RESIDUAL OFF-POLICY RL FOR FINETUNING BEHAVIOR CLONING POLICIES

Lars Ankile[1,2,*] **Zhenyu Jiang**[1]
Rocky Duan[1] **Guanya Shi**[1,3,†] **Pieter Abbeel**[1,4,†] **Anusha Nagabandi**[1]
[1]Amazon FAR [2]Stanford University [3]Carnegie Mellon University [4]UC Berkeley

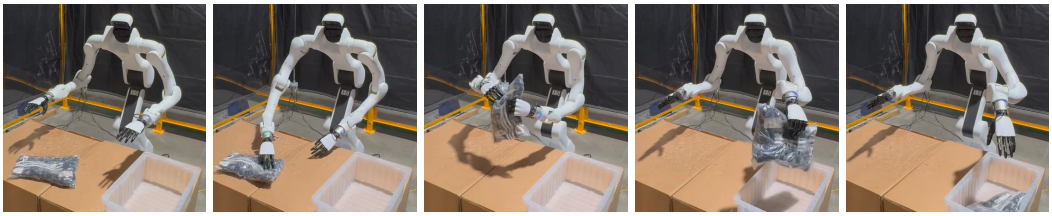

Figure 1. Our residual RL method (ResFiT), performing real-world RL directly on a 29-degree-of-freedom (DoF) wheeled humanoid platform with two 5-fingered hands, is shown here performing the task of bimanual package handover.

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 offline 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 sample-efficient off-policy RL. We demonstrate that our method requires only sparse binary reward signals and can effectively improve manipulation policies on high-degree-of-freedom (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 demonstrate state-of-the-art performance in various vision-based tasks, pointing towards a practical pathway for deploying RL in the real world.

Project website: residual-offpolicy-rl.github.io.

1 INTRODUCTION

Enabling robots to learn and improve directly in their deployment environments remains a fundamental challenge in robotics. Recently, significant progress has been made in training visuomotor control policies in the real world with behavior cloning (BC) from human demonstrations (Brohan & et.al., 2022; Bousmalis et al., 2023; Brohan & et.al., 2023; Chi et al., 2023; Zhao et al., 2023; 2024; Black et al., 2024; Bjorck et al., 2025; Barreiros et al., 2025). However, this success requires

[*]Corresponding author: ankile@stanford.edu; work done while an intern at Amazon FAR
[†]Work done while at Amazon FAR

significant infrastructure, as well as numerous hours of manual and cumbersome 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 (Ross & Bagnell, 2010; Zhao et al., 2024; Yu et al., 2024; Ankile et al., 2024b; Jiang et al., 2025).

Reinforcement learning (RL) offers a complementary paradigm: agents learn autonomously through trial and error in the environment. Deep RL has shown great success in various domains (Mnih et al., 2015; Silver et al., 2016; Farooq & Iqbal, 2024; Mirhoseini et al., 2021; Guo et al., 2025; Yu et al., 2025; Wang et al., 2024c; Torne et al., 2024b), including in-hand manipulation (Chen et al., 2022; Qi et al., 2023) and locomotion (Radosavovic et al., 2024; Gu et al., 2024; Margolis et al., 2024; Hwangbo et al., 2019). However, strong RL performance generally requires large amounts of data from online interactions, so its application has been mainly in simulation (Zhao et al., 2020; Da et al., 2025) since real-world data are expensive and potentially unsafe to gather in large amounts.

A natural direction to improve BC policies is to leverage online RL (Ankile et al., 2024b; Mark et al., 2024; Dong et al., 2025a; Yuan et al., 2024; Tan et al., 2025), combining the strengths of each: BC policies provide a strong prior that can regularize exploration in the RL process, while online RL enhances policy performance by learning from interactions with the environment. However, modern BC architectures are typically deep models with tens of millions to billions of parameters that utilize action chunking or diffusion-based approaches, which can make it challenging to apply RL methods directly to optimize the policy. A simple yet powerful recipe that avoids several of the above issues is residual RL (Silver et al., 2018; Johannink et al., 2019; Alakuijala et al., 2021; Haldar et al., 2023; Ankile et al., 2024b; Yuan et al., 2024; Dodeja et al., 2025; Dong et al., 2025a), 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 demonstrated that residual RL can indeed enhance the reliability of a pretrained policy. Still, it has so far been limited to learning in simulation (Ankile et al., 2024b; Yuan et al., 2024; Dodeja et al., 2025; Dong et al., 2025a) or demonstrating results in simple or constrained settings (Silver et al., 2018; Johannink et al., 2019; Alakuijala et al., 2021; Haldar et al., 2023); Applications to high-DoF systems learning directly in the real world are still lacking.

In this work, we present an off-policy *Res*idual *Fi*ne*T*uning (ResFiT) approach that utilizes online RL to enhance BC policies. By treating the base policy as a black box and learning a per-step residual correction that is independent of chunk size and policy parameterization, we sidestep the challenges of directly optimizing huge base policies. By carefully designing our off-policy recipe, we make the RL process sample-efficient enough to scale to high-DoF bimanual systems, require only sparse binary reward signals, and be safe enough for real-world deployment. We demonstrate robust performance on sparse-reward, long-horizon, vision-based tasks, achieving state-of-the-art results across a range of simulation tasks. We also investigate each design decision in 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.

