EXPO-FT: Sample-Efficient Reinforcement Learning Finetuning for Vision-Language-Action Models

Kuo-Han Hung
Stanford University
khhung@stanford.edu

Abstract: (one-page extended abstract) Recent Vision-Language-Action (VLA) models have demonstrated impressive capabilities in executing complex, multi-step robotic manipulation tasks by leveraging large-scale imitation learning. However, despite their strong generalization abilities, even the most capable pretrained VLAs often fail to achieve the reliability required for real-world deployment, where failures can be costly. Reinforcement learning (RL) fine-tuning offers a promising avenue for improving performance, but existing approaches either train policies from scratch without benefiting from pretrained VLA priors or fine-tune VLAs with poor sample efficiency and limited reliability.

In this work, we present EXPO-FT, a system for efficient and reliable RL fine-tuning of pretrained VLA models. Our framework builds on the recently proposed EXPO algorithm, a stable RL method designed for pretrained generative policies, and extends it to support the temporally extended action representations used by modern VLAs. To further improve sample efficiency, we incorporate human-in-the-loop corrective feedback that enables targeted interventions when autonomous exploration is insufficient. As a concrete instantiation, we adopt the $\pi_{0.5}$ VLA model as the backbone policy and develop a practical framework for online RL fine-tuning in real-world robotic settings.

We evaluate our approach on a diverse set of challenging manipulation tasks requiring varying degrees of precision, adaptability, and dynamic control, including routing string lights and inserting a power connector, pocketing a pool ball, and inserting a flower into a wine bottle. Across all tasks, EXPO-FT consistently outperforms both RL-from-scratch methods and prior VLA fine-tuning approaches. Notably, the full system achieves perfect reliability (30/30 successes) on every task while requiring only approximately 20 minutes of online robot interaction on average. These results demonstrate that, with the right algorithmic and system design choices, RL fine-tuning of pretrained VLAs can simultaneously achieve high sample efficiency and deployment-level reliability. We additionally release an open-source framework to support broader adoption of RL fine-tuning for robotic foundation models.

Disclaimer: some easy task like pick task is finished before this final project and other tasks and experiment and the analysis are done for the final project.

Abstract: The ability to efficiently and reliably learn new tasks has been a foundational challenge in robotics. Vision-Language-Action (VLA) models have demonstrated strong generalization across diverse manipulation tasks, yet pretrained policies consistently fall short of the reliability required for real-world deployment. Reinforcement learning (RL) fine-tuning offers a promising path to bridge this gap, but existing approaches either train from scratch without fully leveraging pretrained priors, or fine-tune VLAs without achieving the sample efficiency and success rates that practical deployment demands. We present EXPO-FT, a system for stable, sample-efficient RL finetuning of pretrained VLA policies that closes this gap. Our system solves a suite of challenging manipulation tasks, including routing string lights and inserting the plug to light it up, striking a pool ball into a pocket, and inserting a flower into a wine bottle, each requiring combinations of high precision, dynamic actions, and robustness to varied initial states. Our system achieves perfect task performance (30/30 successes) across all evaluated tasks within an average of 19.1 minutes of online robot data, outperforming both prior RL-from-scratch and VLA finetuning approaches. We release an open-source codebase with the aim of facilitating broader adoption of RL finetuning of VLA models in robotics. Disclaimer: some easy task like pick task is finished before this final project and other tasks and experiment and the analysis are done for the final project.

1 Introduction

The ability to efficiently and reliably learn new tasks has been a foundational challenge in robotics. Recent advances in large-scale imitation learning with Vision-Language-Action models (VLAs) have yielded pretrained policies capable of executing complex, multi-step behaviors across diverse task settings [1, 2]. Yet despite their promise, even the most capable pretrained models fall persistently short of achieving reliable task success for deployment, which is consequential as failures in the real world are costly. In this work, we study the problem of finetuning VLAs with RL efficiently and stably to a reliable performance. We believe bridging this gap can unlock a new class of policies capable of reliable, real-world deployment.

