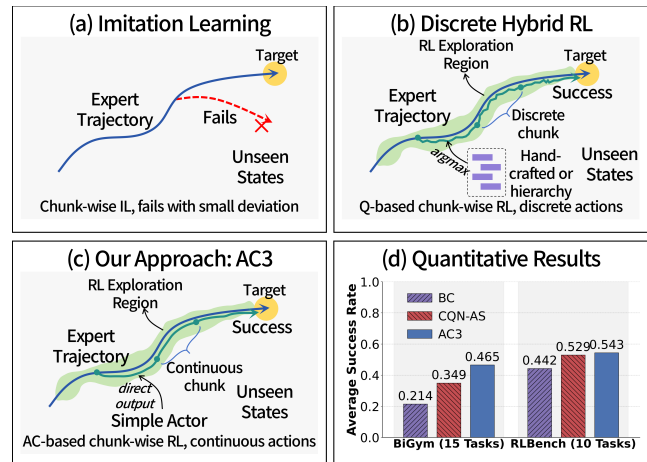


Figure 1: (a) Imitation Learning is not robust to unseen states. (b) Hybrid RL with discrete chunks lacks precision. (c) Our approach, AC3, directly learns continuous chunks for more effective control. (d) AC3 achieves superior performance.

compounded by sparse rewards: the agent can only obtain positive feedback upon task completion, leaving exploration in intermediate steps without effective guiding signals. Consequently, it becomes exceedingly difficult for the agent to determine which preceding actions are crucial for eventual success, leading to inefficient autonomous exploration.

For long-horizon tasks, predicting entire sequences of actions—a paradigm known as action chunking—has proven highly successful, finding its most natural application in Imitation Learning (IL). This success is exemplified by recent methods like ACT (Zhao et al. 2023) and Diffusion Policy (Chi et al. 2023), which use powerful generative models to effectively clone complex, continuous action sequences from expert data. Despite their power, these IL methods are constrained by an upper bound, as their performance is limited by the provided expert demonstrations. Due to distributional shift (Ross, Gordon, and Bagnell 2011), the agent is likely to fail when it encounters unseen states that deviate even slightly from the expert trajectory (Fig. 1(a)). The promise of overcoming these limitations, through online interaction and the potential to discover superior policies, provides a

motivation to integrate the principles of RL.

However, bringing the chunking paradigm into RL is non-trivial. The complexity of the exploration space increases exponentially as the chunk length grows, making exploration and value estimation intractable for standard methods and frequently leading to exploding Q-value estimates (Seo and Abbeel 2025) and subsequent training instability. Recent work CQN-AS (Seo and Abbeel 2025) addressed this by discretizing a library of chunks and learning a Q-function over that set, but the discretization inevitably sacrifices precision and flexibility (as shown in Fig. 1(b)). Although other concurrent work (Li, Zhou, and Levine 2025) has attempted to generate continuous action chunks via complex distillation pipelines, these methods often rely on large-scale offline datasets and are computationally expensive, limiting their application in few-shot, real-time control settings. Consequently, a critical gap remains in the field: an efficient RL framework that can directly and data-efficiently leverage the full potential of action chunking in continuous domains.

To bridge this critical gap, this paper introduces AC3 (Actor-Critic for Continuous Chunks), a novel framework designed to efficiently learn continuous action chunks by leveraging a small number of expert demonstrations. AC3 builds directly upon a DDPG-style framework (Lillicrap et al. 2015) to predict continuous action chunks, thereby enabling more flexible and precise robotic control. However, ensuring the stability of such a framework is paramount, especially when learning from limited data under sparse rewards.

To this end, AC3 incorporates two key innovations for stabilization. First, to ensure reliable policy improvement, the actor is trained with an asymmetric update rule: its policy network learns exclusively from successful trajectories (including expert demonstrations and successful online rollouts). This approach strictly confines the policy optimization to the “trusted region” where the value function is most reliable, thus avoiding misleading gradients from inaccurate Q-values and guaranteeing stable learning. Second, to stabilize the critic’s learning under sparse rewards, its update utilizes intra-chunk n-step returns and is further enhanced by a self-supervised module. This module employs a goal network, pre-trained on expert demonstrations, to provide intrinsic reward signals at anchor points aligned with action chunks, thereby effectively guiding the value function’s learning. Together, these components enable AC3 to robustly learn complex, continuous control policies in a data-efficient manner.

To validate the efficacy of AC3, we conducted extensive experiments on 25 robotic tasks from the BiGym (Chernyadev et al. 2024) and RL Bench (James et al. 2020) benchmarks. The results show that on long-horizon, sparse-reward tasks with high-dimensional states, AC3 achieves superior success rates (Fig. 1(d)) while utilizing a simple model architecture, thus validating its effective and data-efficient design.

The main contributions can be summarized as follows:

- We propose AC3, a novel actor-critic framework for learning continuous action chunks. It is designed to tackle long-horizon, sparse-reward manipulation tasks by efficiently leveraging only a small number of expert demonstrations.
- We introduce two key stabilization mechanisms to ensure

stable learning from sparse rewards: (1) an asymmetric update rule that trains the actor exclusively on successful trajectories for robust policy improvement, and (2) a combined critic update method leveraging intra-chunk n-step returns and a self-supervised module that provides chunk-wise intrinsic rewards.

- We conduct extensive experiments on 25 tasks from the BiGym and RL Bench benchmarks, demonstrating that AC3 achieves superior success rates over existing methods by utilizing a simple model architecture, thus validating the effectiveness and data-efficiency of our approach.

