# Residual Off-Policy RL for Finetuning Behavior Cloning Policies

Anonymous authors

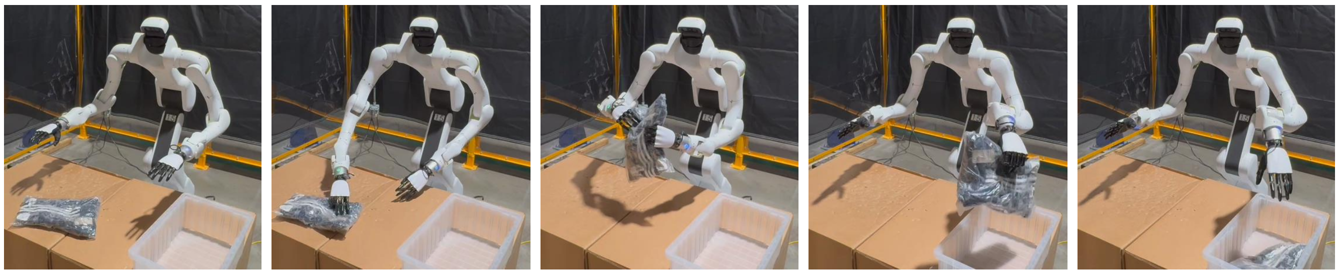


**Fig. 1.** Our residual RL method (ResFiT), performing real-world RL directly on our 29 Degree-of-Freedom (DoF) wheeled humanoid platform with two 5-fingered-hands, shown here performing the task of bimanual package handover.

*Abstract*— **Recent advances in behavior cloning (BC) have enabled impressive visuomotor control policies. However, these approaches are limited by the quality of human demonstrations, the manual effort required for data collection, and the diminishing returns from increasing data. In comparison, reinforcement learning (RL) trains an agent through autonomous interaction with the environment and has shown remarkable success in various domains. Still, training RL policies directly on real-world robots remains challenging due to sample inefficiency, safety concerns, and the difficulty of learning from sparse rewards for long-horizon tasks, especially for high-degree-of-freedom (DoF) systems. We present a recipe that combines the benefits of BC and RL through a residual learning framework. Our approach leverages BC policies as black-box bases and learns lightweight per-step residual corrections via efficient off-policy RL. We show that our method requires only sparse binary reward signals and can effectively learn manipulation tasks on high-DoF systems in both simulation and the real world. In particular, we demonstrate, to the best of our knowledge, the first successful real-world RL training on a humanoid robot with dexterous hands. Our results show state-of-the-art performance in various vision-based tasks, establishing a practical pathway to deploy RL in the real world. Videos and additional results can be found here.**

## I. INTRODUCTION

Developing algorithms to enable robust and high-performing autonomous robots in the real world has long been a goal of roboticists. Recently, significant progress has been made in training visuomotor control policies in the real world with behavior cloning (BC) from human demonstrations [1]–[9]. However, this success requires large amounts of infrastructure as well as many hours of manual and cumbersome human data collection. Even if unlimited data could be collected for every task, not only is human teleoperator performance generally suboptimal, but there is also emerging evidence that policy performance saturates with increasing demonstrations [6], [10]–[13].

Reinforcement learning (RL) offers a complementary paradigm: agents learn autonomously through trial-and-error interaction with the environment. Deep RL has shown great success in various domains [14]–[21], including in-hand manipulation [22], [23] and locomotion [24]–[27]. However, strong RL performance generally requires large amounts of data from online interactions, so its application has been mainly in simulation [28], [29] since real-world data are expensive and potentially unsafe to gather in large amounts.

A natural direction to improve BC policies is with online RL [12], [30]–[32], leveraging the strengths of each: BC policies provide a strong prior that can regularize exploration in the RL process, and online RL boosts policy performance by learning from environment interactions. However, modern BC architectures are typically deep models with tens of millions of parameters that use action chunking or diffusion-based approaches, which can make it challenging to apply RL methods to directly optimize the policy. A simple yet powerful recipe that avoids several of the above issues is residual RL [12], [31]–[36], where RL is applied not to learn a full policy, but only to learn corrective terms on top of a fixed base controller. Previous work has shown that residual RL can indeed be used to boost the reliability of a pre-trained policy, but it has so far been limited to learning in simulation [12], [31], [32] or demonstrating results in simple or constrained settings [33]–[36]; Applications to high-DoF systems learning directly in the real world are still lacking.

In this work, we present an off-policy residual fine-tuning (ResFiT) approach to use online RL to improve BC policies. By treating the base policy as a black box and learning a per-step residual correction independent of chunk size and policy parameterization, we sidestep the challenges of directly optimizing very large base policies. By carefully designing our off-policy recipe, we make the RL sample efficient enough to scale to high-DoF bimanual systems, require only sparse binary reward signals, and be safe enough to deploy in the real world. We demonstrate robust performance on sparse-reward long-horizon vision-based tasks, showing our approach achieves state-of-the-art performance for an array of tasks in simulation, and we investigate each design decision of our recipe. To the best of our knowledge, we provide the first demonstration of RL on a humanoid robot with five-fingered hands, trained entirely in the real world.

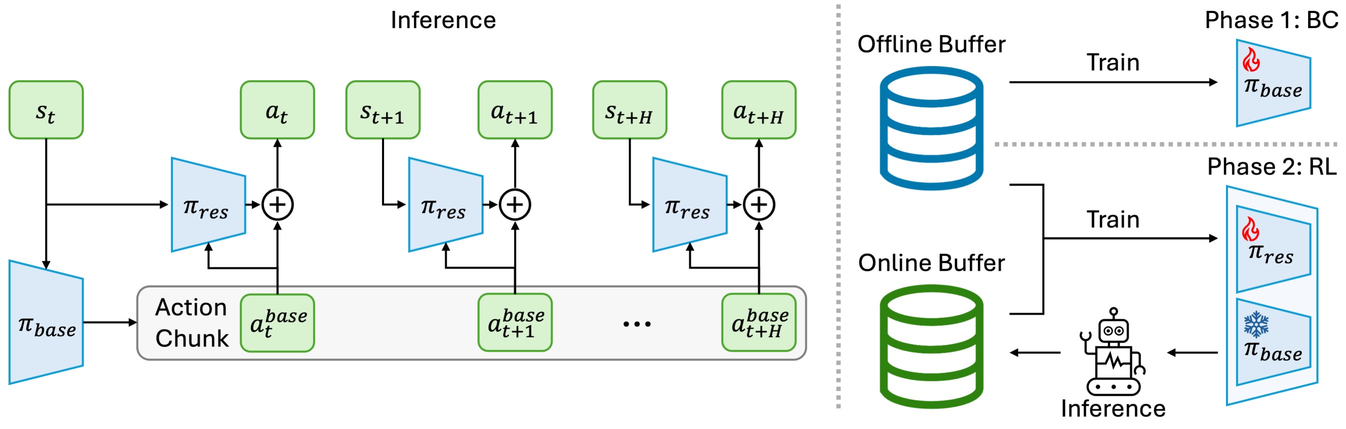


**Fig. 2.** Off-policy residual fine-tuning (ResFiT): A two-phase approach using online RL to improve BC policies. First, we train a base policy with BC on a dataset of demos and freeze it. Then, we learn a residual policy with RL to correct the base. The RL phase uses our off-policy recipe, leverages both demos and interactions, and enables more stable and safe exploration in the real world since we can directly control the magnitude of the residuals. This residual approach is agnostic to the base policy's parameterization and can even be applied to large action chunking BC policies, as single-step corrections.