Existing approaches of training real-world robot policies via RL fall into two camps. The first comprises methods that train RL policies without finetuning a VLA [3, 4, 5, 6], either from scratch or training lightweight models on top of the VLA. While some of these approaches achieve strong asymptotic performance, including near-perfect success rates in certain settings [3], they cannot be used to finetune pretrained VLAs and often cannot leverage the rich semantic and behavioral priors embedded in large pretrained VLAs. The second category consists of methods that finetune pretrained VLAs, thereby inheriting their generalizable representations [7, 8, 9, 10, 11]. However, these methods either fail to reach a reliable success rate, require prohibitively large numbers of environment interactions, or both. The gap of frameworks that simultaneously exploits the prior of a pretrained VLA and achieves the sample efficiency and reliability necessary for practical deployment motivates our work.

In this paper, we develop a system for finetuning pretrained robotic policies with reinforcement learning. Our system, EXPO-FT, demonstrates that RL finetuning of VLA models can be both sample efficient and reliable. We elucidate design choices that address specific challenges of reinforcement learning on top of pretrained models in robotics. To address the challenge of sample efficient finetuning of pretrained robotics policies, we build on the recently proposed EXPO algorithm [12], which provides a principled foundation for RL fine-tuning in this regime. We extend EXPO to incorporate temporally extended actions that most modern VLAs operate over. We also integrate human-in-the-loop feedback [13] to provide corrections during online training, enabling targeted interventions that improve sample efficiency where autonomous exploration is insufficient. As a concrete instantiation of our framework, we adopt $\pi_{0.5}$ [1] as the backbone VLA model—a state-

of-the-art generalist policy that has demonstrated strong performance across a diverse range of manipulation tasks.

We empirically find that our system achieves dexterous and precise manipulation capabilities across a diverse set of challenging tasks, including routing string lights and inserting the power connector to illuminate them, striking a pool ball into a pocket, and inserting a flower into a wine bottle. These tasks require varying combinations of precision, adaptability to diverse initial states, and dynamic actions in the case of striking a pool ball, collectively representing significant challenges for robotic manipulation. Compared to state-of-the-art prior methods, our system outperforms both methods that perform reinforcement learning from scratch and approaches that build on pretrained models, as shown in Figure 1. Notably, the full combination of our proposed components achieves the high reliability that is necessary for real deployment scenarios (30/30 success) across all of these complex tasks in an average of 20 minutes of online data.

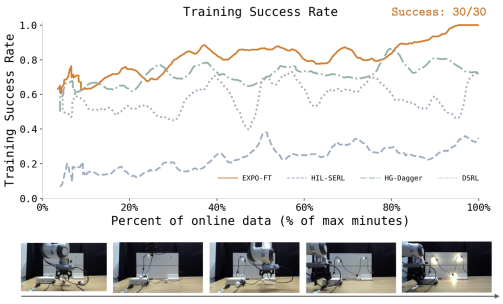

Figure 1: **Average training success rates** of EXPO-FT compared to prior methods. EXPO-FT achieves a reliable performance with high sample efficiency where prior methods often do not converge reliably.

Our primary contributions are a system for highly sample-efficient, reliable finetuning of VLA models with reinforcement learning, and an easy-to-use, open-source codebase to support such fine-tuning. We demonstrate that, with the right design, RL finetuning of VLAs can be sample efficient and achieve highly reliable success rates. We hope this work serves as a stepping stone toward broader adoption of finetuning generalist VLA models with reinforcement learning in the robotics community.

2 Related Work

Real-world reinforcement learning. Reinforcement learning has a long history of real-world deployment in robotics [14, 15, 16, 17, 18, 19, 20, 21]. In real-world reinforcement learning, sample efficiency remains a central challenge addressed by numerous prior algorithmic works [12, 22, 23, 24, 25]. Recent efforts have explored assembling these components together into sample-efficient real-world RL systems [3, 4, 6, 26, 27, 28], with extensions such as human-in-the-loop feedback [3] providing further gains in sample efficiency. While many of these systems achieve high task success rates, they rely on small Gaussian policies that cannot leverage large pretrained priors. In contrast to prior work, we approach real-world RL through the lens of finetuning a pretrained VLA model, leveraging it simultaneously as a generalist policy capable of complex manipulation and as an informative prior that enables sample-efficient online learning.