2 RELATED WORK

Behavior Cloning. BC has shown success in various settings, from autoregressive next-token prediction (Brohan & et.al., 2022; Radosavovic et al., 2024; Cui et al., 2022; Kim et al., 2024) and diffusion policies (Chi et al., 2023; Team et al., 2024; Zhao et al., 2024; Peebles & Xie, 2023; Black et al., 2024), to large transformers that directly output robot actions (Zhao et al., 2023; Kim et al., 2025; Wang et al., 2024b). Recent advances (Intelligence et al., 2025; Bjorck et al., 2025; Barreiros et al., 2025) 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 incredibly high, and recent work has shown diminishing returns and performance plateaus with increasing data (Zhao et al., 2024; Jiang et al., 2025; Ankile et al., 2024b).

Finetuning BC Policies with RL. RL finetuning has been explored as a method to enhance BC policies by allowing the agent to interact directly with the environment. However, directly finetuning modern BC architectures with RL presents significant challenges. State-of-the-art BC policies typically use action-chunking or diffusion-based approaches (Zhao et al., 2024; Chi et al., 2023) with large neural networks, which can lead to unstable learning when combined with conventional

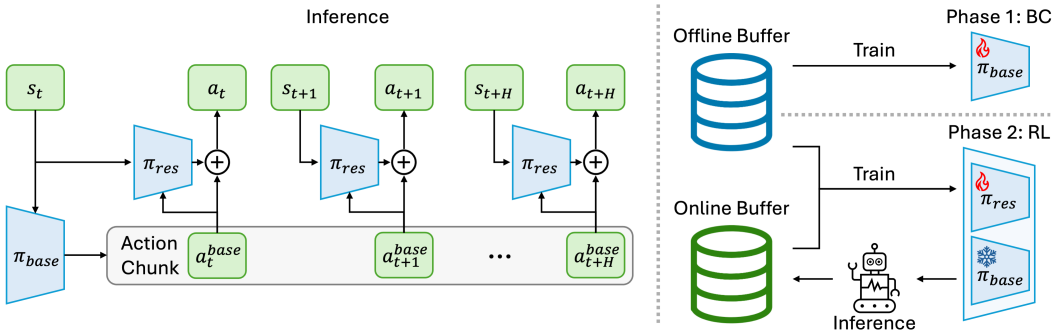

Figure 2. Off-policy residual finetuning (ResFiT): A two-phase approach using online RL to improve BC policies. First, we train a base policy using BC on a dataset of demonstrations and then freeze the base. Then, we learn a residual policy with RL to correct the base actions. The RL phase utilizes our off-policy recipe, leveraging both demonstrations and interactions, and enables more stable and safe exploration in the real world, as we can directly control the magnitude of the residuals. This residual approach is agnostic to the base policy parameterization and can be applied to large action-chunked BC policies to provide closed-loop corrections.

RL methods. Several works address these challenges (Chen et al., 2025a; Hansen-Estruch et al., 2023). IBRL (Hu et al., 2023) trains an imitation policy and then uses it to propose actions for exploration and to bootstrap target values. PA-RL (Mark et al., 2024) sidesteps applying RL to complex model architectures by learning a Q-function to optimize actions instead of the policy. Other approaches (Ren et al., 2024; McAllister et al., 2025; Lu et al., 2023) specifically adapt diffusion- or flow-based policies; e.g., DSRL (Wagenmaker et al., 2025) runs RL over the latent-noise space of a frozen diffusion policy. These methods generally couple algorithm design with the BC policy structure, limiting flexibility across policy classes.

Residual RL and Sample Efficiency. Residual RL (Silver et al., 2018; Johannink et al., 2019; Alakuijala et al., 2021; Haldar et al., 2023) 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, such as ResiP (Ankile et al., 2024b), combines an action-chunked BC policy with a single-step RL residual policy; however, its on-policy RL algorithm has prohibitively high sample complexity for real-world deployment.

Toward the goal of improved sample efficiency, prior work shows promising results through learning from demonstrations (Rajeswaran et al., 2018) and off-policy RL (Haarnoja et al., 2018; Fujimoto et al., 2018); drawing from a host of these related works (Hester et al., 2018; Ashvin et al., 2020; Ball et al., 2023), 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 to regularize the online phase.

Closest to our work, Policy Decorator (Yuan et al., 2024) and EXPO (Dong et al., 2025a) demonstrate that off-policy RL can be used to train residual policies more sample-efficiently. However, Policy Decorator learns residuals over entire action chunks rather than per-step corrections, while EXPO uses base policies without chunking. Concurrently, Dodeja et al. (2025) proposes using uncertainty estimates to guide exploration in residual RL. In all cases, RL training is performed in simulation on single-arm tasks, specifically state-based tasks for EXPO. In contrast, we train visuomotor policies directly in the real world on tasks with higher degrees of freedom and longer horizons, demonstrating RL on a bimanual humanoid robot with two 5-fingered hands. Concurrent work has finetuned a generalist policy with real-world residual RL (Xiao et al., 2025).

Real-World RL. In the category of real-world RL, QT-Opt(Kalashnikov et al., 2018) utilized data from a fleet of 7 robots operating over several weeks to train a robotic arm with a parallel jaw gripper to grasp diverse objects. More recently, SERL (Luo et al., 2024a) demonstrated insertion and pick-and-place policies by combining a few demonstrations with less than an hour of online interaction. A follow-up work(Luo et al., 2024b) 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 (Seo et al., 2024; Hu et al., 2023; Wagenmaker et al., 2025; Yang et al., 2023), including Dong et al. (2025b), who showed that using Q-functions to guide batch online RL is more performant than imitation-based methods.