## II. RELATED WORK

**Behavior Cloning.** BC has shown success in various settings, from autoregressive next-token prediction [1], [24], [37], [38] and diffusion policies [4], [6], [7], [39], [40], to large transformers that directly output robot actions [5], [41], [42]. Recent advances [8], [9], [43] leverage large transformers, diffusion and flow matching heads, and action chunking to handle long-horizon tasks and achieve impressive performance. Despite these advances, BC has fundamental limitations. The manual effort and infrastructure needed to scale up data collection to these levels are extremely high, and recent work has shown diminishing returns and performance plateaus with increasing data [6], [12], [13].

**Fine-tuning BC Policies with RL.** RL fine-tuning has been explored as a way to improve BC policies by letting the agent interact directly with the environment. However, directly fine-tuning modern BC architectures with RL presents significant challenges. State-of-the-art BC policies typically use action-chunking or diffusion-based approaches [4], [6] with large neural networks, which can lead to unstable learning when combined with conventional RL methods. Several works address these challenges [44], [45]. IBRL [46] trains an imitation policy and then uses it to propose actions for exploration and to bootstrap target values. PA-RL [30] sidesteps applying RL to complex model architectures by learning a Q-function to optimize actions instead of the policy. Other approaches [47]–[49] specifically adapt diffusion- or flow-based policies, such as DSRL [50], which runs RL over the latent-noise space of a frozen base diffusion policy and trains a policy to output that noise. Existing methods often couple the algorithm design with the BC policy structure, limiting flexibility across policy classes.

**Residual RL and Sample Efficiency.** Residual RL [33]–[36] provides an alternative approach of learning corrections on top of a fixed base policy, rather than optimizing the entire policy end-to-end. Recent work like ResiP [12] combines a modern action-chunked BC policy with a single-step RL residual policy, although its on-policy RL algorithm has too high sample complexity to be deployed in the real world.

Toward the goal of improved sample efficiency, prior work shows promising results through learning from demonstrations [51] and off-policy RL [52], [53]; drawing from a host of these related works [54]–[56], we show in this work that demos can serve multiple purposes: (1) to pre-train a BC policy, (2) to warm up a critic, and (3) to remain in a buffer and be leveraged throughout the online RL phase.

Closest to our work, Policy Decorator [32] and EXPO [31] demonstrate that off-policy RL can be used to train residual policies more efficiently. However, Policy Decorator learns residuals over entire action chunks rather than per-step corrections, while EXPO uses base policies without chunking. In both cases, the experiments are limited to simulation-only and single-arm tasks, and only state-based tasks for EXPO. In contrast, we demonstrate RL on tasks with higher degrees of freedom and longer horizons, and we show visuomotor policies on a bimanual humanoid robot in the real world.

**Real-World RL.** In the category of real-world RL, QT-Opt [57] used data from a fleet of 7 robots running over the span of weeks to teach a robotic arm with a parallel jaw gripper to grasp diverse objects. More recently, SERL [58] demonstrated insertion and pick-and-place policies by combining a few demonstrations with less than an hour of online interaction. A follow-up work [59] demonstrated more complex behaviors and bimanual control by using a human-in-the-loop approach to overcome the exploration problem. Multiple works have demonstrated RL in the real world for single-arm robots with grippers [46], [50], [60], [61], including [62], who showed that using Q-functions to guide batch online RL is more performant than imitation-based methods. On higher-DOF robots for locomotion [24], [25] and manipulation [13], [63], people often learn in simulation and rely on sim-to-real transfer to run in the real world.

## III. METHOD

Our method of off-policy RL for residual fine-tuning (ResFiT) of BC policies is illustrated in Fig. 2. It takes as a starting point a base policy, typically trained with BC on an offline dataset of demonstrations using any algorithm or architecture. In our experiments, we choose a base policy with action chunking. Then, we perform online RL using our sample-efficient off-policy recipe to learn residual corrections on top of the base policy. In the following, we outline the details of the full method, including the design choice and implementation details that are crucial to its success.

### A. Base Policy: Behavior Cloning with Action Chunking

Consider an agent that receives observation $o_t$ and performs action $a_t$ at each timestep $t$. We first collect a dataset $\mathcal{D}_{\text{demos}}$ of demonstrations, e.g., through human teleoperation in the real world, consisting of successful trajectories $\tau = (o_0, a_0, o_1, a_1, \dots)$. Given this dataset, we train a base policy $\pi_\psi(a_{t:t+k}|o_t)$ with behavior cloning to predict a sequence of $k$ future actions at each timestep. We train the policy to maximize the log-likelihood of the action chunks from the demonstrations, i.e., $\min_\psi -\sum_{o_t, a_{t:t+k} \in \mathcal{D}_{\text{demos}}} \log \pi_\psi(a_{t:t+k}|o_t)$.

Predicting a sequence of actions at each timestep instead of a single action is known to improve performance [4], [5], [64], [65] and alleviate compounding errors in imitation learning by effectively reducing the task horizon of the problem.

Note that although our policies receive only proprioception and image observations $o_t$ as input, and do not have access to the true underlying system state $s_t$, the remainder of this section will use the state notation for simplicity.

### B. fine-tuning with Off-Policy Residual RL

The above behavior cloning process produces a policy $\pi_{\text{base}}$ that has some level of task success. We freeze the base policy and train a new policy $\pi_{\text{res}}$ with RL to learn how to correct mistakes made by $\pi_{\text{base}}$ and improve policy performance in a way that is (1) agnostic to how $\pi_{\text{base}}$ is parameterized and trained and (2) stable since we can control the magnitude of this residual to stay relatively close to the base policy, especially during earlier exploration phases.