Reinforcement Learning with VLAs. There has been growing interest in finetuning VLAs with RL. Classical RL algorithms for continuous control are designed around Gaussian policies, which are not directly applicable to many modern VLAs that use more expressive policy classes such as flow matching or diffusion. Several works address this mismatch by developing RL algorithms tailored to flow or diffusion policies [12, 8, 29, 7, 30, 31, 32, 33]. $\pi_{0.6}$ [34] trains a VLA policy with advantage conditioning but in an offline setting whereas we focus on online finetuning, allowing for more data efficient learning. Multiple online RL methods use on-policy algorithms to finetune the VLA [35, 10, 36, 37, 7, 11]. We instead use off-policy RL for increased sample efficiency. Existing methods for using off-policy RL with VLAs often either train a lightweight auxiliary policy [5, 38, 39, 40, 41] or perform optimization in a latent space [8, 9, 42]. Our work differs from these prior methods along a few key dimensions. First, rather than only training a separate lightweight policy or decoupling VLA training from the RL loop, we directly finetune the full VLA within a unified RL pipeline, yielding more coherent training. Second, we incorporate human-in-the-loop interventions to accelerate learning. Third, many prior methods [38, 40, 41] operate on single-step

actions. In contrast, we optimize over the action chunks natively produced by the VLA, preserving the temporal abstraction central to its design. Together, these design choices yield an efficient and performant VLA finetuning framework, as demonstrated by our experimental results.

3 Background

We consider the standard reinforcement learning framework, where problems are modeled by a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, T, \gamma, \rho)$. In the MDP, $\mathcal{S}$ is the state space and $\mathcal{A}$ is the action space. At each timestep t, the agent observes state $s_t \in \mathcal{S}$ and selects action $a_t \in \mathcal{A}$ according to policy $\pi : \mathcal{S} \to \mathcal{A}$, and receives scalar reward $r(s_t, a_t) \in \mathbb{R}$. The environment transitions following the transition dynamics of the MDP $s_{t+1} \sim T(\cdot \mid s_t, a_t)$, with starting state initialized from $s_0 \sim \rho(\cdot)$. The tuple (s_t, a_t, r_t, s_{t+1}) is added in the replay buffer $\mathcal{D}$ for learning. The goal of reinforcement learning is to maximizes the expected discounted return $\mathbb{E}_\pi\left[\sum_{t=0}^{T} \gamma^t\, r(s_t, a_t)\right]$, where $\gamma \in [0, 1]$ is the discount factor.

EXPO [12]. To finetune the VLA policy with RL and leverage the prior of the pretraining model, we build on the recently proposed highly sample efficient and stable RL algorithm for training expressive flow or diffusion policies, EXPO [12]. Unlike classical sample-efficient RL algorithms which are designed around Gaussian policies and cannot be applied to pretrained VLAs that use flow or diffusion policies, EXPO provides a principled foundation for RL fine-tuning in this regime.

EXPO maintains two parameterized policies: a base flow policy, which in our case is the VLA model π_{VLA} trained with a supervised loss, and a lightweight edit policy π_{edit} trained to maximize the Q-function:

$$\mathcal{L}(\pi_{\text{edit}}) = -\mathbb{E}_{(s_t, a_t)\sim\mathcal{D},\ \hat{a}_t \sim \pi_{\text{edit}}}\left[Q_\phi(s_t,\ a_t + \hat{a}_t) - \alpha \log \pi_{\text{edit}}(\hat{a}_t | s_t, a_t)\right] \tag{1}$$

The edit policy predicts an edit $\hat{a}$ constrained to $[-\beta, \beta]$ that is added to the base action a to produce the edited actions $\tilde{a} = a + \hat{a}$. This avoids gradient backpropagation through the VLA model while still grounding TD updates in near-optimal actions. The final inference and TD backup policy is an on-the-fly (OTF) policy that selects the value-maximizing candidate across base and edited actions:

$$\tilde{a}^* = \underset{a \,\in\, \bigcup_{i=1}^{N} \{a_i, \tilde{a}_i\}}{\arg\max} \; Q_\phi(s, a) \tag{2}$$

with the Q-function trained via TD learning:

$$\mathcal{L}(\phi) = \mathbb{E}_{(s_t, a_t, s_{t+1})\sim\mathcal{D}}\left[\left(r_t + \gamma Q_{\phi'}(s_{t+1}, \tilde{a}^*_{t+1}) - Q_\phi(s_t, a_t)\right)^2\right]. \tag{3}$$

4 EXPO-FT

In this section, we present our complete system for sample-efficient, reliable finetuning for VLA models with reinforcement learning. Our central design goal is to finetune VLA models to a performance necessary for real-world deployment with high sample efficiency across different use cases. We start by describing the problem statement (Section 4.1), then describe the approach used for finetuning (Section 4.2), and lastly describe the implementation details (Section 4.3). The complete system of EXPO-FT is summarized in Figure 2.