Related Work

Imitation Learning from Demonstrations. Imitation Learning (IL) is a data-driven approach leveraging expert demonstrations to learn policies, bypassing RL exploration challenges. The basic method, Behavioral Cloning (BC), treats it as supervised learning but suffers from distributional shift (Ross, Gordon, and Bagnell 2011). To better address long-horizon tasks, the IL paradigm has evolved towards action chunking, where the policy learns to predict entire sequences of actions. State-of-the-art methods like ACT (Zhao et al. 2023; Lee et al. 2024) and Diffusion Policy (Chi et al. 2023; Ren et al. 2024; Wang et al. 2025), employ powerful generative models (e.g., Transformers (Vaswani et al. 2017) and diffusion models (Ho, Jain, and Abbeel 2020)) to approximate the action distribution, and have been extended to multi-task setups (Bharadhwaj et al. 2024; Liu et al. 2024; Doshi et al. 2024), mobile manipulation (Fu, Zhao, and Finn 2024; Motoda et al. 2025) and humanoid control (Fu et al. 2024). However, their performance is fundamentally capped by the expert data and they cannot improve through online interaction, which motivates integrating the principles of RL.

Reinforcement Learning with Action Chunks. Structured modeling of action sequences is vital for efficient policy learning and execution in RL (Saanum et al. 2023). However, applying the action chunking paradigm to a RL context is challenging due to the expanded action space and difficult exploration in sparse-reward, long-horizon environments; existing work thus largely uses demonstrations for guidance. ResiP (Ankile et al. 2024) applies RL as a residual component to fine-tune action chunk predicted by BC, but its RL process still performs policy updates at the low-level, single-timestep scale. T-SAC (Tian et al. 2025) learns a critic on action chunks by integrating n-step returns while still optimizing a single-timestep actor. CQN-AS (Seo and Abbeel 2025), in contrast, uses the action chunk as the high-level unit for RL exploration. However, its reliance on Q-learning forces a discretization of the action space, fundamentally limiting the policy’s precision in continuous control. While more recent work Q-Chunking (Li, Zhou, and Levine 2025) overcomes this by generating continuous chunks via a complex distillation pipeline, it introduces computational overhead and relies on large-scale offline datasets. Therefore, developing a framework that can learn continuous action chunks in a direct, stable, and computationally efficient manner remains a challenge. To address this challenge, we introduce AC3.

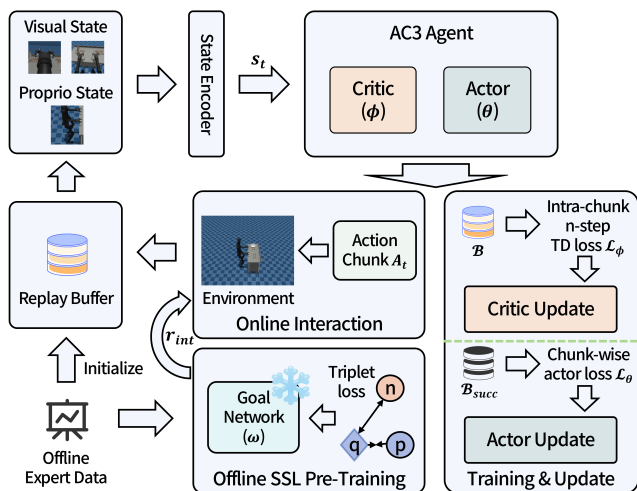


Figure 2: Overall framework of AC3. First, a Goal Network is pre-trained using expert data via self-supervised learning to provide intrinsic rewards r_{int} during subsequent online interactions. Next, during online interaction, the Actor outputs a continuous action chunk and stores new experiences in the Replay Buffer after execution. For training, the Critic is updated via an intra-chunk n-step TD loss, while the Actor learns only from the successful trajectories buffer $\mathcal{B}_{\text{succ}}$ to promote stable policy improvement.

Method

Our goal is to tackle long-horizon robotic manipulation tasks where reward is granted only upon success, using just a few expert demonstrations—a setting that naturally reflects real-world data constraints. To this end, we introduce AC3 (Actor-Critic for Continuous Chunks), an off-policy actor-critic framework designed to learn a policy that generates sequences of continuous actions. As illustrated in Fig. 2, its core design follows three levels: First, we construct a basic AC3 Agent, whose Actor network is designed to directly output continuous action chunks, while the Critic network is responsible for evaluating the value of these chunks. Second, in terms of update rules, we adopt intra-chunk n-step returns to stabilize the Critic’s training, while constraining the Actor’s update to use only transitions from successful trajectories to ensure reliable policy optimization. Third, we introduce a self-supervised reward shaping module, which sets anchor points in units of action chunks to provide relatively dense and reliable intrinsic rewards for the Critic, thereby effectively guiding the learning of the value function.

Preliminaries: Problem Formulation

We model the robotic manipulation task as a semi-Markov Decision Process (SMDP) (Sutton, Precup, and Singh 1999) with continuous chunk space, defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$. Here, \mathcal{S} is a high-dimensional state space, which includes RGB visual states and the robot’s proprioceptive states. \mathcal{A} is a continuous action chunk with fixed length, P is a state transition dynamics, $\gamma \in [0, 1)$ is a discount factor, and r is the reward function. For the tasks we consider, the

reward $r \in \{0, 1\}$ is sparse, providing a positive signal only upon successful task completion. We utilize a small offline dataset of N expert demonstrations $\mathcal{D}_{\text{demo}} = \{\tau_1, \dots, \tau_N\}$, where each trajectory τ is a sequence of state-action pairs. Our objective is to learn a policy $\pi_\theta(s)$ that maximizes the expected discounted return $\mathbb{E}[\sum_{t=0}^T \gamma^t r_t]$.