Consider a Markov decision process (MDP) [66] with states $s_t \in S$, actions $a_t \in A$, rewards $r_t = R(s_t, a_t)$, discount factor $\gamma$, and horizon $H$. RL aims to learn the parameters $\theta$ of some policy $\pi_\theta(s_t)$ such that the expected sum of rewards $R(\tau) = \sum_{t=0}^{H} \gamma^t r_t$ for a trajectory $\tau$ is maximized under the trajectory distribution induced by the policy. In this work, we will learn both a critic $Q_\phi$ as well as a policy $\pi_\theta$, as shown in DDPG [67].

Whereas standard off-policy RL methods learn $Q_\phi(s_t, a_t)$ and $\pi_\theta(s_t)$, we reparameterize this for our residual setting as $Q_\phi(s_t, a_t^{\text{base}} + \pi_\theta(s_t, a_t^{\text{base}}))$ and $\pi_\theta(s_t, a_t^{\text{base}})$. In other words, the residual policy receives the current observation as well as the base action; the full action is the sum $a_t = a_t^{\text{base}} + a_t^{\text{res}}$, and the critic predicts the values of the full actions.

**Algorithm 1** ResFiT: Residual fine-tuning w/ Off-Policy RL

```
    Input: pre-trained base $\pi_b$, dataset of demos $\mathcal{D}_{\text{offline}}$
    Initialize: residual policy $\pi_\theta$, Q ensemble $Q_{\phi_1}, \dots Q_{\phi_N}$, vision encoder $f_\omega$, empty buffer $\mathcal{D}_{\text{online}}$
    Initialize: set target networks $\theta' \leftarrow \theta$, $\phi'_i \leftarrow \phi_i$
 1: for step = 1, 2, ..., buffer_warmup_steps do
 2:     Sample noise $\epsilon_t \sim \mathcal{U}(-\text{noise\_scale}, \text{noise\_scale})$
 3:     Step env with $a_t = \epsilon_t + a_t^{\text{b}}$ where $a_t^{\text{b}} \sim \pi_b(s_t)$
 4:     Observe next state $s_{t+1}$, reward $r_t$, done flag $d_t$
 5:     Add transition $(s_t, a_t^{\text{b}}, a_t, s_{t+1}, a_{t+1}^{b}, r_t, d_t)$ to $\mathcal{D}_{\text{online}}$
 6: end for
 7: repeat
 8:     $a_t^{\text{b}} \sim \pi_b(s_t)$ and $a_t^{\text{r}} \sim \pi_\theta(s_t, a_t^{\text{b}})$
 9:     Step env with $a_t = a_t^{\text{b}} + a_t^{\text{r}}$
10:     Add $(s_t, a_t^{\text{b}}, a_t, s_{t+1}, a_{t+1}^{b}, r_t, d_t)$ to $\mathcal{D}_{\text{online}}$
11:     Repeat UTD times:
12:         Sample batch $B$ evenly from $\mathcal{D}_{\text{offline}}$, $\mathcal{D}_{\text{online}}$
13:         Update critic using $B$:
14:             $a_{t+1}^{\text{r}} \sim \pi_{\theta'}(s_{t+1}, a_{t+1}^{\text{b}})$
15:             Compute $a_{t+1} = a_{t+1}^{\text{b}} + a_{t+1}^{\text{r}}$
16:             $y = r_t + (1 - d_t) * \gamma * \min_{\text{subset(i)}} Q_{\phi'_i}(s_{t+1}, a_{t+1})$
17:             Update $\phi_i, \omega$ to minimize MSE$(Q_{\phi_i}(s_t, a_t), y)$
18:             Update critic targets $\phi'_i \leftarrow \rho\phi'_i + (1 - \rho)\phi_i$
19:     Update actor using latest $B$:
20:         Update $\theta$ to maximize:
                $\frac{1}{N}\sum_{i=1}^{N} Q_{\phi_i}(s_t, \pi_\theta(s_t, a_t^{\text{b}}) + a_t^{\text{b}})$
21:         Update actor target $\theta' \leftarrow \rho\theta' + (1 - \rho)\theta$
22: until convergence
```

For the critic, recall the Bellman equation describing the optimal action-value function for a standard $Q^\star(s_t, a_t)$:

$$Q^\star(s_t, a_t) = \mathop{\mathrm{E}}_{s_{t+1} \sim P}\left[r_t + \gamma \max_{a'} Q^\star(s_{t+1}, a')\right]. \quad (1)$$

We train our value approximator $Q_\phi$ to approximate $Q^\star$ by using the Mean-Squared Bellman Error (MSBE) loss, which simply tells us how close our approximator is to satisfying the Bellman equation. Given a dataset $\mathcal{D}$ of transitions $(s_t, a_t, r_t, s_{t+1}, d_t)$, $s_{t+1}$ is the resulting state from taking action $a_t$ from state $s_t$, $r_t$ is the resulting reward, and $d_t$ indicates whether state $s_{t+1}$ is terminal. Using our policy in place of the $\max Q$, the loss for our residual setting becomes:

$$L(\phi) = \mathop{\mathrm{E}}_{(s_t, a_t, r_t, s_{t+1}, d_t) \sim \mathcal{D}}\Bigg[\bigg(Q_\phi(s_t, a_t) - \quad (2)$$
$$\left(r_t + \gamma(1-d)Q_\phi(s_{t+1}, a_{t+1}^{\text{base}} + \pi_\theta(s_{t+1}, a_{t+1}^{\text{base}}))\right)\bigg)^2\Bigg]$$

where $a_{t+1}^{\text{base}} = \pi_{\text{base}}(s_{t+1})$. Moving on to the policy, recall that given an optimal action-value function $Q^\star(s, a)$, the optimal action $a^\star$ can be found by simply taking the max over actions of $Q^\star(s, a)$. So in our case of continual actions, and given that Q is differentiable, we can train our policy

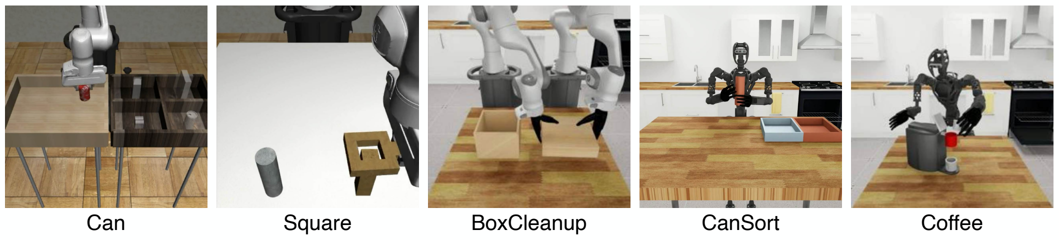


**Fig. 3.** Our simulation tasks from Robomimic and DexMimicGen, spanning single-arm manipulation as well as bimanual coordination tasks.