4.1 Problem Statement

We consider the problem of finetuning a pretrained vision-language-action (VLA) model π_{VLA}, which often is a generalist policy trained on a diverse corpus of demonstrations and conditioned on natural language instructions to specify the desired task, with reinforcement learning. Modern VLAs often employs action chunking, predicting a sequence of H future actions $a_{t:t+H}$ at each timestep. Because of the rich behavioral prior acquired during pretraining, VLA policies are often capable of completing tasks zero-shot to some success rate. However, for tasks that π_{VLA} does not reach a reasonable

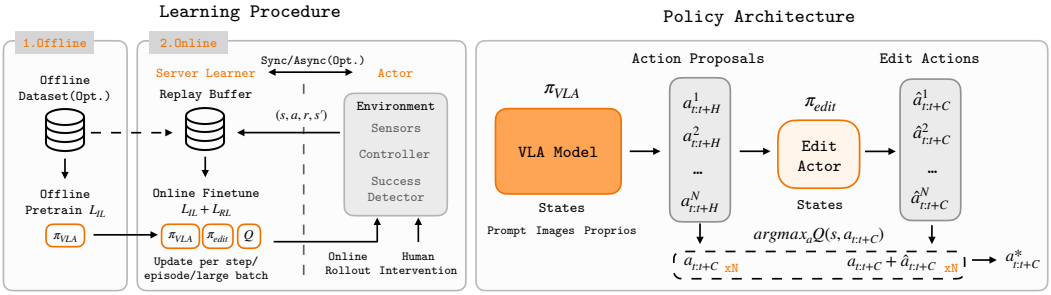

Figure 2: **Left: Overview of EXPO-FT.** EXPO-FT features a server that handles VLA training and inference and a learner process that steps in the environment to enable VLA finetuning with RL. **Right: Architecture of EXPO-FT**. EXPO-FT finetunes the VLA model with EXPO for sample-efficient training.

zero-shot performance, we follow standard offline-to-online RL practices and assume access to a small offline dataset of expert demonstrations $\mathcal{D}_0$, collected via human teleoperation,, which is used to bootstrap the finetuning process. The dataset is collected by a human teleoperation. We adopt a sparse binary reward $r \in [0, 1]$ indicating successful task completion. The task completion classifier may be either rule-based or learned. Observations consist of multi-view RGB images—comprising a wrist-mounted camera and a fixed side-view camera—augmented with robot proprioceptive state. Policy parameters are updated either after every environment step, at the end of each episode, or at fixed episode-batch intervals. The objective is to maximize task success rate.

A central requirement of our setting is practicality on real robots, which places strong constraints on sample efficiency and algorithmic compatibility. To this end, we build upon EXPO because of its sample efficiency and compatibility with diffusion- and flow-matching-based policy architectures, which are predominant in modern VLAs. However, the original formulation of EXPO neither supports action chunking nor incorporates human-in-the-loop feedback. The former is important because most modern VLAs operate by predicting multi-step action chunks rather than single actions; the latter has been shown to significantly improve sample efficiency. In the following section, we discuss how these challenges are addressed in EXPO-FT.