AC3 Algorithm

AC3 is an off-policy actor-critic algorithm derived from the DDPG-style framework. It is specially designed for long-horizon tasks by making two principal modifications: 1) the actor network outputs temporal action chunks; 2) the critic network is trained using intra-chunk n-step returns. The training pipeline of AC3 is summarized in Algorithm 1.

Actor: Continuous Action Chunking Policy. Temporal abstraction is a critical technique in long-horizon robotic manipulation tasks. It aligns motion planning with the core temporal logic of the entire task, thereby mitigating local sub-optimality that arise from step-wise decision-making. At the same time, this approach also enhances the control over the long-horizon operational sequence, which in turn improves the coherence and precision of action execution. Based on this principle, our policy π_θ , parameterized by θ , directly predicts a continuous action chunk $\mathcal{A}_t \in \mathbb{R}^{C \times d_a}$, which is a sequence of C continuous actions:

$$\mathcal{A}_t = \pi_\theta(s_t) = \{a_t, a_{t+1}, \dots, a_{t+C-1}\}, \quad (1)$$

where s_t is the joint state representation at timestep t , which is formed by concatenating the robot’s proprioceptive state features and current RGB visual features. To capture the temporal dependencies within action sequences while ensuring computational efficiency, the policy network employs a simple MLP+GRU architecture.

Critic: Intra-Chunk n-step Q-value Estimation. The critic, $Q_\phi(s_t, \mathcal{A}_t)$ with parameters ϕ , estimates the expected return after executing the entire chunk \mathcal{A}_t starting from state s_t . In long-horizon tasks, a relatively large chunk size (e.g., 16) is crucial for capturing temporal dependencies and also demonstrates significant performance gains in complex tasks (see Fig. 5). However, in such cases, directly evaluating the Q-value of the entire chunk leads to two key issues: 1) The reward estimation is more noisy, resulting in reduced sample efficiency. As a result, more diverse data are required to converge to an effective policy, yet robot data collection is inherently challenging; 2) The Critic cannot provide fine-grained feedback for each primitive action within the chunk, thereby losing the precision in action execution.

To improve training stability and sample efficiency, we instead use n-step TD target to update the critic. This target is composed of the observed reward sum from the replay buffer and a bootstrapped value estimate from the target networks ($Q_{\phi'}$) at the landing state s_{t+n} . The target value is formulated as:

$$y_t = \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n \min_{i=1,2} Q_{\phi'_i}(s_{t+n}, \pi_\theta(s_{t+n})) + \epsilon, \quad (2)$$

similar to TD3 (Fujimoto, Hoof, and Meger 2018), the exploration noise ϵ is sampled from $\text{clip}(\mathcal{N}(0, \sigma), -c, c)$, and

we keep two target critics and use their clipped double-Q estimate $\min_i Q_{\phi'_i}$ to curb over-estimation bias and stabilize learning. The critic network still employs a simple MLP+GRU architecture similar to actor network.

Training and Updates. The actor and critic are trained by sampling mini-batches of transitions $\tau = (s_t, \mathcal{A}_t^{\text{exp}}, r_{t:t+n-1}, s_{t+n})$ from the replay buffer \mathcal{B} .

Critic Update: The critic’s role is to learn an accurate and robust value function by minimizing the TD error, thus providing global guidance for the policy gradient update direction. Specifically, the parameters ϕ of both critic networks are updated by minimizing the MSE loss between the current Q-value predictions and the target value y_t calculated in Eq. 2. The combined critic loss is:

$$\mathcal{L}_\phi = \sum_{i=1,2} \mathbb{E}_{\tau \sim \mathcal{B}} [(Q_{\phi_i}(s_t, \mathcal{A}_t^{\text{exp}}) - y_t)^2]. \quad (3)$$

Actor Update: Under the conditions of high dimensionality of the state space and sparse rewards, the blind exploration of RL will become inefficient. Therefore, a common practice is to use expert demonstrations for guidance. Specifically, we construct a new replay buffer $\mathcal{B}_{\text{succ}}$, which filters the successful trajectory transitions from \mathcal{B} . After that, we directly use the BC loss for imitation.

$$\mathcal{L}_{\theta_{BC}} = \mathbb{E}_{(s_t, \mathcal{A}_t^{\text{exp}}) \sim \mathcal{B}_{\text{succ}}} [\|\pi_\theta(s_t) - \mathcal{A}_t^{\text{exp}}\|^2]. \quad (4)$$

To ensure the policy actively improves itself by leveraging online experience, we further incorporate the Q-value signal provided by the critic:

$$\mathcal{L}_{\theta_Q} = -\mathbb{E}_{s_t \sim \mathcal{B}_{\text{succ}}} \left[\min_{i=1,2} Q_{\phi_i}(s_t, \pi_\theta(s_t)) \right]. \quad (5)$$

Unlike the critic network, we found that if the actor is trained on all transitions, the model’s performance actually degrades (see Fig. 8). This phenomenon is intuitive: in a high-dimensional state space and under conditions of extremely sparse rewards, the critic cannot assign an accurate Q-value to most of the state space, and a new state might cause the model to update in an unpredictable direction. Therefore, we strictly constrain the calculation of the policy gradient to the “trusted region” defined by the successful trajectories. In this way, the policy’s optimization is confined to the regions where the value function is most reliable, effectively avoiding the policy degradation that could result from exploring uncertain states.

