2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

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], and Anusha Nagabandi[1]

[1]Amazon FAR (Frontier AI & Robotics) [2]Stanford University [3]Carnegie Mellon University [4]UC Berkeley

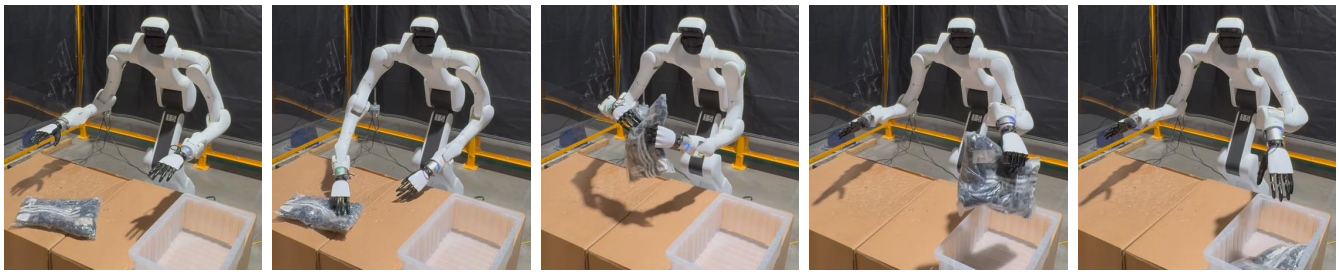

Fig. 1. Our residual RL method (ResFiT), performing real-world RL directly on our 29-degree-of-freedom (DoF) wheeled humanoid platform with two 5-fingered hands, is shown here performing the task of bimanual package handover.

***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 and safety concerns. These challenges are compounded for high-degree-of-freedom (DoF) systems that must learn from sparse rewards over long horizons.**

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.

I. 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 [1]–[9]. However, this success requires 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 [6], [10]–[13].

[*]Work done while an intern at Amazon FAR
[†]Work done while at Amazon FAR

Reinforcement learning (RL) offers a complementary paradigm where agents learn autonomously through trial and error. Deep RL has achieved impressive results across many domains [14]–[21], including manipulation [22], [23] and locomotion [24]–[27], but its high sample complexity has largely confined it to simulation [28], [29].

A natural direction is to leverage online RL to improve BC policies [12], [30]–[33], combining the strengths of each: BC policies provide a strong prior that can regularize exploration, while online RL enhances performance through environment interaction. However, modern BC architectures are typically deep models with tens of millions to billions of parameters that utilize action chunking or diffusion-based approaches [4], [6], making direct RL optimization challenging. Residual RL [12], [31], [32], [34]–[38] sidesteps this by learning only corrective terms atop a fixed base controller. However, it has so far been limited to simulation [12], [31], [32], [38] or simple settings [34]–[37]; applications to high-DoF systems in the real world are still lacking.

In this work, we present off-policy **Res**idual **Fi**ne**T**uning (ResFiT), a recipe that, for the first time, enables residual RL to scale to high-DoF, vision-based, sparse-reward tasks in the real world. By treating the base policy as a frozen black box and learning per-step residual corrections, we sidestep the challenges of directly optimizing large action-chunked policies. Our key contributions are: (1) an off-policy residual RL recipe combining a frozen action-chunked base policy for exploration and safety, per-step residual corrections, high update-to-data ratio, n-step returns, layer normalization, symmetric demo sampling, and visual encoder with augmentation—achieving sample efficiency sufficient for real-world deployment; (2) state-of-the-art results on vision-based simulation benchmarks with sparse rewards, with extensive ablations validating each design decision; and (3) to our knowledge, the first demonstration of RL on a humanoid robot with dexterous five-fingered hands, trained entirely in the real world using only sparse binary rewards.