On higher-DOF robots for locomotion (Radosavovic et al., 2024; Gu et al., 2024) and manipulation (Jiang et al., 2025; Chen et al., 2025b), people often learn in simulation and rely on sim-to-real transfer to run in the real world. For manipulation, CASHER (Torne et al., 2024a) trains policies in crowdsourced digital twins, achieving zero-shot transfer with sublinear scaling of human effort. However, it is limited to simple manipulation due to challenges in contact simulation.

3 METHOD

Our off-policy RL method for *Res*idual *Fi*ne*T*uning (ResFiT) of BC policies is illustrated in Figure 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 chose 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 sections, we outline the details of the method, including the design choices and implementation details that are crucial to its success.

3.1 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 $\pi_\psi(a_{t:t+k}|o_t)$ using 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, rather than a single action, is known to improve performance (Lai et al., 2022; Zhao et al., 2023; Chi et al., 2023; Ankile et al., 2024a) 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 actual underlying system state s_t, the remainder of this section will use the state notation for simplicity.

3.2 FINE-TUNING WITH OFF-POLICY RESIDUAL RL

The above behavior cloning process produces a policy π_{base} that has some level of task success. We freeze the base policy and train a new policy π_{res} with RL to learn how to correct mistakes made by π_{base} and improve policy performance in a way that is (1) agnostic to how π_{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) (Bellman, 1957) with states $s_t \in S$, actions $a_t \in A$, rewards $r_t = R(s_t, a_t)$, discount factor γ, and horizon H. RL aims to learn the parameters θ 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 τ is maximized under the trajectory distribution induced by the policy. In this work, we will learn both a critic Q_ϕ as well as a policy π_θ, as shown in DDPG (Lillicrap et al., 2015).

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 $Q_\phi(s_t, a_t^{\text{base}} + \pi_\theta(s_t, a_t^{\text{base}}))$ (similar to (Yuan et al., 2024; Dodeja et al., 2025; Dong et al., 2025a)) 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. We provide the full algorithm pseudocode and loss derivations in Appendix A.

3.3 DESIGN DECISIONS

In the following, we detail the key design choices that were required to achieve stable residual fine-tuning performance across different tasks and embodiments, and later validate these decisions through ablations.

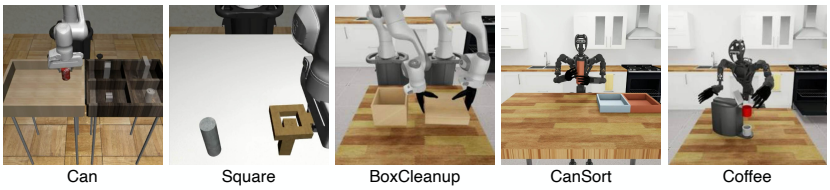

Figure 3. Our simulation tasks from Robomimic and DexMimicGen, spanning single-arm manipulation as well as bimanual coordination tasks.

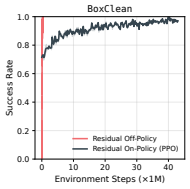

Figure 4. PPO vs. off-policy RL in ResFiT.

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 (Ball et al., 2023; Chen et al., 2021). We also use n-step returns (in particular, our $n = 3$), similar to CQN (Seo et al., 2024) and IBRL (Hu et al., 2023), which is shown to help with long-horizon and sparse-reward tasks (Park et al., 2025). 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: $\sum_{i=0}^{n-1} \gamma^i r + \gamma^n Q(s_{t+n}, a_{t+n})$ (Sutton & Barto, 2018). Note that we omit this detail in Algorithm 1 (see Appendix A) 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 (Ball et al., 2023), we add layer normalization (Ba et al., 2016) to the critic to mitigate catastrophic overestimation 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 (Fujimoto et al., 2018), 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 (Chen et al., 2021) 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 (Dosovitskiy et al., 2020) encoder with DrQ-style (Kostrikov et al., 2021) random shift augmentations to help with overfitting, introduced by Hu et al. (2023). Finally, we use our demonstration data not only to train our base policy but also during the online RL phase via symmetric sampling (Ball et al., 2023), by sampling 50% of each batch from the offline data, and the remaining 50% from the growing online buffer.

4 EXPERIMENTAL SETUP

4.1 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 our approach on tasks from Robomimic (Mandlekar et al., 2021) and DexMimicGen (Jiang et al., 2025), which vary in terms of robots, control modes, degrees of freedom, task horizons, and precision requirements. Our experiments utilize the Robosuite environment (Zhu et al., 2020), which is built on MuJoCo (Todorov et al., 2012), and we operate at a control frequency of 20 Hz.

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

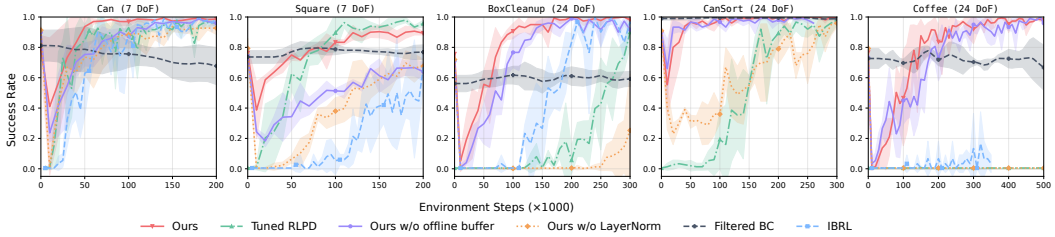

Figure 5. Success rates of different approaches on our simulation tasks, showing ResFiT converging to high-performing policies for all tasks. Note that all approaches here start with the same number of demos.

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 grasping a small 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.