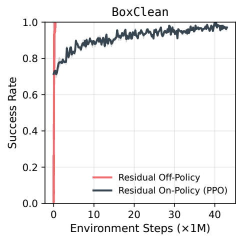


**Fig. 4.** PPO vs. off-policy RL in ResFiT.

$\pi_\theta(s)$ by simply performing gradient ascent on the value function, with respect to the policy parameters:

$$L(\theta) = -\mathop{\mathrm{E}}_{(s_t, a_t^{\text{base}})\sim\mathcal{D}} \left[ Q_\phi(s_t, a_t^{\text{base}} + \pi_\theta(s_t, a_t^{\text{base}})) \right]. \quad (3)$$

### C. Design Decisions

As described above, our approach uses off-policy data and interleaves using the Bellman equation to train the Q-function with using the Q-function to train the policy. In the following, we detail the key design choices that were required to achieve stable residual fine-tuning performance across different tasks and embodiments, present our complete method in Algorithm 1, and later validate these decisions through ablations.

We use Update-to-Data ratio (UTD) $> 1$, as prior works have shown that increasing the UTD, i.e., the number of model updates one takes per data point collected in the environment, can be an effective way of improving sample efficiency [56], [68]. We also use $n$-step returns (in particular, our $n = 3$), similar to CQN [60] and IBRL [46], which is shown to help with long-horizon and sparse-reward tasks [69]. Whereas Monte Carlo methods roll out the entire trajectory before doing updates, and standard Q-learning looks ahead just 1 step with value bootstrapping $r_t + \gamma Q(s_{t+1}, a_{t+1})$, here we instead look ahead $n$ steps: $\sum_{i=0}^{n-1} \gamma^i r + \gamma^n Q(s_{t+n}, a_{t+n})$ [70]. Note that we omit this detail in Algorithm 1 for simplicity.

A common issue for off-policy RL is that when the Q-function gets queried with out-of-distribution actions, the values can often be overestimated due to the use of function approximation. Drawing from RLPD [56], we add layer normalization [71] to the critic to mitigate catastrophic over-estimation without explicitly constraining the policy, which turns out to be crucial to the performance of the policy.

We also incorporate several other established practices in our implementation. Following TD3 [53], we employ delayed actor updates (every 2-8 critic updates) to alleviate instability that stems from updates done with poor value estimates, we use Polyak averaging ($\tau = 0.005$) to update target networks, and we perform target policy smoothing by adding noise to the action that forms the Q target, to help with brittle behavior caused by sharp peaks in learned Q function approximators. We also use Randomized Ensembled Double Q-Learning [68] for reducing overestimation bias, by computing TD-targets as $y = r + \gamma \min_{i\in S} Q_{\phi_i}(s', \pi_\theta(s'))$ from a random subset $S$ of $N$ Q functions, while policy updates utilize the full ensemble: $\nabla_\theta \frac{1}{N}\sum_{i=1}^{N} Q_{\phi_i}(s, \pi_\theta(s))$. For visual inputs, we adopt a shallow ViT [72] encoder with DrQ-style [73] random shift augmentations to help with overfitting. Finally, we use our demonstration data not only to train our base policy but also during the online RL phase via symmetric sampling [56], by sampling 50% of each batch from this frozen offline data, and the remaining 50% from our constantly growing online buffer.

## IV. EXPERIMENTAL SETUP

### A. Simulation Experiments

We evaluate ResFiT's real-world tractability via simulation experiments using realistic constraints: a single simulation environment, image and robot joint state observations (no privileged object state information), and sparse binary rewards. We test on tasks from Robomimic [74] and DexMimicGen [13], with varying robots, control modes, degrees of freedom, task horizons, and precision requirements. Our experiments use the Robosuite environment [75] built on MuJoCo [76], and we operate at 20 Hz control frequency.

**Tasks.** Our task selection (see Fig. 3) spans two categories: single-arm manipulation tasks and bimanual coordination tasks. For single-arm, we use `Can` and `Square` from Robomimic. These tasks use a Franka robot with a parallel-jaw gripper and have an action space dimension of 7, 6 for the delta end-effector pose, and 1 for the gripper action. To demonstrate the versatility of ResFiT, we also include the bimanual `BoxCleanup`, `CanSort`, and `Coffee` tasks from DexMimicGen, which require coordinated control of two arms with 6-DoF dexterous hands. `BoxCleanup` uses dual Frankas to coordinate picking up a box lid and placing it precisely on top of the box, testing spatial coordination and alignment. `CanSort` and `Coffee` both use a GR1 humanoid robot with a fixed base; `CanSort` entails picking up and passing a cylinder between its hands, and `Coffee` requires dexterously grasping a coffee pod and placing it precisely into a coffee maker before closing the lid with the other hand. The bimanual tasks have action spaces with 24 dimensions: 6 for the absolute end-effector pose per arm, and 6 for the actuated joint positions per hand. Note that the end-effector control modes differ between these robots, dual Frankas use delta pose control (same as single-arm), while the GR1 uses absolute pose control. For all tasks, we use the released demonstration datasets with 300 demos from the Multi-Human dataset per single-arm task and 1000 demos created with DexMimicGen per bimanual task.

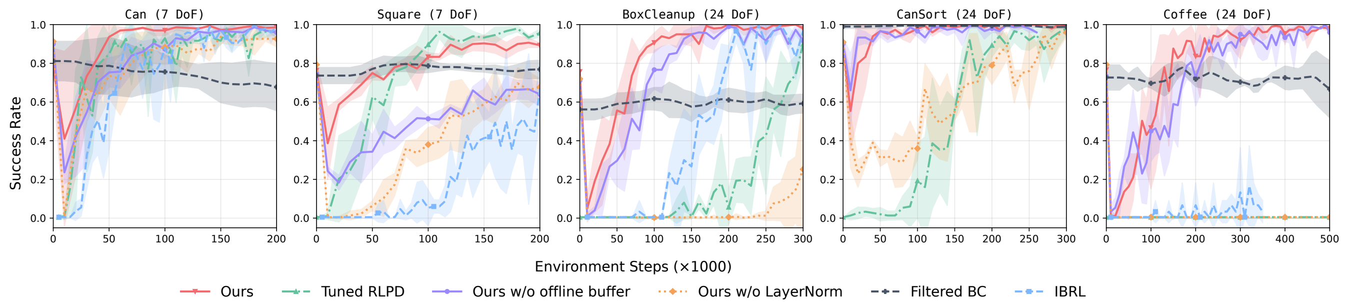


**Fig. 5.** Success rates of different approaches on our simulation tasks, showing ResFiT converging to high-performing policies for all tasks. Note that all approaches here start with the same number of demos.