4.2 Sample-efficient VLA finetuning with RL

Temporally extended actions. Rather than predicting single-step actions, most modern VLAs operate over temporally extended action sequences, outputting a chunk of H consecutive actions $a_{t:t+H} = (a_t, a_{t+1}, \ldots, a_{t+H-1})$ and executing $C \leq H$ at each timestep. On top of EXPO, we incorporate temporally extended actions to be compatible with the native interface of most modern VLAs. We adapt the edit policy and Q-function to operate over chunks of executed actions $a_{t:t+C}$. Concretely, the edit policy predicts edits $\hat{a}_{t:t+C}$ and the edited actions that are selected are $\tilde{a}^i_{t:t+C} = a^i_{t:t+C} + \hat{a}^i_{t:t+C}$. The edit policy loss is

$$\mathcal{L}(\pi_{\text{edit}}) = -\mathbb{E}_{(s_t, a_{t:t+C}) \sim \mathcal{D},\ \hat{a}_{t:t+C} \sim \pi_{\text{edit}}} \left[Q_\phi(s_t,\ a_{t:t+C} + \hat{a}_{t:t+C}) - \alpha \log \pi_{\text{edit}}(\hat{a}_{t:t+C} | s_t, a_{t:t+C}) \right] \quad (4)$$

with the Q-function trained via TD learning:

$$\mathcal{L}(\phi) = \mathbb{E}_{(s_t, a_{t:t+C}, s_{t+C}) \sim \mathcal{D}} \left[\left(r_t + \gamma Q_{\phi'}(s_{t+C}, \tilde{a}^*_{t+C:t+2\times C}) - Q_\phi(s_t, a_{t:t+C}) \right)^2 \right] \quad (5)$$

The base VLA is updated using its original training objective, without modification.

Human-in-the-loop interventions. To make learning more sample-efficient and reliable, we use human-in-the-loop interventions during the training process. Specifically, during RL training, a human operator can intervene to provide action corrections when necessary. Interventions can be done for any individual actions within an action chunk. At any consecutive timesteps t' to t'' within the action chunk $a_{t:t+C}$, the human can intervene with actions $\bar{a}_{t':t''}$ where the robot then performs the actions the human provided, and the corresponding timesteps are overwritten with the corrective action, keeping the rest of the action chunk the same yielding action sequence $(a_t, \ldots, \bar{a}_{t'}, \ldots, \bar{a}_{t''} \ldots, a_{t+C-1})$. Interventions can happen across action chunks as well. The

chunks of actions are then added to the replay buffer after execution. We find the interventions are especially helpful to provide correct signal to reduce the amount of exploration necessary.

Training process. We now describe the end-to-end training procedure of EXPO-FT. For each task, we begin by configuring the camera setup and defining the reward signal, which can take the form of a rule-based criterion, a learned binary classifier, or any alternative reward specification such as a learned reward model. Before online RL begins, we assess whether the VLA can perform the task zero-shot at a reasonable success rate. If not, we collect a set of human demonstrations and finetune the VLA with imitation learning until it reaches a success rate around 40% or above. This data can either be retained as a separate offline dataset or used to initialize the replay buffer. We initialized the replay buffer directly in our experiments as our dataset sizes are small. With the supervised finetuned VLA policy, we start online RL training. The actor executes rollouts in the environment, with a human operator available to intervene via teleoperation. The updates can be done per step, per episode, or per batch of episodes, depending on the task requirements and available compute.

4.3 Implementation Details

We make several design choices to allow flexibility of our system depending on the task and compute available. We first present these details of our system then describe the training process.

Vision backbones. Visual inputs appear in both the actor and the critic from the side and wrist cameras. Since we use a VLA as the actor, it comes equipped with a pretrained visual encoder. For the critic, one natural option is to share this encoder, leveraging the same rich visual representations. However, the VLA encoder is large, making shared inference computationally prohibitive during the tight actor-learner loop. We find that equipping the critic with a separate, lightweight ResNet-50 [43] encoder can achieve high task performance at substantially lower computational cost. We include full details on architecture choices in Appendix Section C.

Human interventions. In our experiments, the human provides more interventions earlier in the training process where the robot need more corrections, and the intervention rate reduces to zero as the robot learns its policy to successful complete the task. We use a SpaceMouse to provide interventions. The human operator observes the policy actions and engages with the SpaceMouse when interventions are required. The actions from the SpaceMouse then automatically overrides the policy actions and gets executed by the robot.