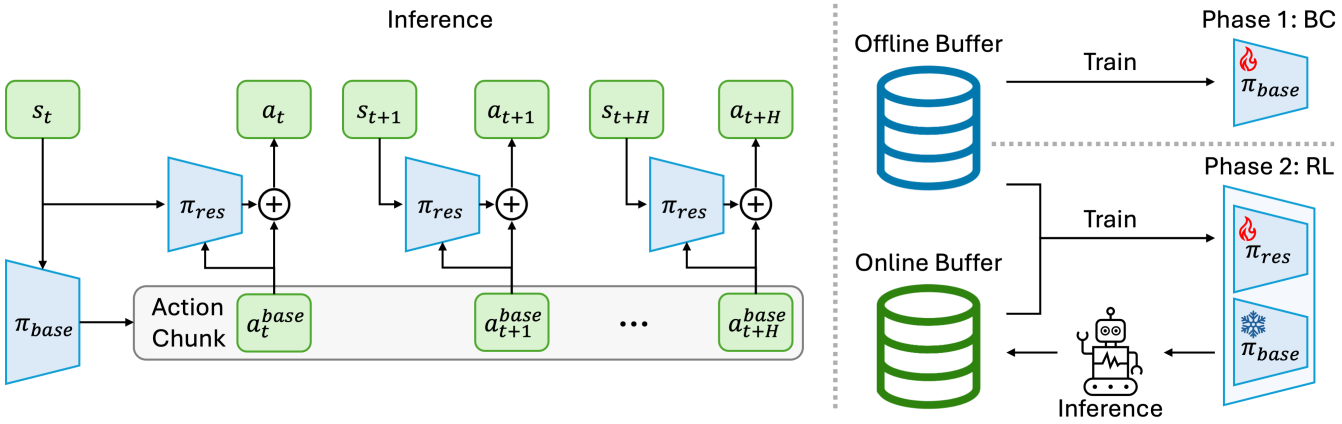

Fig. 2. Off-policy residual finetuning (ResFiT): A two-phase approach using online RL to improve BC policies. First, we train a base policy using BC on a dataset of demonstrations and then freeze the base. Then, we learn a residual policy with RL to correct the base actions. The RL phase utilizes our 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.

II. RELATED WORK

Behavior Cloning. BC has shown success in various settings, from autoregressive next-token prediction [1], [24], [39], [40] and diffusion policies [4], [6], [7], [41], [42], to large transformers that directly output robot actions [5], [43], [44]. Recent advances [8], [9], [45] 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 [6], [12], [13].

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 [4], [6] with large neural networks, which can lead to unstable learning when combined with conventional RL methods. Several works address these challenges [46], [47]. IBRL [48] 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 [49]–[51] specifically adapt diffusion- or flow-based policies, such as DSRL [52], 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 [34]–[37] 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 [12], combines a modern action-chunked BC policy with a single-step RL residual policy; however, 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 [53] and off-policy RL [54], [55]; drawing from a host of these related works [56]–[58], 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. Concurrently, [38] 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 also shown success of finetuning a generalist policy with real-world residual RL [59].

Real-World RL. Prior work on real-world RL has progressed from fleet-scale grasping [60] to sample-efficient single-arm manipulation. Notably, SERL [61] demonstrated insertion and pick-and-place with few demonstrations and less than an hour of interaction, and follow-up work [62] extended this to bimanual control via human-in-the-loop exploration. Other works have demonstrated real-world RL for single-arm robots [48], [52], [63]–[65]. For higher-DoF systems such as locomotion [24], [25] and dexterous manipulation [13], [66], the predominant approach is sim-to-real transfer, though sim fidelity limits this to simpler contact

scenarios [67]. In contrast, we train RL directly on a high-DoF bimanual humanoid in the real world.

III. METHOD

Our method of off-policy RL for residual finetuning (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 all our experiments (both simulation and real-world), we use ACT [5] as the base policy, which employs 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.

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 $\pi_\psi(a_{t:t+k}|o_t)$ 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 [4], [5], [68], [69] 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.

B. Finetuning 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) [70] 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 [71].

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 [31], [32], [38]) 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. At inference, the base policy is queried once per chunk, exactly as in standard BC—the first h actions in the chunk is played open-loop. At each timestep within the chunk, a residual correction is computed from the latest observation and added to the corresponding base action, making the combined policy closed-loop without modifying how the base policy is called.