The final actor loss is a weighted combination of the BC loss and the Q loss:

$$\mathcal{L}_\theta = \lambda_{BC} \mathcal{L}_{\theta_{BC}} + \lambda_Q \mathcal{L}_{\theta_Q}, \quad (6)$$

where λ_{BC} and λ_Q are weighting coefficients.

Self-Supervised Reward Shaping

Eq. 6 utilizes a few offline demonstrations to provide direct guidance for RL exploration. However, the environment remains reward-sparse, thus the critic network still struggles to learn a meaningful state-value function. To alleviate this issue, we further construct a self-supervised reward shaping module by pre-training a state goal network, $G_\omega(\cdot)$, on the offline

Algorithm 1: AC3

Initialize critic networks Q_{ϕ_1}, Q_{ϕ_2} , and actor network π_θ with random parameters ϕ_1, ϕ_2, θ

Initialize critic target networks $\phi'_1 \leftarrow \phi_1, \phi'_2 \leftarrow \phi_2$

Initialize replay buffer \mathcal{B} with expert demonstrations $\mathcal{D}_{\text{demo}}$

Pre-train goal network G_ω with parameters ω using Eq. 7

for timestep $t = 1$ to T **do**

if $t \bmod C$ **then**

 Compute action chunk with exploration noise
 $\mathcal{A}_t^{\text{exp}} \sim \pi_\theta(s_t) + \epsilon, \epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c)$

end if

 Pop action a_t from current action chunk, execute a_t in the environment, obtain reward r_t and new state s_{t+1}

$r_t \leftarrow r_{\text{int}}(s_t)$ using Eq. 9 if not terminal s_t

 Store transition tuple (s_t, a_t, r_t, s_{t+1}) in \mathcal{B}

 Sample B transitions $(s_t, \mathcal{A}_t^{\text{exp}}, r_{t:t+n-1}, s_{t+n})$ from replay buffer \mathcal{B} as a mini-batch

 // *Critic Update*

 Compute target value y_t using Eq. 2

$\mathcal{L}_{\phi_i} = \frac{1}{B} \sum [(Q_{\phi_i}(s_t, \mathcal{A}_t^{\text{exp}}) - y_t)^2] \quad \forall i \in \{1, 2\}$

$\phi_i \leftarrow \phi_i - \alpha \nabla_{\phi_i} \mathcal{L}_{\phi_i} \quad \forall i \in \{1, 2\}$

$\phi'_i \leftarrow (1 - \mu) \phi'_i + \mu \phi_i \quad \forall i \in \{1, 2\}$

 // *Actor Update*

 Filter B_{succ} successful trajectories’ transitions from current mini-batch through successful mask m_{succ}

$\mathcal{L}_{\theta_{BC}} = \frac{1}{B_{\text{succ}}} \sum [\|\pi_\theta(s_t) - \mathcal{A}_t^{\text{exp}}\|^2 * m_{\text{succ}}]$

$\mathcal{L}_{\theta_Q} = -\frac{1}{B_{\text{succ}}} \sum [\min_{i=1,2} Q_{\phi_i}(s_t, \pi_\theta(s_t)) * m_{\text{succ}}]$

$\mathcal{L}_\theta = \lambda_{BC} \mathcal{L}_{\theta_{BC}} + \lambda_Q \mathcal{L}_{\theta_Q}$

$\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_\theta$

end for

demonstrations $\mathcal{D}_{\text{demo}}$, with the same architecture as the state encoder. The goal of G_ω is to learn a low-dimensional latent space that captures the key features of successful trajectories.

Specifically, we employ a contrastive learning paradigm to train G_ω . States that are temporally close within the same demonstration trajectory form positive pairs, while states from different trajectories or those temporally distant in the same trajectory form negative pairs. Triplet loss is used to pull positive pairs closer together and push negative pairs further apart in the embedding space.

$$\mathcal{L}_\omega = \mathbb{E}_{(s_q, s_p, s_n) \sim \mathcal{D}_{\text{demo}}} [\max(\|G_\omega(s_q) - G_\omega(s_p)\|^2 - \|G_\omega(s_q) - G_\omega(s_n)\|^2 + m, 0)], \quad (7)$$

where s_q is the query sample, s_p is the positive sample, s_n is the negative sample, and m is a margin hyperparameter that enforces the distance to the negative sample to be greater than the distance to the positive sample by at least m .

During the online policy learning, we introduce a semi-dense reward mechanism based on “Anchor Points”. Instead of calculating an intrinsic reward at every timestep, we set Anchor Points at a fixed interval of K timesteps (e.g., $K = C$) along the exploration trajectory. The intrinsic reward calculation is activated only when a timestep t is an Anchor Point (i.e., $t \bmod K = 0$). This design naturally aligns with our chunk-based AC3 policy, effectively providing an intrinsic