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 (Ball et al., 2023), 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. 3.3) 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 (Hu et al., 2023), 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 (Peters & Schaal, 2007) (with 0/1 weights) and self-imitation (Oh et al., 2018; Riedmiller et al., 2021), which prior work has shown can improve performance over the demo-only case (Bousmalis et al., 2023; Ankile et al., 2024a; Lampe et al., 2023; Wang et al., 2024a).

We also ablate layer norm for actor and critic MLPs, demo inclusion during online RL, the UTD ratio, and n-step returns. Finally, we compare to on-policy residual RL (Ankile et al., 2024b).

4.2 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 bin. In `PackageHandover`, the right hand picks up a deformable package, hands it to the left hand, and places it into a bin 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

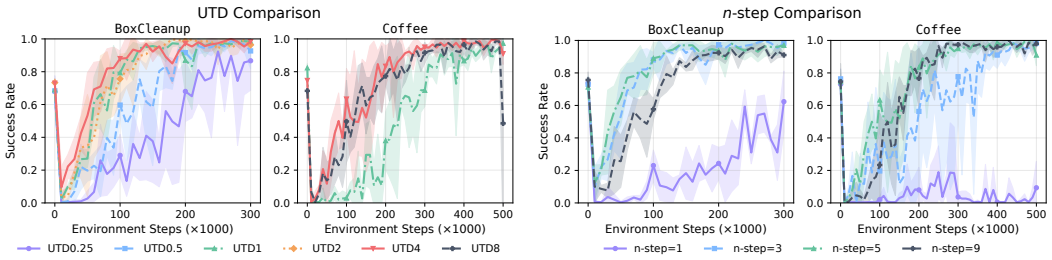

Figure 6. Impact of UTD ratio (left) and `n-step` (right) on the `BoxCleanup` and `Coffee` tasks.

self-collisions. We developed a system for performing RL in the real world, where the operator either lets each episode time out or marks the episode result 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 at the end of successful episodes, 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 (Barreiros et al., 2025; Kress-Gazit et al., 2024). 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 ensures that both policies face nearly identical environmental conditions in each round.

5 RESULTS

5.1 SIMULATION RESULTS

We evaluate various aspects of ResFiT on simulated benchmark tasks, consisting of both single-arm and bimanual manipulation tasks. On the simulated `BoxCleanup` task, we first compare our off-policy residual RL recipe to an existing approach (Ankile et al., 2024b) of performing residual RL in sim with the on-policy PPO (Schulman et al., 2017) algorithm. As shown in Figure 4, our approach converging at 200k steps versus 40M steps, we see a $\sim$200$\times$ 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, achieve high success rates over $\sim$150k steps, whereas ResFiT converges faster at $\sim$75k steps. In the `Square` task, only ResFiT and our optimized off-policy approach ("Tuned RLPD"), which does not use a base policy but incorporates 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 (Ankile et al., 2024a).

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.

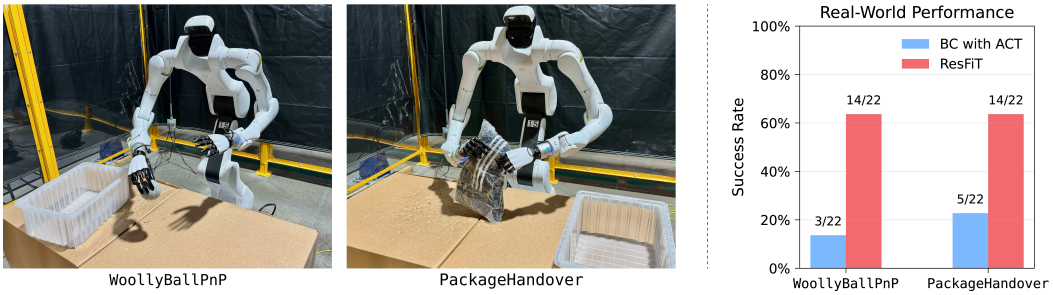

Figure 7. Real-world results of applying ResFiT, where our residual RL approach shows a significant boost in performance over the base model, on a 29-DoF bimanual wheeled humanoid robot with two 5-fingered hands.

5.2 Real-World RL Results

In the real world, we apply our methods to two tasks, `WoollyBallPnP` and `PackageHandover`. We use ACT (Zhao et al., 2024) as the base policy for both tasks, but any BC method would in principle work.

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.

6 Discussion

By learning single-step residual corrections atop a frozen base policy, we resolve a key tension: action chunking improves BC but creates intractably high-dimensional action spaces for RL (870 dims in our case). The base policy also serves a dual purpose beyond initialization — it acts as an implicit safety constraint and provides an exploration prior that enables RL from sparse rewards in high-DOF spaces.

The primary limitation is that learned behaviors remain constrained around the base policy. A promising direction is distilling the improved combined policy back into the base to create room for further residual improvements, particularly in multitask settings. Our real-world results validate this approach on bimanual dexterous platforms, though enabling fully autonomous improvement without human supervision for resets and reward labeling remains an open challenge.

ACKNOWLEDGMENTS

We thank Himanshu Gaurav Singh for technical discussions, helping with the filtered BC baseline, and feedback on the paper draft. Younggyo Seo provided valuable input for real-world implementation and offered constructive feedback on the paper. We also thank Carmelo Sferrazza and Ruihan Yang for their comments on the manuscript. Benjamin Colby, Hassan Farooq, and Sumedh Sontakke helped with real-world experiments. Finally, we appreciate the technical discussions during early project development from Quiang Li, Seohong Park, Perry Dong, Hengyuan Hu, and Suvir Mirchandani.

REFERENCES

Minttu Alakuijala, Gabriel Dulac-Arnold, Julien Mairal, Jean Ponce, and Cordelia Schmid. Residual reinforcement learning from demonstrations. *arXiv:2106.08050*, 2021.