Learning procedure. Training a VLA in an online RL loop presents a practical latency challenge as the model size is large and because of that, inference, environment interaction, and gradient updates all take a significant amount of time. We therefore decouple the system into a learner and an actor. The learner is responsible for VLA training and inference, and maintaining the data and replay buffer. The actor runs separately and communicates with the real-world environment to execute actions, collect human interventions, and send (s, a, r, s') tuples back to the learner's replay buffer. The learner then updates the agent either per step, per episode, or per large batch of data. The first option is the most sample efficient given enough compute, whereas updating per episode or large batch of data can better satisfy the safety requirement of deployment. The learner can communicate both synchronously and asynchronously between the server and actor; the asynchronous mode allows gradient updates and environment interaction to proceed in parallel, improving overall training throughput when additional GPUs are available. We use synchronous communication as it is faster when GPU is limited (2 or under). This learner actor interface described is released with our open-source codebase.

Reward design. An accurate reward function is important for obtaining a reliable performance for deployment. We adopt a sparse binary reward signal to avoid the burden of task-specific reward shaping. For each task, we specify a rule-based binary classifier that gives a positive reward only upon task completion and zero otherwise. For example, in the candy scoop task, we first verify that candies are present in the scoop once the scoop is raised above a height threshold above the bin, with candies determined by pixel color matching, then check whether the candy count decreases by at least a threshold amount within a specified xyz region corresponding to the target container. We defer the reward classifier of all tasks to Appendix Section B. This approach is straightforward

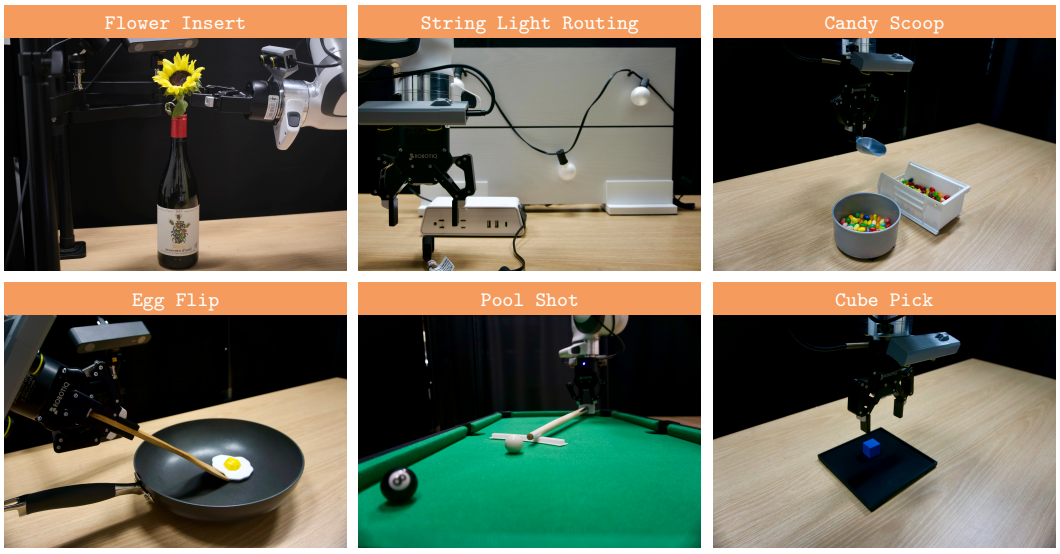

Figure 3: **Eight real-world manipulation tasks in our evaluation suite.** Flower Insert (tight insertion tolerances), String Light Routing - RouteI/II, Insert (long-horizon precise alignment), Egg Flip (dynamic contact-rich tool use), Candy Scoop (stable control in visually messy scenes), Pool Shot (precise speed control) and Cube Pick (large scene randomization). The tasks span dexterous, precise, deformable, and dynamic manipulation.

to set up, generalizes across diverse tasks, and sidesteps the engineering overhead of dense reward specification.

5 Experiments

We evaluate EXPO-FT on 8 challenging real-world robotic manipulation tasks requiring a combination of dynamic actions, high precision, and diverse initial states, and compare against strong prior methods for finetuning pretrained models and training from scratch to answer whether EXPO-FT can efficiently finetune VLAs with RL to a high performance and reliability. We start by presenting the experiment details and task setup, followed by the results.