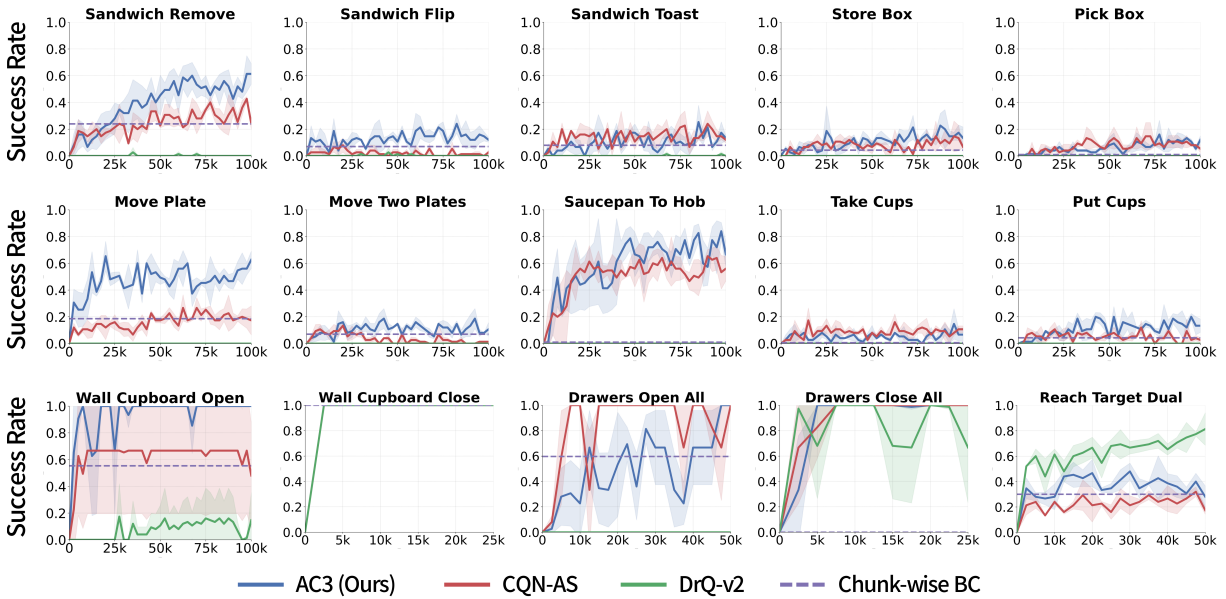


Figure 3: The performance of 15 bi-manual mobile manipulation tasks in BiGym. All tasks use 10 expert demonstrations as offline data, and all RL algorithms use an auxiliary BC loss for exploration guidance. The solid line and the shaded regions represent the mean performance and standard deviation, respectively.

reward signal to the entire action chunk and encouraging random exploration within it.

To formalize the reward calculation, we first define the minimum squared latent distance, $d(s_t)$, between the current state s_t and the set of demonstration states:

$$d(s_t) = \min_{s_d \in \mathcal{D}_{\text{demo}}} \|G_\omega(s_t) - G_\omega(s_d)\|^2. \quad (8)$$

Using this distance metric, the intrinsic reward $r_{\text{int}}(s_t)$ is then given by:

$$r_{\text{int}}(s_t) = \begin{cases} a & \text{if } t \equiv 0 \pmod{K} \text{ and } d(s_t) < m, \\ 0 & \text{otherwise.} \end{cases} \quad (9)$$

Here, m is equivalent to the margin threshold in Eq. 7, thus ensuring the anchor state is always near the successful trajectory. In addition, a can be any positive reward that is not greater than the task success reward, thereby functioning as an exploration anchor. We simply set 0.1 in our experiments.

Experiments

Experimental Setup

Dataset and Setup. We conduct evaluations on two robotic manipulation benchmarks with sparse reward. Details on experimental setup are available in supplementary materials.

- **BiGym:** BiGym (Chernyadev et al. 2024) is a benchmark focused on bi-manual mobile manipulation tasks. Each task contains human-collected demonstrations, with each trajectory receiving a reward of 1 only upon task success, and 0 otherwise. We select 15 tasks covering scenarios like multiple sub-tasks (e.g., *saucepan to hob*) and fine-grained operations (e.g., *move plate*). We use RGB observations

with 84×84 resolution from *head*, *left wrist*, and *right wrist* cameras. For each task, we select 10 demonstrations as offline expert data in each run.

- **RLBench:** RLBench (James et al. 2020) is a benchmark focusing on tabletop manipulation. We select 10 tasks, with the same reward setting as BiGym. We use RGB observations with 84×84 resolution from *front*, *wrist*, *left shoulder*, and *right shoulder* cameras. We maintain the same demonstration collection method as CQN (Seo, Uruç, and James 2024), with 100 demonstrations per task.

Baseline and Evaluation Metrics. We report the performance of four algorithms: 1) Our proposed AC3; 2) CQN-AS (Seo and Abbeel 2025), which learns action chunks using hierarchical discrete Q-learning; 3) DrQ-v2 (Yarats et al. 2021), an actor-critic-based deterministic policy algorithm that outputs actions for each timestep. To enable DrQ-v2 to converge to effective policies in sparse reward settings, we employ an additional auxiliary BC loss; 4) Chunk-wise BC, which uses the same policy (*i.e.*, Actor) model architecture of AC3 and only employs offline data for chunk-based imitation learning. We train 50k steps on each task, and take the mean performance of the last three checkpoints as the chunk-wise BC baseline, the quantitative results shown in Fig. 1(d) are also calculated using a similar rule. We report the success rate of 25 episodes at each checkpoint. In all figures we plot the mean performance over 3 seeds together with the shaded regions representing the standard deviation.