Lars Ankile, Anthony Simeonov, Idan Shenfeld, and Pulkit Agrawal. Juicer: Data-efficient imitation learning for robotic assembly. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5096–5103. IEEE, 2024a.

Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement–residual rl for precise assembly. *arXiv:2407.16677*, 2024b.

Nair Ashvin, Dalal Murtaza, Gupta Abhishek, and L Sergey. Accelerating online reinforcement learning with offline datasets. *CoRR, vol. abs/2006.09359*, 2020.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL `https://arxiv.org/abs/1607.06450`.

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pp. 1577–1594. PMLR, 2023.

Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kunimatsu Hashimoto, Muhammad Zubair Irshad, Masha Itkina, et al. A careful examination of large behavior models for multitask dexterous manipulation. *arXiv preprint arXiv:2507.05331*, 2025.

Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pp. 679–684, 1957.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π0: A vision-language-action flow model for general robot control. *arXiv preprint ARXIV.2410.24164*, 2024.

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X Lee, Maria Bauzá, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. Robocat: A self-improving generalist agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

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

Anthony Brohan and et.al. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control, 2023. arXiv:2307.15818.

Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, Steven Palma, Pepijn Kooijmans, Michel Aractingi, Mustafa Shukor, Dana Aubakirova, Martino Russi, Francesco Capuano, Caroline Pascal, Jade Choghari, Jess Moss, and Thomas Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. `https://github.com/huggingface/lerobot`, 2024.

Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. In *Conference on Robot Learning*, pp. 297–307. PMLR, 2022.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning: Learning fast without a model, 2021. URL `https://arxiv.org/abs/2101.05982`.

Yuxin Chen, Devesh K Jha, Masayoshi Tomizuka, and Diego Romeres. Fdpp: Fine-tune diffusion policy with human preference. *arXiv preprint arXiv:2501.08259*, 2025a.

Zeyuan Chen, Qiyang Yan, Yuanpei Chen, Tianhao Wu, Jiyao Zhang, Zihan Ding, Jinzhou Li, Yaodong Yang, and Hao Dong. Clutterdexgrasp: A sim-to-real system for general dexterous grasping in cluttered scenes. *arXiv preprint arXiv:2506.14317*, 2025b.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, pp. 02783649241273668, 2023.

Zichen Jeff Cui, Yibin Wang, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022.

Longchao Da, Justin Turnau, Thirulogasankar Pranav Kutralingam, Alvaro Velasquez, Paulo Shakarian, and Hua Wei. A survey of sim-to-real methods in rl: Progress, prospects and challenges with foundation models. *arXiv preprint arXiv:2502.13187*, 2025.

Lakshita Dodeja, Karl Schmeckpeper, Shivam Vats, Thomas Weng, Mingxi Jia, George Konidaris, and Stefanie Tellex. Accelerating residual reinforcement learning with uncertainty estimation. *arXiv preprint arXiv:2506.17564*, 2025.

Perry Dong, Qiyang Li, Dorsa Sadigh, and Chelsea Finn. Expo: Stable reinforcement learning with expressive policies. *arXiv preprint arXiv:2507.07986*, 2025a.

Perry Dong, Suvir Mirchandani, Dorsa Sadigh, and Chelsea Finn. What Matters for Batch Online Reinforcement Learning in Robotics?, May 2025b. URL `http://arxiv.org/abs/2505.08078`. arXiv:2505.08078 [cs].

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

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

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Xinyang Gu, Yen-Jen Wang, Xiang Zhu, Chengming Shi, Yanjiang Guo, Yichen Liu, and Jianyu Chen. Advancing humanoid locomotion: Mastering challenging terrains with denoising world model learning. *arXiv preprint arXiv:2408.14472*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, 2018.

Siddhant Haldar, Jyothish Pari, Anant Rai, and Lerrel Pinto. Teach a robot to fish: Versatile imitation from one minute of demonstrations. *arXiv preprint arXiv:2303.01497*, 2023.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. IDQL: Implicit Q-Learning as an Actor-Critic Method with Diffusion Policies, 2023. arXiv:2304.10573.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI conference on artificial intelligence*, 2018.

Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. Imitation bootstrapped reinforcement learning. *arXiv preprint arXiv:2311.02198*, 2023.

Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. pi0.5: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning, 2025.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 international conference on robotics and automation (ICRA)*, 2019.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, 2018.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels, 2021. URL https://arxiv.org/abs/2004.13649.

Hadas Kress-Gazit, Kunimatsu Hashimoto, Naveen Kuppuswamy, Paarth Shah, Phoebe Horgan, Gordon Richardson, Siyuan Feng, and Benjamin Burchfiel. Robot learning as an empirical science: Best practices for policy evaluation, 2024. URL https://arxiv.org/abs/2409.09491.

Lucy Lai, Ann Zixiang Huang, and Samuel J Gershman. Action chunking as policy compression. *PsyArXiv*, 2022.

Thomas Lampe, Abbas Abdolmaleki, Sarah Bechtle, Sandy H. Huang, Jost Tobias Springenberg, Michael Bloesch, Oliver Groth, Roland Hafner, Tim Hertweck, Michael Neunert, Markus Wulfmeier, Jingwei Zhang, Francesco Nori, Nicolas Heess, and Martin Riedmiller. Mastering Stacking of Diverse Shapes with Large-Scale Iterative Reinforcement Learning on Real Robots, December 2023. URL http://arxiv.org/abs/2312.11374. arXiv:2312.11374 [cs].

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2023.

Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. SERL: A Software Suite for Sample-Efficient Robotic Reinforcement Learning, January 2024a. URL http://arxiv.org/abs/2401.16013. arXiv:2401.16013 [cs].

Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and Dexterous Robotic Manipulation via Human-in-the-Loop Reinforcement Learning, November 2024b. URL http://arxiv.org/abs/2410.21845. arXiv:2410.21845 [cs].

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. *The International Journal of Robotics Research*, 43(4):572–587, 2024.

Max Sobol Mark, Tian Gao, Georgia Gabriela Sampaio, Mohan Kumar Srirama, Archit Sharma, Chelsea Finn, and Aviral Kumar. Policy Agnostic RL: Offline RL and Online RL Fine-Tuning of Any Class and Backbone, December 2024. URL http://arxiv.org/abs/2412.06685. arXiv:2412.06685 [cs].

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. *arXiv:2507.21053*, 2025.

Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nova, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 2015.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International conference on machine learning*, pp. 3878–3887. PMLR, 2018.

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes rl scalable. *arXiv preprint arXiv:2506.04168*, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF international conference on computer vision*, 2023.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, 2007.

Haozhi Qi, Brent Yi, Sudharshan Suresh, Mike Lambeta, Yi Ma, Roberto Calandra, and Jitendra Malik. General in-hand object rotation with vision and touch. In *Conference on Robot Learning*, pp. 2549–2564. PMLR, 2023.

Ilija Radosavovic, Tete Xiao, Bike Zhang, Trevor Darrell, Jitendra Malik, and Koushil Sreenath. Real-world humanoid locomotion with reinforcement learning. *Science Robotics*, 9(89):eadi9579, 2024.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations, June 2018. URL http://arxiv.org/abs/1709.10087. arXiv:1709.10087 [cs].

Allen Z Ren, Justin Lidard, Lars L Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. *arXiv preprint arXiv:2409.00588*, 2024.

Martin Riedmiller, Jost Tobias Springenberg, Roland Hafner, and Nicolas Heess. Collect & Infer – a fresh look at data-efficient Reinforcement Learning, August 2021. URL http://arxiv.org/abs/2108.10273. arXiv:2108.10273 [cs].

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 661–668. JMLR Workshop and Conference Proceedings, 2010. URL https://proceedings.mlr.press/v9/ross10a.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Younggyo Seo, Jafar Uruç, and Stephen James. Continuous control with coarse-to-fine reinforcement learning. *arXiv preprint arXiv:2407.07787*, 2024.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2018.

Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive post-training for vision-language-action models. *arXiv preprint arXiv:2505.17016*, 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*, 2012.

Marcel Torne, Arhan Jain, Jiayi Yuan, Vidaaranya Macha, Lars Ankile, Anthony Simeonov, Pulkit Agrawal, and Abhishek Gupta. Robot learning with super-linear scaling. *arXiv preprint arXiv:2412.01770*, 2024a.

Marcel Torne, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. Reconciling Reality through Simulation: A Real-to-Sim-to-Real Approach for Robust Manipulation, 2024b. arXiv:2403.03949.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.

Jun Wang, Ying Yuan, Haichuan Che, Haozhi Qi, Yi Ma, Jitendra Malik, and Xiaolong Wang. Lessons from learning to spin "pens", 2024a. URL https://arxiv.org/abs/2407.18902.

Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. *Advances in neural information processing systems*, 37:124420–124450, 2024b.

Shuhe Wang, Shengyu Zhang, Jie Zhang, Runyi Hu, Xiaoya Li, Tianwei Zhang, Jiwei Li, Fei Wu, Guoyin Wang, and Eduard Hovy. Reinforcement learning enhanced llms: A survey. *arXiv preprint arXiv:2412.10400*, 2024c.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Yuqi Xie, Fengyuan Hu, Jimmy Wu, Zhengyi Luo, Linxi Fan, et al. Self-improving vision-language-action models with data generation via residual rl. *arXiv preprint arXiv:2511.00091*, 2025.

Jingyun Yang, Max Sobol Mark, Brandon Vu, Archit Sharma, Jeannette Bohg, and Chelsea Finn. Robot Fine-Tuning Made Easy: Pre-Training Rewards and Policies for Autonomous Real-World Reinforcement Learning, October 2023. URL http://arxiv.org/abs/2310.15145. arXiv:2310.15145 [cs].

Alan Yu, Ge Yang, Ran Choi, Yajvan Ravan, John Leonard, and Phillip Isola. Learning visual parkour from generated images. In *8th Conference on Robot Learning*, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Mengke Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. *arXiv preprint arXiv:2412.13630*, 2024.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Tony Z Zhao, Jonathan Tompson, Danny Driess, Pete Florence, Kamyar Ghasemipour, Chelsea Finn, and Ayzaan Wahid. Aloha unleashed: A simple recipe for robot dexterity. *arXiv preprint arXiv:2410.13126*, 2024.

Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE symposium series on computational intelligence (SSCI)*, pp. 737–744. IEEE, 2020.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, Yifeng Zhu, and Kevin Lin. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.

A ALGORITHM DETAILS

We present the full pseudocode for ResFiT in Algorithm 1 and derive the critic and actor losses below.