**Baselines and Ablations.** First, we examine how off-policy residual RL on top of an action-chunked base policy compares to directly performing off-policy RL to learn a single-action policy, starting from the same demos. For this, we chose the state-of-the-art algorithm RLPD [56], an off-policy approach that includes both offline demos and online data in each training batch, and uses layer norm to help alleviate the overestimation of the value function. However, with default settings, it is unable to solve our high-DoF tasks when learning from images. Instead, we create an optimized version of RLPD by incorporating the same off-policy RL design decisions (see Sec. III-C) that we use in our method. We refer to this off-policy RL implementation as "Tuned RLPD". This means that the same number of demos are used, the same off-policy RL algorithm is used, and the same vision encoders and network architectures are used; the main difference is whether a base policy is being used. For another baseline of using demos but not performing residual learning, we compare to IBRL [46], which uses a pre-trained BC policy during RL to propose actions and bootstrap target values. Finally, we include an online BC fine-tuning baseline called "Filtered BC," which starts with the same base policy as ResFiT, but does not use residual RL to perform the fine-tuning. It fine-tunes by iteratively performing rollouts and adding successes back into the dataset for continued BC training. This approach is similar in spirit to reward-weighted regression [77] (with 0/1 weights) and self-imitation [78], [79], which prior work has shown can improve performance over the demo-only case [2], [65], [80], [81].

Other design decisions that we ablate include whether we use the layer norm for the actor and critic MLPs, whether we incorporate the demos during the online RL phase, the choice of the UTD ratio, and the choice of $n$ for $n$ step returns used in the TD targets. Finally, we also compare to the more common version of residual RL [12], which is to use online RL (PPO) to learn residuals.

## B. *Real-World Experiments*

We demonstrate ResFiT on a real-world bimanual dexterous manipulation platform - the wheeled Vega humanoid from Dexmate. Vega has two 7-DoF arms, two 6-DoF OyMotion dexterous hands, and 2 image streams coming from a Zed camera mounted on its 3-DoF head. We use absolute joint position control in the real world for each of these joints, making the action space dimension 29.

**Tasks.** We perform two tasks in the real world: `WoollyBallPnP` and `PackageHandover`. In `WoollyBallPnP`, the right hand picks up a ball from a random table location and puts it into a randomly placed tote. In `PackageHandover`, the right hand picks up a deformable package, hands it to the left hand, and places it into a tote on the left side of the workspace.

**Real-world Infrastructure.** Our real-world setup includes safety limits implemented using the force-torque sensor in the robot's wrists, and a collision checking implementation to prevent robot self-collisions. We built a system for performing RL in the real world, where the operator either lets each episode timeout or marks the result of each episode as success or failure, and then resets the scene. During RL training, we noticed that the bottleneck in wall-clock time was due to the model updates, as opposed to the data collection, due to the high UTD ratio. To enable higher data throughput, we split the actor and the learner into separate processes. In particular, after a completed trajectory that either succeeded, failed, or timed out, the learner process loads the latest replay buffer and performs model updates, while the data collector resets the scene and starts the next trajectory with the actor process. Note that the only reward for these rollouts was a single 1 for success and 0 otherwise.

**Evaluation protocol.** Real-world robot evaluations often suffer from confounding factors like environmental drift and lighting changes, especially when policies are evaluated sequentially rather than in matched conditions [9], [82]. To address these issues and fairly quantify performance differences between pre-trained and fine-tuned policies, we use blind A/B testing with matched initial conditions. For each evaluation round, we (1) sample a random scene configuration (object, tote), (2) randomly assign each policy to labels A and B, (3) execute both policies from identical initial states, and (4) reveal policy identities after completion. This approach attempts to mitigate evaluator bias, controls for sensitivity to initial conditions, and allows both policies to face similar environmental conditions within each round.

## V. RESULTS

## A. *Simulation Results*

We evaluate various aspects of ResFiT on simulated benchmark tasks, consisting of both single-arm and bimanual

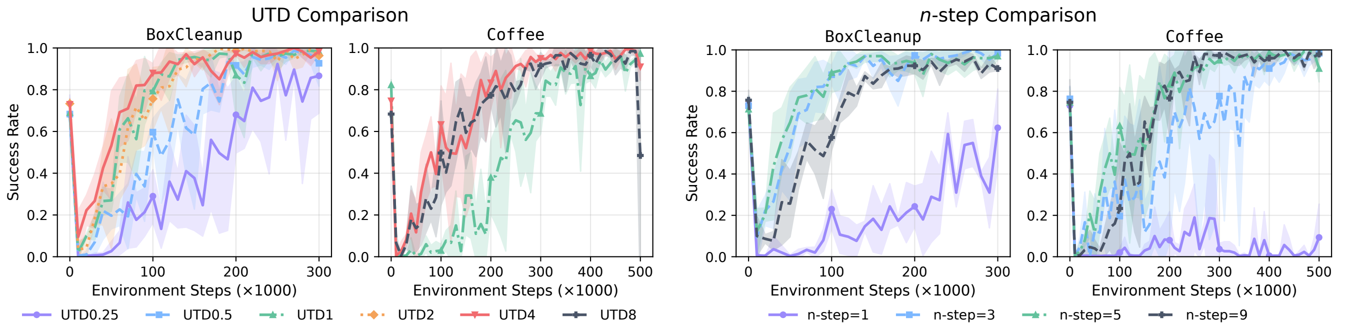
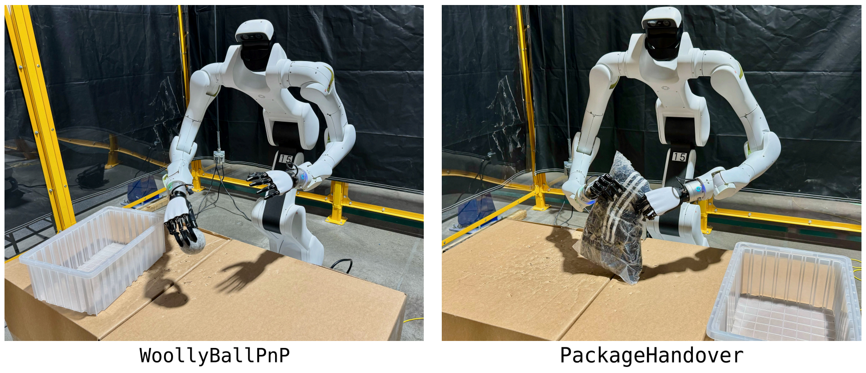