Implementation Details. We use the AdamW optimizer for all networks. The learning rates are set to 1.0×10^{-4} for the actor π_θ and critic Q_ϕ , and 1.0×10^{-5} for the goal network G_ω . For the basic RL settings, we set soft update coefficient for the target networks $\mu = 0.005$, discount factor $\gamma = 0.99$,

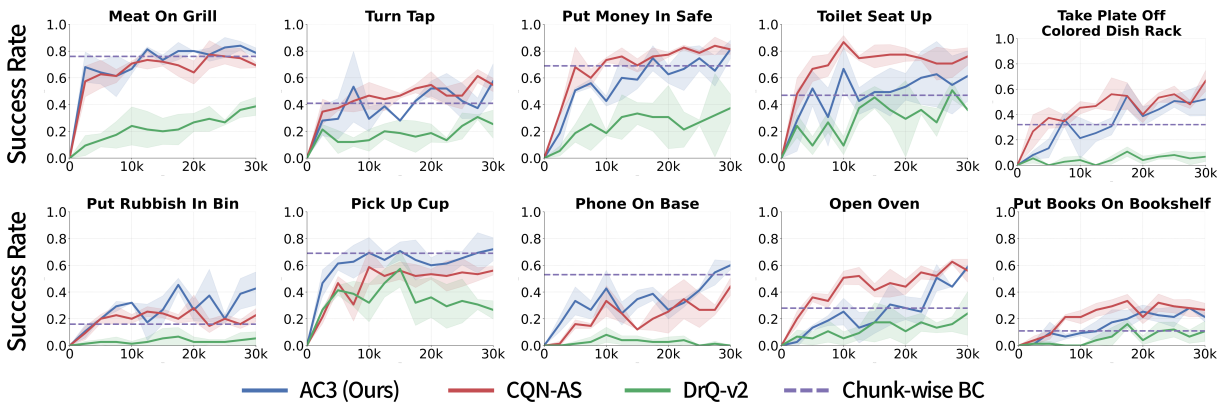


Figure 4: The performance of 10 tabletop manipulation tasks in RL Bench. All tasks use 100 synthetic demonstrations as offline data, and all RL algorithms use an auxiliary BC loss for exploration guidance. The solid line and the shaded regions represent the mean performance and standard deviation, respectively.

exploration noise $\sigma = 0.01$, and its clip threshold $c = 0.1$. We employ the temporal ensemble method proposed in ACT to smooth action execution. For intrinsic reward shaping, we set anchor interval $K = 16$, margin threshold $m = 0.5$, and intrinsic reward $a = 0.1$. In the actor loss function, $\lambda_{BC} = 1.0$ and $\lambda_Q = 0.1$. In the main results, we set the action chunk size $C = 16$. For network updates, BiGym and RL Bench tasks both use 4-step returns. All our experiments are conducted using a RTX 3090 GPU.

Performance Comparison

BiGym Results. As shown in Fig. 3, with the same few demonstrations (10 demos), AC3 achieves the highest task success rate in most tasks. Compared with the BC baseline, AC3 consistently discovers a better strategy and rapidly increases the success rate within 10k steps. For instance, in tasks like *sandwich remove* and *move plate*, its success rate curve exhibits a more prominent upward trend and maintains a relatively high level, indicating the effectiveness and stability of AC3 in exploring the high-dimensional action.

RL Bench Results. The tasks in RL Bench not only provide cleaner and more demonstrations but also feature smaller action spaces. As a result, as shown in Fig. 4, chunk-wise BC can typically learn effective policies for task completion. Building on this, AC3 still outperforms the BC baseline on most tasks and, with a simpler network architecture, achieves similar performance to CQN-AS. In contrast, DrQ-v2 performs significantly worse than the BC baseline on most tasks, demonstrating the critical role and necessity of chunk-based prediction in long-horizon, sparse-reward tasks.

Ablation Study and Analysis

In the ablation study, we primarily explore several key designs of AC3, with our experiments focusing mainly on comparing the representative *move plate* task from BiGym and *open oven* task from RL Bench: 1) *How the size of action chunks affects policy*; 2) *Whether the intra-chunk n-step return impacts policy stability*; 3) *Whether the self-supervised reward design facilitates policy training*; 4) *Training stability*

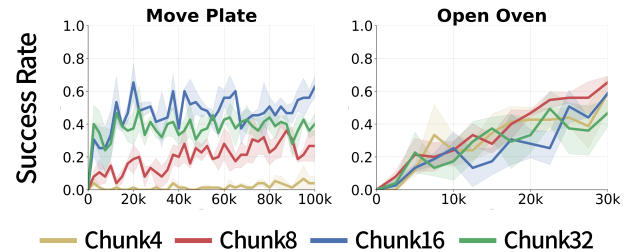


Figure 5: Effect of action chunk length.

and efficiency analysis. We provide a more comprehensive and detailed analysis in supplementary materials.

Discussion on action chunk length. The length of action chunks C is crucial for AC3 training. Larger action chunks provide longer action consistency and incorporate more temporal dependency information, but they also increase the dimensionality of the action exploration space and introduce more exploration noise. As shown in Fig. 5, in the *move plate* task with high-dimensional action space, a longer action chunk ensures the coherence and effectiveness of the bi-manual movements, whereas a setting of $C = 4$ renders the policy almost completely ineffective. In contrast, for the simpler *open oven* task with fewer DoF, overly long action chunks increase exploration difficulty and noise, while a moderate chunk length is sufficient to strike the optimal balance between action coherence and exploration efficiency.