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

Input: pre-trained base π_{b}, dataset of demos $\mathcal{D}_{\text{offline}}$
Initialize: residual policy π_θ, Q ensemble $Q_{\phi_1}, \dots Q_{\phi_N}$, vision encoder f_ω, empty buffer $\mathcal{D}_{\text{online}}$
Initialize: set target networks $\theta' \leftarrow \theta$, $\phi'_i \leftarrow \phi_i$
for step $= 1, 2, \dots,$ `buffer_warmup_steps` **do**
  Sample noise $\epsilon_t \sim \mathcal{U}(-\text{noise\_scale}, \text{noise\_scale})$
  Step env with $a_t = \epsilon_t + a^{\text{b}}_t$ where $a^{\text{b}}_t \sim \pi_{\text{b}}(s_t)$
  Observe next state s_{t+1}, reward r_t, done flag d_t
  Add transition $(s_t, a^{\text{b}}_t, a_t, s_{t+1}, a^{b}_{t+1}, r_t, d_t)$ to $\mathcal{D}_{\text{online}}$
end for
repeat
  $a^{\text{b}}_t \sim \pi_{\text{b}}(s_t)$ and $a^{\text{r}}_t \sim \pi_\theta(s_t, a^{\text{b}}_t)$
  Step env with $a_t = a^{\text{b}}_t + a^{\text{r}}_t$
  Add $(s_t, a^{\text{b}}_t, a_t, s_{t+1}, a^{b}_{t+1}, r_t, d_t)$ to $\mathcal{D}_{\text{online}}$
  Repeat UTD times:
    Sample batch B evenly from $\mathcal{D}_{\text{offline}}$, $\mathcal{D}_{\text{online}}$
    Update critic using B:
      $a^{\text{r}}_{t+1} \sim \pi_{\theta'}(s_{t+1}, a^{\text{b}}_{t+1})$
      Compute $a_{t+1} = a^{\text{b}}_{t+1} + a^{\text{r}}_{t+1}$
      $y = r_t + (1 - d_t) * \gamma * \min_{\text{subset(i)}} Q_{\phi'_i}(s_{t+1}, a_{t+1})$
      Update ϕ_i, ω to minimize $\text{MSE}(Q_{\phi_i}(s_t, a_t), y)$
      Update critic targets $\phi'_i \leftarrow \rho\phi'_i + (1 - \rho)\phi_i$
  Update actor using latest B:
    Update θ to maximize:
      $\frac{1}{N} \sum_{i=1}^{N} Q_{\phi_i}(s_t, \pi_\theta(s_t, a^{\text{b}}_t) + a^{\text{b}}_t)$
    Update actor target $\theta' \leftarrow \rho\theta' + (1 - \rho)\theta$
until convergence

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

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

We train our value approximator Q_ϕ 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:

$$L(\phi) = \mathop{\text{E}}_{(s_t, a_t, r_t, s_{t+1}, d_t) \sim \mathcal{D}}\Bigg[\Bigg(Q_\phi(s_t, a_t) - \big(r_t + \gamma(1 - d) Q_\phi(s_{t+1}, a^{\text{base}}_{t+1} + \pi_\theta(s_{t+1}, a^{\text{base}}_{t+1}))\big)\Bigg)^2\Bigg] \tag{A.2}$$

where $a^{\text{base}}_{t+1} = \pi_{\text{base}}(s_{t+1})$. 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 $\pi_\theta(s)$ by simply performing gradient ascent on the value function, with respect to the policy

parameters:

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

B IMPLEMENTATION DETAILS

B.1 BC HYPERPARAMETERS

We list the hyperparameters used for the training of the BC policies we used as the base policies in subsequent experiments in Table B.1. In these experiments, we use the Hugging Face LeRobot (Cadene et al., 2024) implementation of the Action-Chunking Transformer (ACT) (Zhao et al., 2023) as a stand-in for any action-chunked large BC model.

Table B.1. ACT BC Training Hyperparameters for Different Tasks

Hyperparameter	`Can`	`Square`	`BoxCleanup`	`CanSort`	`Coffee`
			Training Configuration		
Total Steps	50,000	50,000	50,000	100,000	200,000
Batch Size			256		
Learning Rate			1e-4		
Weight Decay			1e-4		
Backbone LR			1e-5		
Gradient Clip Norm			10.0		
			Policy Architecture (ACT)		
Chunk Size			20		
Action Steps			20		
Observation Steps			1		
Vision Backbone			ResNet-18		
Pretrained Weights			ImageNet		
Model Dimension			512		
Attention Heads			8		
Feedforward Dim			3200		
Encoder Layers			4		
Decoder Layers			1		
VAE Enabled			False		
Dropout			0.1		
			Evaluation Configuration		
Rollout Frequency	1,000	1,000	1,000	1,000	5,000
Num Parallel Envs			16		
Episodes per Eval			100		
Camera Size			84		
Render Size			84		
Policy Image Keys	agentview robot0_eye_in_hand	agentview robot0_eye_in_hand	agentview robot0_eye_in_hand robot1_eye_in_hand	frontview robot0_eye_in_left_hand robot0_eye_in_right_hand	agentview robot0_eye_in_left_hand robot0_eye_in_right_hand
Checkpoint selection			Best		

The datasets are standard datasets provided by Robomimic (Mandlekar et al., 2021) and DexMimicGen (Jiang et al., 2025), hosted here and here, respectively.

Before we start BC training, we convert the datasets into the `LeRobotDataset` format. The resulting converted datasets can be found here:

- `https://huggingface.co/datasets/ankile/robomimic-mh-can-image`
- `https://huggingface.co/datasets/ankile/robomimic-mh-square-image`
- `https://huggingface.co/datasets/ankile/dexmg-two-arm-box-cleanup`
- `https://huggingface.co/datasets/ankile/dexmg-two-arm-can-sort-random`
- `https://huggingface.co/datasets/ankile/dexmg-two-arm-coffee`

For all datasets, we use only the RGB observations and the robot state, and render the camera observations at a resolution of 84 by 84. For all environments except `CanSort`, we use the `agentview` as the third-person camera and `frontview` for `CanSort`. We use wrist cameras for all robots.