**Fig. 6.** Impact of UTD ratio (left) and `n-step` (right) on the `BoxCleanup` and `Coffee` tasks.



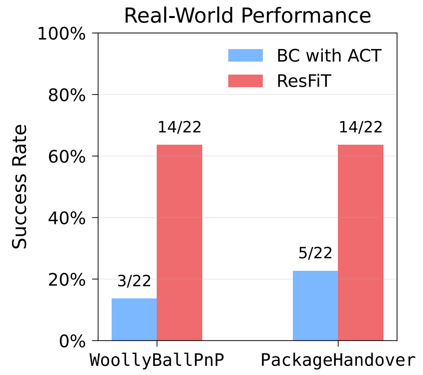


**Fig. 7.** Real-world results of applying ResFiT, where our residual RL approach shows a significant boost in performance over the base model, on a 29-DoF bimanual wheeled humanoid robot with two 5-fingered hands.

manipulation tasks. On the simulated `BoxCleanup` task, we first compare our off-policy residual RL recipe to an existing approach [12] of performing residual RL in sim with the on-policy PPO [83] algorithm. Here, with our approach converging at 200k steps versus 40M steps, we see a ∼200× boost in sample efficiency, demonstrating the need for off-policy approaches when considering performing the RL directly in the real world.

Examining the main simulation results in Figure 5, we first observe that ResFiT converges to near-perfect policies for all tasks. We find that for the simplest task `Can`, the baseline off-policy methods as well as the ablated versions of our method all achieve high success rates in ∼150k steps, while ResFiT converges faster, at ∼75k steps. In the `Square` task, only ResFiT and our optimized off-policy approach ("Tuned RLPD"), which does not use a base policy or residuals but does incorporate all of ResFiT's other off-policy RL design decisions, can achieve over 90% success rate in 150k steps.

For the harder tasks with more DoFs and longer horizons, like `BoxCleanup`, `CanSort`, and `Coffee`, baselines and ablated versions either collapse to zero success rate or take longer to achieve high performance, while ResFiT can still converge to near-perfect task performance efficiently. On `Coffee`, which has the longest task horizon while still requiring the most precision, we notice that not having demos during the online RL of ResFiT phase may not matter, but all of the approaches without action-chunking fail in this sparse reward and vision-based setting.

Across all tasks, the Filtered BC baseline remained stable but showed minimal improvement over the initial BC policy performance. We hypothesize that this is because the main failure mode is precision, which is hard to gain without explicit value maximization. In general, we find that Filtered BC is able to improve BC policies that start from a lower initial performance (e.g., ∼20%), but quickly saturate, which is supported by previous work [65].

In Figure 6, we study the effect of UTD and `n-step` returns for sparse reward tasks. Here, we see the importance of using an `n-step` larger than 1 for tasks where rewards are sparse. However, since a larger `n-step` also increases bias, a too large value can affect performance. For UTD ratios, we find that for a task with a horizon length of 150-250 steps, such as `BoxCleanup`, UTD values greater than 1 provide clear benefits, but gains plateau at moderate values. Specifically, learning is noticeably slower for a UTD of $\frac{1}{2}$, but increasing to very high UTDs of 8 or higher, as in some prior work, yields diminishing returns, with UTDs of 4 already capturing most of the benefit while still being stable.

## B. *Real-World RL Results*

In the real world, we apply our methods to two tasks, `WoollyBallPnP` and `PackageHandover`. We use ACT [6] as the base policy for both tasks.

For `WoollyBallPnP`, we trained our base ACT policy on a pick-and-place task with around 1,000 demonstrations across 4 different objects with random robot start locations, random object locations, and random placement tote locations. We then evaluated it on the object that it most struggled with: a small, gray, and woolly ball. The base policy achieved only 14% success on this task, with the main failure mode being hovering and missed grasps with the dexterous hand. With 134 rollouts of autonomous RL execution ($\approx$ 15 minutes worth of robot execution data) specifically on this woolly ball, ResFiT boosted the performance of the base model from 14% to 64%.

`PackageHandover` is a more difficult task since it requires two-arm coordination and involves a longer task horizon before experiencing any potential reward. For this task, our base policy was able to achieve a 23% success rate with around 900 demonstrations, and after 343 RL episodes ($\approx$ 76 minutes worth of data) in the real world, ResFiT was able to boost the task performance to 64%. To the best of our knowledge, this is the first demonstration of real-world RL performed on a bimanual dexterous manipulation humanoid with two five-fingered hands, trained fully in the real world.

## VI. DISCUSSION

We demonstrate that decoupling the pre-training and fine-tuning stages can resolve a tension in the current robot learning paradigm: action chunking and larger policies improve BC performance but create intractably high-dimensional action spaces for RL (870 dims in our case). By learning only single-step corrections atop a frozen base policy, we maintain long-horizon reasoning while keeping optimization tractable.

An insight from our experiments is that the base policy serves a dual purpose beyond providing a good initialization. First, it acts as an implicit safety constraint: Policies trained without this regularization converge to faster but more aggressive behaviors that are unsuitable for real-world deployment. Second, it provides a powerful exploration prior that enables RL from sparse rewards even in high-DOF spaces. This suggests that hybrid BC-RL approaches are promising for high-DoF and long-horizon manipulation.

The approach's primary limitation is that learned behaviors may remain constrained around the base policy. Although our method clearly improves success rates in both simulation and real experiments, it cannot discover fundamentally different strategies or skills beyond what the base policy already encodes while remaining stable overall.

Looking ahead, figuring out the right way to relax the frozen base constraint without sacrificing stability could provide further improvement in performance and robustness. If we can distill the more precise, reliable, and fast behavior from the combined policy back into the base policy, that would provide more room for the residual model to improve further. This would be particularly powerful in the multitask setting, where one can distill task-specific residual improvements into an increasingly capable generalist.

Our real-world deployment validates that sample-efficient RL can work on bimanual platforms with 5-fingered hands. However, our experiments still required human supervision for resets and reward labeling. Without automatic reset mechanisms, success detection, and safety rails, even sample-efficient RL methods cannot enable autonomous skill improvement that scales independently of human oversight.

## REFERENCES

[1] A. Brohan and et.al., "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[2] K. Bousmalis, G. Vezzani, D. Rao, C. Devin, A. X. Lee, M. Bauzá, T. Davchev, Y. Zhou, A. Gupta, A. Raju *et al.*, "Robocat: A self-improving generalist agent for robotic manipulation," *arXiv preprint arXiv:2306.11706*, 2023.

[3] A. Brohan and et.al., "RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control," 2023, arXiv:2307.15818.

[4] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02783649241273668, 2023.

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

[6] T. Z. Zhao, J. Tompson, D. Driess, P. Florence, K. Ghasemipour, C. Finn, and A. Wahid, "Aloha unleashed: A simple recipe for robot dexterity," *arXiv preprint arXiv:2410.13126*, 2024.