Algorithm 1 ResFiT: Residual Finetuning 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 $a_t = \epsilon_t + a_t^{\text{b}}$ where $a_t^{\text{b}} \sim \pi_{\text{b}}(s_t)$
  Observe next state s_{t+1}, reward r_t, done flag d_t
  Add transition $(s_t, a_t^{\text{b}}, a_t, s_{t+1}, a_{t+1}^{b}, r_t, d_t)$ to $\mathcal{D}_{\text{online}}$
end for
repeat
  $a_t^{\text{b}} \sim \pi_{\text{b}}(s_t)$ and $a_t^{\text{r}} \sim \pi_\theta(s_t, a_t^{\text{b}})$
  Step env with $a_t = a_t^{\text{b}} + a_t^{\text{r}}$
  Add $(s_t, a_t^{\text{b}}, a_t, s_{t+1}, a_{t+1}^{b}, r_t, d_t)$ to $\mathcal{D}_{\text{online}}$
  Repeat UTD times:
    Sample batch B evenly from $\mathcal{D}_{\text{offline}}$, $\mathcal{D}_{\text{online}}$
    Update critic using B:
      $a_{t+1}^{\text{r}} \sim \pi_{\theta'}(s_{t+1}, a_{t+1}^{\text{b}})$
      Compute $a_{t+1} = a_{t+1}^{\text{b}} + a_{t+1}^{\text{r}}$
      $y = r_t + (1 - d_t) * \gamma * \min_{\text{subset(i)}} Q_{\phi_i'}(s_{t+1}, a_{t+1})$
      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:
      $\frac{1}{N} \sum_{i=1}^{N} Q_{\phi_i}(s_t, \pi_\theta(s_t, a_t^{\text{b}}) + a_t^{\text{b}})$
    Update actor target $\theta' \leftarrow \rho\theta' + (1-\rho)\theta$
until convergence

Following DDPG [71] and TD3 [55], we train the critic Q_ϕ via the mean-squared Bellman error and the actor via gradient ascent on the Q-values, adapted to our residual parameterization. Letting $\hat{a}_{t+1} = a_{t+1}^{\text{base}} + \pi_\theta(s_{t+1}, a_{t+1}^{\text{base}})$:

$$L(\phi) = \mathop{\mathrm{E}}_{\mathcal{D}}\Big[\big(Q_\phi(s_t, a_t) - r_t - \gamma(1-d_t)\, Q_\phi(s_{t+1}, \hat{a}_{t+1})\big)^2\Big], \tag{1}$$

$$L(\theta) = -\mathop{\mathrm{E}}_{\mathcal{D}}\Big[Q_\phi\big(s_t,\, a_t^{\text{base}} + \pi_\theta(s_t, a_t^{\text{base}})\big)\Big]. \tag{2}$$

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

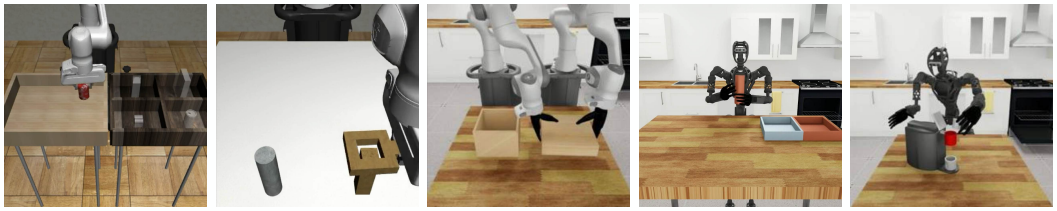

Fig. 3. Our simulation tasks from Robomimic and DexMimicGen, spanning single-arm manipulation as well as bimanual coordination tasks.

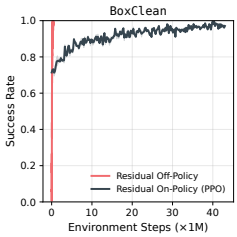

Fig. 4. PPO vs. off-policy RL in ResFiT.

the following, we detail the key design choices that were required to achieve stable residual finetuning 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 [58], [72]. We also use n-step returns ($n = 3$), following CQN [63] and IBRL [48], which help with sparse-reward tasks [73] by looking ahead n steps: $\sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n Q(s_{t+n}, a_{t+n})$ [74]. This is particularly important in our setting because with sparse binary rewards, 1-step returns only propagate the reward signal to the immediately preceding transition, whereas n-step returns propagate it n steps back, effectively speeding up the sparse reward propagation. 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 [58], we add layer normalization [75] 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 established practices: delayed actor updates, Polyak-averaged target networks ($\tau = 0.005$), and target policy smoothing, following TD3 [55]; REDQ [72] ensembles for reducing overestimation bias; a shallow ViT [76] encoder with DrQ-style [77] random shift augmentations; and symmetric sampling [58], where each training batch is drawn 50% from frozen demonstration data and 50% from the 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 our approach on tasks from Robomimic [78] and DexMimicGen [13], which vary in terms of robots, control modes, degrees of freedom, task horizons, and precision requirements. Our experiments utilize the Robosuite environment [79], which is built on MuJoCo [80], 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 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 sparse binary rewards: a reward of 1 is given only upon task completion, and 0 otherwise. 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. We compare against several baselines, all starting from the same demos. As an off-policy RL baseline without a base policy, we use RLPD [58]; since default RLPD cannot solve our high-DoF vision tasks, we create "Tuned RLPD" by incorporating the same design decisions from Sec. III-C. This means that the same number of demos, the same off-policy RL algorithm, and the same vision encoders and network architectures are used; the only difference from ResFiT is the absence of a base policy and residuals. We also compare to IBRL [48], which uses a pre-trained BC policy to propose actions and bootstrap target values. Finally, "Filtered BC" starts from the same base policy as ResFiT but finetunes via iterative rollouts, adding successes back for continued BC training, similar to reward-weighted regression [81] and self-imitation [2], [69], [82]–[85].