5.1 Experiment details

For all experiments, the robot is controlled using an end-effector action space consisting of Cartesian velocity commands and gripper velocity commands running at 10 HZ. The observations consist of two RGB images from side and wrist view cameras at 224x224 resolution and proprioceptive state representation, which includes the robot end-effector position and orientation. Environment resets are performed either automatically or manually, depending on the task.

We use a binary sparse reward indicating task completion for all tasks, with rule-based success detectors that achieve over 95% accuracy in identifying successful task completion. For evaluation, a human observes if the task is successful. For each task, we report the success rate across 30 evaluation trials and the total amount of online data used for learning. During evaluation, initial states are randomized using either scripted robot motions or human resets. Detailed reward specifications, task success detectors, reset mechanisms, and task randomization strategies for each task are provided in Appendix Section B. Our experiments are performed on two NVIDIA H200 GPUs for 8k to 20k environment steps depending on how many steps it takes to converge for the task.

5.2 Description of tasks

We select tasks that cover a broad range of manipulation challenges, including precise object alignment, flexible object handling, and dynamic interaction. An illustration of each task can be found in Figure 3 and Figure 6.

Egg Flip. In this task, the robot is required to flip a 3D printed fried egg in a pan using a spatula. The robot must slide the spatula underneath the egg, lift it with appropriate force and orientation, and execute a flipping motion without dropping or damaging the egg. The task is considered successful if the egg is fully flipped and remains inside the pan after manipulation. This task requires contact-rich manipulation and precise control of the spatula pose during dynamic interaction, where excessive force or inaccurate timing can cause the egg to slide, rotate unexpectedly, or fail to flip properly.

String Light Routing - Route I & II & Insert. The string light routing task requires the robot to complete a three-stage light assembly and activation process. The task is divided into three subtasks: hanging the first light bulb onto a nail, hanging the second light bulb onto another nail, and plugging the power cable into the power source to activate the lights. Each subtask requires precise alignment and manipulation within a limited workspace, where small positioning errors can prevent successful hanging or cable insertion. The success of each subtask is evaluated individually based on whether the corresponding light bulb is correctly hung or whether the power cable is successfully plugged in. This task requires accurate object alignment and stable control across sequential manipulation steps.

Candy Scoop. In this task, the robot is required to scoop candies from a container using a spoon and transfer them into another container. The robot must carefully control the spoon's pose and motion to collect and transport the candies with minimal spillage. The task is considered successful if most of the candies are transferred into the target container. This task requires stable motion control and precise scooping depth selection, since shallow or overly deep scoops reduce success. It also requires robust handling of deformable and noisy candy distributions with varying colors, which increases perceptual and manipulation difficulty.

Cube Pick. In this task, the robot is required to pick up a cube from a surface and lift it. The robot must move its gripper to the cube, align properly, and grasp it securely without slipping. The task is considered successful if the cube is successfully lifted off the surface. This task requires accurate positioning and stable grasping under large scene randomization, including edge cases where cubes are placed near the edge of the plate, which increases the risk of failed grasps or unstable contacts.

Flower Insert. In this task, the robot is required to insert a flower stem into a corresponding vase or holder. The robot must align the stem with the narrow opening and carefully guide it into the slot with controlled downward motion. The task is considered successful if the flower is fully inserted. This task requires precise alignment under tight tolerances, where small position or orientation errors can lead to failed insertion or deformation of the stem.

Pool Shot. In this task, the robot is required to strike the cue ball to pocket the black ball into a designated hole. The robot must control the striking force applied to the cue ball while ensuring a clean hit. The task is considered successful if the black ball is successfully pocketed and the cue ball remains out of the hole. This task requires precise force control, where insufficient force fails to pocket the ball and excessive force leads to undesired cue ball trajectories.

5.3 Comparisons

We compare our approach to two categories of methods: (1) methods that train reinforcement learning policies from scratch in the real world, and (2) methods that finetune pretrained vision-language-action (VLA) models. We describe each method below.