[7] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter *et al.*, "$\pi$0: A vision-language-action flow model for general robot control," *arXiv preprint ARXIV.2410.24164*, 2024.

[8] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang *et al.*, "Gr00t n1: An open foundation model for generalist humanoid robots," *arXiv preprint arXiv:2503.14734*, 2025.

[9] J. Barreiros, A. Beaulieu, A. Bhat, R. Cory, E. Cousineau, H. Dai, C.-H. Fang, K. Hashimoto, M. Z. Irshad, M. Itkina *et al.*, "A careful examination of large behavior models for multitask dexterous manipulation," *arXiv preprint arXiv:2507.05331*, 2025.

[10] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 661–668. [Online]. Available: https://proceedings.mlr.press/v9/ross10a.html

[11] A. Yu, G. Yang, R. Choi, Y. Ravan, J. Leonard, and P. Isola, "Learning visual parkour from generated images," in *8th Conference on Robot Learning*, 2024.

[12] L. Ankile, A. Simeonov, I. Shenfeld, M. Torne, and P. Agrawal, "From imitation to refinement–residual rl for precise assembly," *arXiv:2407.16677*, 2024.

[13] Z. Jiang, Y. Xie, K. Lin, Z. Xu, W. Wan, A. Mandlekar, L. Fan, and Y. Zhu, "Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning," 2025.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, 2015.

[15] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[16] A. Farooq and K. Iqbal, "A survey of reinforcement learning for optimization in automation," in *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2024, pp. 2487–2494.

[17] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nova *et al.*, "A graph placement methodology for fast chip design," *Nature*, vol. 594, no. 7862, pp. 207–212, 2021.

[18] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi *et al.*, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025.

[19] Q. Yu, Z. Zhang, R. Zhu, Y. Yuan, X. Zuo, Y. Yue, W. Dai, T. Fan, G. Liu, L. Liu *et al.*, "Dapo: An open-source llm reinforcement learning system at scale," *arXiv preprint arXiv:2503.14476*, 2025.

[20] S. Wang, S. Zhang, J. Zhang, R. Hu, X. Li, T. Zhang, J. Li, F. Wu, G. Wang, and E. Hovy, "Reinforcement learning enhanced llms: A survey," *arXiv preprint arXiv:2412.10400*, 2024.

[21] M. Torne, A. Simeonov, Z. Li, A. Chan, T. Chen, A. Gupta, and P. Agrawal, "Reconciling Reality through Simulation: A Real-to-Sim-to-Real Approach for Robust Manipulation," 2024, arXiv:2403.03949.

[22] T. Chen, J. Xu, and P. Agrawal, "A system for general in-hand object reorientation," in *Conference on Robot Learning*. PMLR, 2022, pp. 297–307.

[23] H. Qi, B. Yi, S. Suresh, M. Lambeta, Y. Ma, R. Calandra, and J. Malik, "General in-hand object rotation with vision and touch," in *Conference on Robot Learning*. PMLR, 2023, pp. 2549–2564.

[24] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Real-world humanoid locomotion with reinforcement learning," *Science Robotics*, vol. 9, no. 89, p. eadi9579, 2024.

[25] X. Gu, Y.-J. Wang, X. Zhu, C. Shi, Y. Guo, Y. Liu, and J. Chen, "Advancing humanoid locomotion: Mastering challenging terrains with denoising world model learning," *arXiv preprint arXiv:2408.14472*, 2024.

[26] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," *The International Journal of Robotics Research*, vol. 43, no. 4, pp. 572–587, 2024.

[27] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[28] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE symposium series on computational intelligence (SSCI)*. IEEE, 2020, pp. 737–744.

[29] L. Da, J. Turnau, T. P. Kutralingam, A. Velasquez, P. Shakarian, and H. Wei, "A survey of sim-to-real methods in rl: Progress, prospects and challenges with foundation models," *arXiv preprint arXiv:2502.13187*, 2025.

[30] M. S. Mark, T. Gao, G. G. Sampaio, M. K. Srirama, A. Sharma, C. Finn, and A. Kumar, "Policy Agnostic RL: Offline RL and Online RL Fine-Tuning of Any Class and Backbone," Dec. 2024, arXiv:2412.06685 [cs]. [Online]. Available: http://arxiv.org/abs/2412.06685

[31] P. Dong, Q. Li, D. Sadigh, and C. Finn, "Expo: Stable reinforcement learning with expressive policies," *arXiv preprint arXiv:2507.07986*, 2025.

[32] X. Yuan, T. Mu, S. Tao, Y. Fang, M. Zhang, and H. Su, "Policy decorator: Model-agnostic online refinement for large policy model," *arXiv preprint arXiv:2412.13630*, 2024.

[33] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, "Residual policy learning," *arXiv preprint arXiv:1812.06298*, 2018.

[34] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 international conference on robotics and automation (ICRA)*.

[35] M. Alakuijala, G. Dulac-Arnold, J. Mairal, J. Ponce, and C. Schmid, "Residual reinforcement learning from demonstrations," *arXiv:2106.08050*, 2021.

[36] S. Haldar, J. Pari, A. Rai, and L. Pinto, "Teach a robot to fish: Versatile imitation from one minute of demonstrations," *arXiv preprint arXiv:2303.01497*, 2023.

[37] Z. J. Cui, Y. Wang, N. M. M. Shafiullah, and L. Pinto, "From play to policy: Conditional behavior generation from uncurated robot data," *arXiv preprint arXiv:2210.10047*, 2022.

[38] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi *et al.*, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[39] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu *et al.*, "Octo: An open-source generalist robot policy," *arXiv preprint arXiv:2405.12213*, 2024.

[40] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *IEEE/CVF international conference on computer vision*, 2023.

[41] M. J. Kim, C. Finn, and P. Liang, "Fine-tuning vision-language-action models: Optimizing speed and success," *arXiv preprint arXiv:2502.19645*, 2025.

[42] L. Wang, X. Chen, J. Zhao, and K. He, "Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers," *Advances in neural information processing systems*, vol. 37, pp. 124 420–124 450, 2024.

[43] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai *et al.*, "pi0.5: a vision-language-action model with open-world generalization," *arXiv preprint arXiv:2504.16054*, 2025.

[44] Y. Chen, D. K. Jha, M. Tomizuka, and D. Romeres, "Fdpp: Fine-tune diffusion policy with human preference," *arXiv preprint arXiv:2501.08259*, 2025.

[45] P. Hansen-Estruch, I. Kostrikov, M. Janner, J. G. Kuba, and S. Levine, "IDQL: Implicit Q-Learning as an Actor-Critic Method with Diffusion Policies," 2023, arXiv:2304.10573.