Other design decisions that we ablate include whether we use the layer norm for the actor and critic MLPs, whether

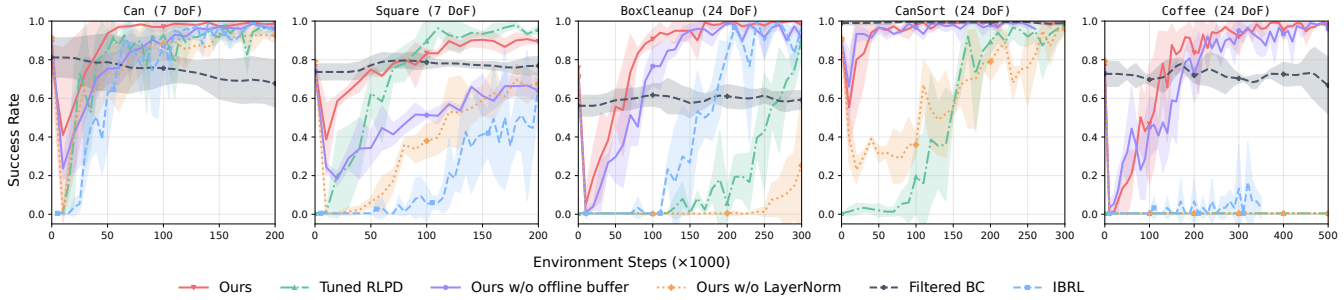

Fig. 5. Success rates of different approaches on our simulation tasks, showing ResFiT converging to high-performing policies for all tasks. Note that all approaches here start with the same number of demos.

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], [86]. To address these issues and fairly quantify performance differences between pre-trained and finetuned 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 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 [87] algorithm. As shown in Fig. 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 Fig. 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 $\sim$150k steps, while 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 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

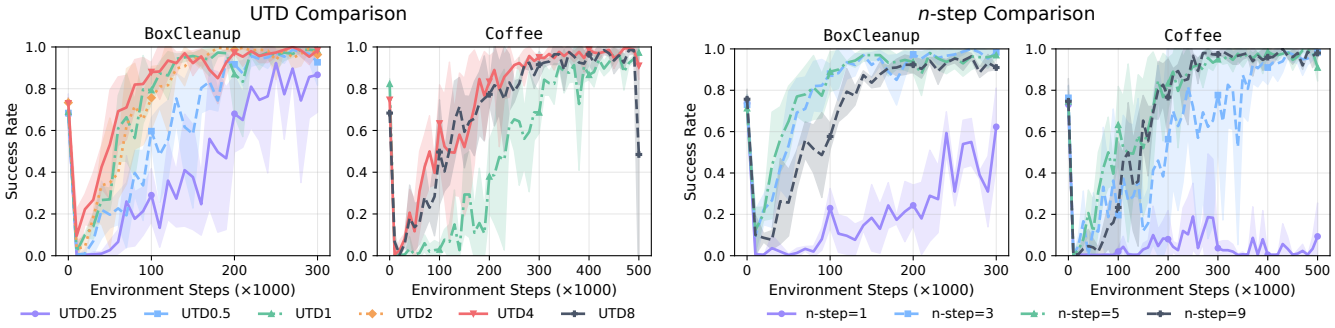

Fig. 6. Impact of UTD ratio (left) and `n-step` (right) on the `BoxCleanup` and `Coffee` tasks.

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 [69].