We use all the LeRobot default hyperparameters for the ACT model, except that we set the chunk size equal to the control frequency of 20 to have each chunk span 1 second of real-time, as is most standard implementations (Zhao et al., 2023; Black et al., 2024).

To select the checkpoint to use, we typically choose the one with the highest evaluation success rate during training.

B.2 Residual RL Hyperparameters

We list detailed hyperparameters for the main off-policy RL Method, ResFiT, presented in this paper in Table B.1.

Table B.2. Hyperparameters for Residual RL Training Across Tasks

Hyperparameter	`Can`	`Square`	`BoxCleanup`	`CanSort`	`Coffee`
Training Configuration					
Total Timesteps	200,000	200,000	300,000	300,000	500,000
Batch Size			256		
Buffer Size	200,000	200,000	300,000	300,000	300,000
Learning Starts (steps)			10,000		
Offline Episodes	300	300	1,000	1,000	1,000
Demo source	Multi-Human	Multi-Human	DexMimicGen	DexMimicGen	DexMimicGen
Offline Fraction			0.5		
Episode timeout (steps)	200	300	300	300	400
Algorithm Parameters					
Discount Factor (γ)	0.99	0.99	0.99	0.995	0.995
N-step Returns	3	3	3	5	5
Updates per Iteration			4		
Actor Updates per Iteration			1		
Update Every N Steps			1		
Critic Warmup Steps			10,000		
Learning Rates & Optimization					
Actor Learning Rate	1×10^{-6}	1×10^{-6}	5×10^{-6}	1×10^{-6}	1×10^{-6}
Critic Learning Rate			1×10^{-4}		
Critic Target Tau			0.005		
Actor LR Warmup Steps			0		
Action & Exploration					
Control Frequency (Hz)			20		
Action Scale	0.1	0.1	0.2	0.1	0.2
Stddev Max			0.05		
Stddev Min			0.05		
Stddev Clip			0.3		
Random Action Noise Scale			0.2		
Use Base Policy for Warmup			True		
Network Architecture					
Actor Last Layer Init Scale			0.0		
Actor Hidden Dim			1024		
Actor Feature Dim			128		
Encoder Type			ViT		
Use Layer Norm			True		

More explanation of some of the above hyperparameter terms:

- *Batch Size:* The batch size is the total batch size after combining the sampled data from the offline and online buffers.
- *Learning Starts:* Before we do any model updates, we roll out the policy with random action noise in the environment for this number of steps, which is standard practice in off-policy RL.
- *Updates per Iteration:* This is essentially the Update-to-Data (UTD) ratio of the RL procedure. For the sim experiments, we take a single step in the environment and then perform this number of critic model updates.

- *Updates per Iteration:* Similar to prior work (Chen et al., 2021; Ball et al., 2023; Hu et al., 2023), we update the actor only one time for every environment step, regardless of how many critic updates we take.
- *Critic Warmup Steps:* Because we start with a pre-trained base policy, the combined policy that the Q-function is supposed to estimate the expected returns for is not entirely randomly initialized, and as such, we run a number of TD-backups according to equation Eq. (A.2) so that the Q-function will be initialized to represent the starting policy better.

B.3 TUNED RLPD HYPERPARAMETERS

Table B.3. Key Parameter Differences: RLPD vs Residual RL

Parameter Category	**Residual RL**	**RLPD**
Core Algorithm Differences		
Base Policy	Required (ACT policy)	None (direct RL)
Action Space	Residual actions	Direct actions
Observation Space	Includes base_action	Standard observations
Learning Rate Differences		
Actor Learning Rate	1×10^{-6} (Can/Square/CanSort/Coffee) 5×10^{-6} (BoxCleanup)	1×10^{-4} (all tasks)
Critic Target Tau	0.005	0.01
Training Differences		
Critic Warmup Steps	10,000	0
Actor Init Scale	0.0 (zero initialization)	None (default init)
Action Scale	0.1-0.2 (task dependent)	1.0 (all tasks)
Exploration Differences		
Stddev (Can/Square/Box)	0.05	0.1
Stddev (CanSort/Coffee)	0.025	0.1
Random Noise Scale	0.2	1.0
Use Base Policy Warmup	True	N/A

B.4 REAL-WORLD RL SYSTEM

Here, we describe the detailed setup of our real-world experiments. We build a simple system for conducting residual RL fine-tuning in physical environments. The policy execution is controlled via a keyboard interface, which allows the user to:

- **Roll out** the residual policy with exploration noise.
- **Reset** the robot to its initial pose, with small random perturbations added to improve policy robustness.
- Mark the episode as a **success**, which terminates the episode and assigns the last transition $reward = 1, done = 1$.
- Mark the episode as a **failure**, which terminates the episode and assigns the last transition $reward = 0, done = 1$.

We also impose a maximum time limit based on the average episode length observed in the demonstration data for each task. If an episode times out, all transitions within it are assigned $reward = 0, done = 0$.

Thanks to the residual formulation, the real-world RL policy does not begin with purely random exploration but instead generally follows a reasonable trajectory toward task completion. However, the residual policy and/or the exploration noise may occasionally drive the robot into collisions. To prevent hardware damage during RL exploration, we use the force-torque sensor equipped on the Dexmate robot and implement a force–torque-based termination mechanism. If the measured

force or torque magnitude exceeds a manually determined threshold, the episode is immediately terminated and the final transition is marked with $reward = 0, done = 1$.