[46] H. Hu, S. Mirchandani, and D. Sadigh, "Imitation bootstrapped reinforcement learning," *arXiv preprint arXiv:2311.02198*, 2023.

[47] A. Z. Ren, J. Lidard, L. L. Ankile, A. Simeonov, P. Agrawal, A. Majumdar, B. Burchfiel, H. Dai, and M. Simchowitz, "Diffusion policy policy optimization," *arXiv preprint arXiv:2409.00588*, 2024.

[48] D. McAllister, S. Ge, B. Yi, C. M. Kim, E. Weber, H. Choi, H. Feng, and A. Kanazawa, "Flow matching policy gradients," *arXiv:2507.21053*, 2025.

[49] C. Lu, H. Chen, J. Chen, H. Su, C. Li, and J. Zhu, "Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2023.

[50] A. Wagenmaker, M. Nakamoto, Y. Zhang, S. Park, W. Yagoub, A. Nagabandi, A. Gupta, and S. Levine, "Steering your diffusion policy with latent space reinforcement learning," *arXiv preprint arXiv:2506.15799*, 2025.

[51] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations," Jun. 2018, arXiv:1709.10087 [cs]. [Online]. Available: http://arxiv.org/abs/1709.10087

[52] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," 2018.

[53] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[54] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband *et al.*, "Deep q-learning from demonstrations," in *Proceedings of the AAAI conference on artificial intelligence*, 2018.

[55] N. Ashvin, D. Murtaza, G. Abhishek, and L. Sergey, "Accelerating online reinforcement learning with offline datasets," *CoRR, vol. abs/2006.09359*, 2020.

[56] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine, "Efficient online reinforcement learning with offline data," in *International Conference on Machine Learning*. PMLR, 2023, pp. 1577–1594.

[57] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on robot learning*, 2018.

[58] J. Luo, Z. Hu, C. Xu, Y. L. Tan, J. Berg, A. Sharma, S. Schaal, C. Finn, A. Gupta, and S. Levine, "SERL: A Software Suite for Sample-Efficient Robotic Reinforcement Learning," Jan. 2024, arXiv:2401.16013 [cs]. [Online]. Available: http://arxiv.org/abs/2401.16013

[59] J. Luo, C. Xu, J. Wu, and S. Levine, "Precise and Dexterous Robotic Manipulation via Human-in-the-Loop Reinforcement Learning," Nov. 2024, arXiv:2410.21845 [cs]. [Online]. Available: http://arxiv.org/abs/2410.21845

[60] Y. Seo, J. Uruç, and S. James, "Continuous control with coarse-to-fine reinforcement learning," *arXiv preprint arXiv:2407.07787*, 2024.

[61] J. Yang, M. S. Mark, B. Vu, A. Sharma, J. Bohg, and C. Finn, "Robot Fine-Tuning Made Easy: Pre-Training Rewards and Policies for Autonomous Real-World Reinforcement Learning," Oct. 2023, arXiv:2310.15145 [cs]. [Online]. Available: http://arxiv.org/abs/2310.15145

[62] P. Dong, S. Mirchandani, D. Sadigh, and C. Finn, "What Matters for Batch Online Reinforcement Learning in Robotics?" May 2025, arXiv:2505.08078 [cs]. [Online]. Available: http://arxiv.org/abs/2505.08078

[63] Z. Chen, Q. Yan, Y. Chen, T. Wu, J. Zhang, Z. Ding, J. Li, Y. Yang, and H. Dong, "Clutterdexgrasp: A sim-to-real system for general dexterous grasping in cluttered scenes," *arXiv preprint arXiv:2506.14317*, 2025.

[64] L. Lai, A. Z. Huang, and S. J. Gershman, "Action chunking as policy compression," *PsyArXiv*, 2022.

[65] L. Ankile, A. Simeonov, I. Shenfeld, and P. Agrawal, "Juicer: Data-efficient imitation learning for robotic assembly," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 5096–5103.

[66] R. Bellman, "A markovian decision process," *Journal of mathematics and mechanics*, pp. 679–684, 1957.

[67] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[68] X. Chen, C. Wang, Z. Zhou, and K. Ross, "Randomized ensembled double q-learning: Learning fast without a model," 2021. [Online]. Available: https://arxiv.org/abs/2101.05982

[69] S. Park, K. Frans, D. Mann, B. Eysenbach, A. Kumar, and S. Levine, "Horizon reduction makes rl scalable," *arXiv preprint arXiv:2506.04168*, 2025.

[70] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, 2018.

[71] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016. [Online]. Available: https://arxiv.org/abs/1607.06450

[72] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[73] I. Kostrikov, D. Yarats, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," 2021. [Online]. Available: https://arxiv.org/abs/2004.13649

[74] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," *arXiv preprint arXiv:2108.03298*, 2021.

[75] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, Y. Zhu, and K. Lin, "robosuite: A modular simulation framework and benchmark for robot learning," in *arXiv preprint arXiv:2009.12293*, 2020.

[76] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *International Conference on Intelligent Robots and Systems*, 2012.

[77] J. Peters and S. Schaal, "Reinforcement learning by reward-weighted regression for operational space control," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 745–750.

[78] J. Oh, Y. Guo, S. Singh, and H. Lee, "Self-imitation learning," in *International conference on machine learning*. PMLR, 2018, pp. 3878–3887.

[79] M. Riedmiller, J. T. Springenberg, R. Hafner, and N. Heess, "Collect & Infer – a fresh look at data-efficient Reinforcement Learning," Aug. 2021, arXiv:2108.10273 [cs]. [Online]. Available: http://arxiv.org/abs/2108.10273

[80] T. Lampe, A. Abdolmaleki, S. Bechtle, S. H. Huang, J. T. Springenberg, M. Bloesch, O. Groth, R. Hafner, T. Hertweck, M. Neunert, M. Wulfmeier, J. Zhang, F. Nori, N. Heess, and M. Riedmiller, "Mastering Stacking of Diverse Shapes with Large-Scale Iterative Reinforcement Learning on Real Robots," Dec. 2023, arXiv:2312.11374 [cs]. [Online]. Available: http://arxiv.org/abs/2312.11374

[81] J. Wang, Y. Yuan, H. Che, H. Qi, Y. Ma, J. Malik, and X. Wang, "Lessons from learning to spin "pens"," 2024. [Online]. Available: https://arxiv.org/abs/2407.18902

[82] H. Kress-Gazit, K. Hashimoto, N. Kuppuswamy, P. Shah, P. Horgan, G. Richardson, S. Feng, and B. Burchfiel, "Robot learning as an empirical science: Best practices for policy evaluation," 2024. [Online]. Available: https://arxiv.org/abs/2409.09491

[83] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.