Effectiveness of intra-chunk n-step return. In theory, $n = C$ is an unbiased choice under ideal conditions. However, this approach reduces the policy’s prediction units to entire chunks: larger chunks introduce greater exploration noise, and the critic fails to provide precise guidance for intra-chunk actions. In environments with limited demonstrations and sparse rewards, such noise and variance can be critical. As shown in Fig. 6, $n = C$ (i.e., 16) setting in the *move plate* task directly leads to policy failure, whereas performance is significantly better in *open oven* task—where offline data is more diverse and action space is relatively low.

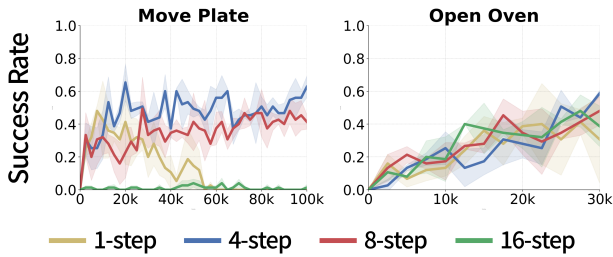


Figure 6: Effect of intra-chunk n-step return.

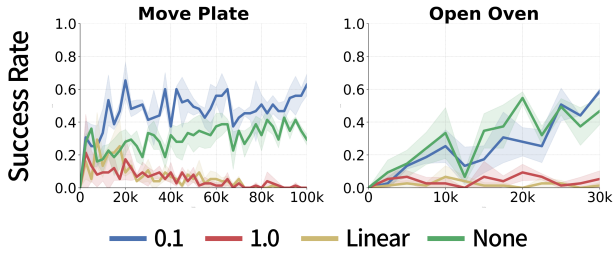


Figure 7: Reward shaping effects. *Linear*: Increases linearly with the current trajectory length; *None*: no reward shaping.

Intra-chunk n-step return serves as a compromise: it allows minor fine-tuning within chunks, reducing exploration noise. Intuitively, the critic evaluates short-term execution against a longer-term plan, enabling the model to better balance macro planning (*i.e.*, which chunk to execute) and micro control (*i.e.*, how to optimize intra-chunk actions). By leveraging the more stable learning signals from intra-chunk n-step returns, it effectively eases the learning difficulty under sparse rewards, thus achieving more efficient policy improvement. We find that $n = 4$ and $n = 8$ are both good choices, while $n = 1$ tends to cause Q-value explosion in higher-dimensional tasks, leading to policy failure in the later stages of training.

Analysis of self-supervised reward shaping setup. Setting intrinsic rewards for long-horizon, sparse-reward tasks is a common approach. We directly leverage a small amount of offline data for feature pre-training to generate anchor signals from successful trajectories, which in turn guide exploration. Fig. 7 shows the results for different reward settings. We find that compared to the baseline without reward shaping, $r_{\text{int}} = 0.1$ exhibits a significant performance advantage in the more complex *move plate* task, while also maintaining decent performance in simpler *open oven* tasks. This demonstrates that appropriate anchor point settings play a positive role in exploration. However, we also find that when the anchor signal is either identical to the task success reward, or increases linearly as the trajectory progresses, both cases lead to Q-value explosion and policy failure. We attribute this to the agent learning to exploit the misspecified intrinsic reward rather than pursuing the true sparse-reward objective.

Training Stability and Efficiency Analysis. To demonstrate the necessity of our asymmetric update rule, we conducted an ablation study allowing the actor to learn from all experiences, including failures. As shown in Fig. 8, this

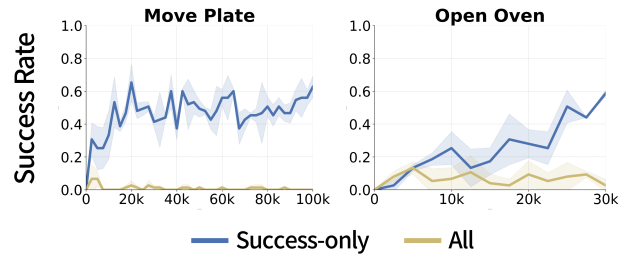


Figure 8: Comparison between actor learning from success-only and all trajectories.

Methods	Trainable Parameters (M)	Inference Speed (ms)
Chunk-wise BC	6.56 (Actor-only)	2.9
CQN-AS	28.58 (Q-Network)	9.5
AC3	14.44 (Actor-Critic)	2.9

Table 1: Comparison of model complexity and inference speed. For AC3, inference is performed using only the Actor network (6.56M parameters).

change caused the policy to degrade dramatically or collapse entirely. This failure occurs because, in high-dimensional, sparse-reward settings, the critic generates severely biased Q-value estimates for unseen states. These flawed estimates produce misleading gradients that disrupt the actor’s update. Therefore, AC3’s asymmetric rule, which shields the actor from these noisy signals, is crucial for stable training.

Furthermore, Tab. 1 highlights AC3’s efficiency. By using only its lightweight Actor (6.56M) for inference, AC3 requires just 2.9ms to predict an action chunk, making it over 3x faster than CQN-AS. This speed is vital for high-frequency environments, as it can accommodate the overhead of action smoothing like temporal ensembling. Ultimately, this makes AC3 not only robust but also a practical choice for low-latency deployment.