In Fig. 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 [5] 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 (≈ 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 (≈ 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.

In our real-world experiments, the base policies achieved relatively low success rates (14% and 23%), partly due to environmental drift between data collection and deployment. Rather than investing in retraining the base, we focus on the *improvement*: ResFiT can take whatever BC policy happens to be available and substantially boost its performance. The fact that our method can recover from such low starting points highlights the practical value of residual RL as a lightweight post-hoc improvement step.

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, relaxing the frozen base constraint—e.g., by distilling the improved combined policy back into the base to create room for further residual improvement—could yield additional gains. 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, though our experiments still required human supervision for resets and reward labeling. Without automatic reset mechanisms and success detection, even sample-efficient RL cannot scale independently of human oversight. A direct comparison with

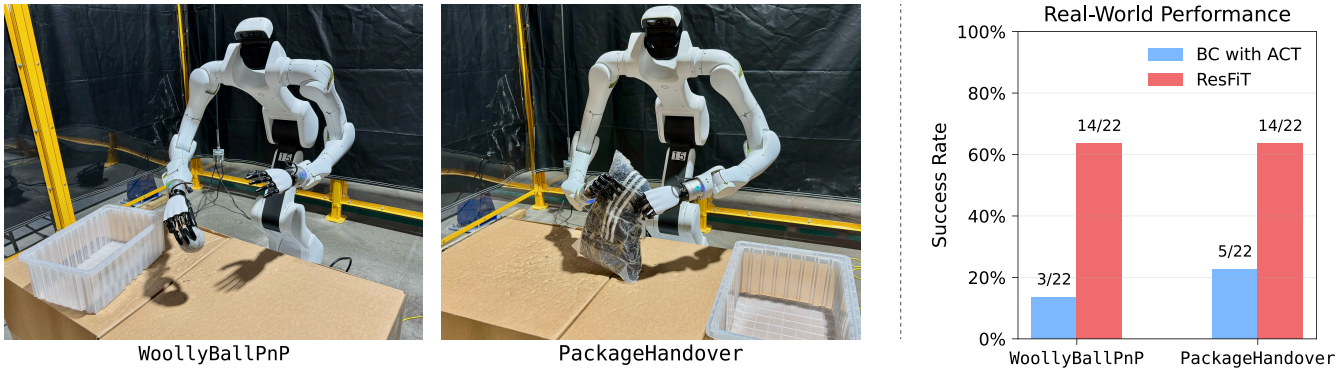

Fig. 7. Real-world results of applying ResFiT, where our residual RL approach shows a significant boost in performance over the base model, on a 29-DoF bimanual wheeled humanoid robot with two 5-fingered hands.

concurrent approaches such as DSRL [52], as well as more extensive real-world ablations, would be valuable future work.

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

[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.*, "π0: 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 re-orientation," 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] S. Tan, K. Dou, Y. Zhao, and P. Krähenbühl, "Interactive post-training for vision-language-action models," *arXiv preprint arXiv:2505.17016*, 2025.

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

[35] 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)*, 2019.

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

[37] 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.

[38] L. Dodeja, K. Schmeckpeper, S. Vats, T. Weng, M. Jia, G. Konidaris, and S. Tellex, "Accelerating residual reinforcement learning with uncertainty estimation," *arXiv preprint arXiv:2506.17564*, 2025.

[39] 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.

[40] 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.

[41] 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.

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

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

[44] 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.

[45] 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.

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

[47] 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.

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

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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

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

[55] 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.

[56] 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.

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

[58] 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.

[59] W. Xiao, H. Lin, A. Peng, H. Xue, T. He, Y. Xie, F. Hu, J. Wu, Z. Luo, L. Fan *et al.*, "Self-improving vision-language-action models with data generation via residual rl," *arXiv preprint arXiv:2511.00091*, 2025.

[60] 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.

[61] 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

[62] 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

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

[64] 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

[65] 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

[66] 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.

[67] M. Torne, A. Jain, J. Yuan, V. Macha, L. Ankile, A. Simeonov, P. Agrawal, and A. Gupta, "Robot learning with super-linear scaling," *arXiv preprint arXiv:2412.01770*, 2024.

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

[69] 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.

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

[71] 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.

[72] 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

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

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

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

[76] 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.

[77] 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

[78] 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.

[79] 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.

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

[81] 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.

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

[83] 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

[84] 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

[85] 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

[86] 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

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