Conclusion

This paper introduces AC3 (Actor-Critic for Continuous Chunks), a novel framework designed to address long-horizon, sparse-reward robotic manipulation. AC3 directly learns to generate continuous action chunks, which enables more precise and flexible control. Its stability and data efficiency originate from two key stabilization mechanisms. First, an asymmetric actor update rule ensures reliable policy improvement by learning exclusively from successful trajectories. Second, the critic’s learning is stabilized with intra-chunk n-step returns and guided by a self-supervised intrinsic reward, which facilitates effective learning despite sparse rewards. We evaluate AC3 on 25 tasks from the Bi-Gym and RL Bench benchmarks. The results show that AC3 achieves superior success rates with only a simple model architecture and a few expert demonstrations. By effectively leveraging online self-improvement, AC3 presents a stable and data-efficient solution for complex manipulation tasks.

Acknowledgments

This work was supported by the Science and Technology Commission of Shanghai Municipality(No. 24511103100)

References

- Ankile, L.; Simeonov, A.; Shenfeld, I.; Torne, M.; and Agrawal, P. 2024. From Imitation to Refinement–Residual RL for Precise Assembly. *arXiv preprint arXiv:2407.16677*.
- Bharadhwaj, H.; Vakil, J.; Sharma, M.; Gupta, A.; Tulsiani, S.; and Kumar, V. 2024. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 4788–4795. IEEE.
- Chernyadev, N.; Backshall, N.; Ma, X.; Lu, Y.; Seo, Y.; and James, S. 2024. Bigym: A demo-driven mobile bi-manual manipulation benchmark. *arXiv preprint arXiv:2407.07788*.
- Chi, C.; Xu, Z.; Feng, S.; Cousineau, E.; Du, Y.; Burchfiel, B.; Tedrake, R.; and Song, S. 2023. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 02783649241273668.
- Doshi, R.; Walke, H.; Mees, O.; Dasari, S.; and Levine, S. 2024. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. *arXiv preprint arXiv:2408.11812*.
- Fu, Z.; Zhao, Q.; Wu, Q.; Wetzstein, G.; and Finn, C. 2024. Humanplus: Humanoid shadowing and imitation from humans. *arXiv preprint arXiv:2406.10454*.
- Fu, Z.; Zhao, T. Z.; and Finn, C. 2024. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*.
- Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, 1587–1596. PMLR.
- Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. 2018. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.
- Herzog, A.; Rao, K.; Hausman, K.; Lu, Y.; Wohlhart, P.; Yan, M.; Lin, J.; Arenas, M. G.; Xiao, T.; Kappler, D.; et al. 2023. Deep rl at scale: Sorting waste in office buildings with a fleet of mobile manipulators. *arXiv preprint arXiv:2305.03270*.
- Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33: 6840–6851.
- James, S.; Ma, Z.; Arrojo, D. R.; and Davison, A. J. 2020. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2): 3019–3026.
- Kalashnikov, D.; Irpan, A.; Pastor, P.; Ibarz, J.; Herzog, A.; Jang, E.; Quillen, D.; Holly, E.; Kalakrishnan, M.; Vanhoucke, V.; et al. 2018. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, 651–673. PMLR.
- Kalashnikov, D.; Varley, J.; Chebotar, Y.; Swanson, B.; Jonshkowsky, R.; Finn, C.; Levine, S.; and Hausman, K. 2021. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*.
- Lee, A.; Chuang, I.; Chen, L.-Y.; and Soltani, I. 2024. Interact: Inter-dependency aware action chunking with hierarchical attention transformers for bimanual manipulation. *arXiv preprint arXiv:2409.07914*.
- Li, Q.; Zhou, Z.; and Levine, S. 2025. Reinforcement Learning with Action Chunking. *arXiv preprint arXiv:2507.07969*.
- Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.
- Liu, S.; Wu, L.; Li, B.; Tan, H.; Chen, H.; Wang, Z.; Xu, K.; Su, H.; and Zhu, J. 2024. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*.
- Motoda, T.; Hanai, R.; Nakajo, R.; Murooka, M.; Erich, F.; and Domae, Y. 2025. Learning bimanual manipulation via action chunking and inter-arm coordination with transformers. *arXiv preprint arXiv:2503.13916*.
- Ren, A. Z.; Lidard, J.; Ankile, L. L.; Simeonov, A.; Agrawal, P.; Majumdar, A.; Burchfiel, B.; Dai, H.; and Simchowitz, M. 2024. Diffusion policy policy optimization. *arXiv preprint arXiv:2409.00588*.
- Ross, S.; Gordon, G.; and Bagnell, D. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 627–635. JMLR Workshop and Conference Proceedings.
- Saanum, T.; Éltető, N.; Dayan, P.; Binz, M.; and Schulz, E. 2023. Reinforcement learning with simple sequence priors. *Advances in Neural Information Processing Systems*, 36: 61985–62005.
- Seo, Y.; and Abbeel, P. 2025. Coarse-to-fine Q-Network with Action Sequence for Data-Efficient Robot Learning. *arXiv:2411.12155*.
- Seo, Y.; Uruç, J.; and James, S. 2024. Continuous control with coarse-to-fine reinforcement learning. *arXiv preprint arXiv:2407.07787*.
- Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211.
- Tian, D.; Li, G.; Zhou, H.; Celik, O.; and Neumann, G. 2025. Chunking the critic: A transformer-based soft actor-critic with N-step returns. *arXiv preprint arXiv:2503.03660*.
- Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.
- Wang, D.; Liu, C.; Chang, F.; and Xu, Y. 2025. Hierarchical Diffusion Policy: manipulation trajectory generation via contact guidance. *IEEE Transactions on Robotics*.

Yarats, D.; Fergus, R.; Lazaric, A.; and Pinto, L. 2021. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*.

Zhao, T. Z.; Kumar, V.; Levine, S.; and Finn, C. 2023. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*.