HIL-SERL [3]. HIL-SERL is a sample efficient real-world reinforcement learning system that incorporates human-in-the-loop interventions for policy learning, demonstrating reliable performance on complex manipulation tasks such as motherboard and IKEA furniture assembly. While HIL-SERL shares our design of leveraging human feedback to guide online learning, it is designed to train

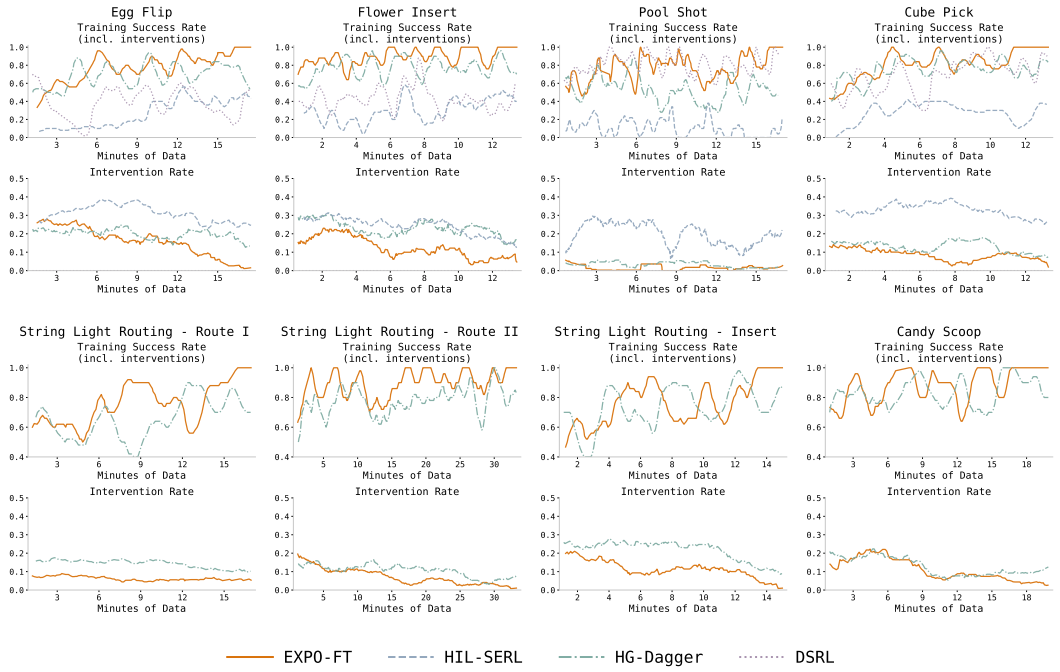

Figure 4: **Training success and intervention rates across all tasks. Top row:** Egg Flip, Flower Insert, Pool Shot, Cube Pick. **Bottom row:** String Light Routing - Route I, String Light Routing - Route II, String Light Routing - Insert, Candy Scoop.

policies from scratch and cannot finetune a pretrained VLA, nor exploit the behavioral priors encoded within one. As a result, despite achieving highly reliable performance in its original evaluations, it struggles in more complex settings such as those with large initial-state distributions, where EXPO-FT can naturally handle both with a more sample efficient RL algorithm and by bootstrapping from a pretrained VLA.

DSRL [8]. DSRL is a state-of-the-art reinforcement learning method for finetuning pretrained diffusion and flow-matching policies. By steering the latent noise, it learns the optimal actions within the prior, yielding substantial performance improvements over the base diffusion or flow policy. DSRL can be directly applied to VLA finetuning by perturbing the model's noise samples to improve downstream task performance. However, DSRL is constrained to optimizing within the modes of the prior distribution, which heavily limits its ability to acquire new behavior, and cannot incorporate human interventions during online execution.

HG-DAgger [13]. HG-DAgger is a human-gated variant of the DAgger [44] algorithm that intermittently solicits expert corrections during online execution. HG-DAgger is a natural choice for improving VLA models because of its simplicity and direct applicability to pretrained VLAs; however, HG-DAgger relies exclusively on behavioral cloning from human interventions, optimizes no reward signal, and cannot leverage suboptimal experience collected during autonomous execution.

Supervised finetuning. As an additional reference point, we report the performance of standard supervised finetuning of the base VLA on the task demonstration data, prior to any online interaction.

5.4 Experiment results

We now present the experimental results. The full results are shown in Table 1 and Table 2. Across all eight tasks, EXPO-FT achieves an average success rate of 30/30, outperforming both HG-DAgger (22.1/30), SFT (20.5/30) on all tasks, and HIL-SERL (5.5/30), and DSRL (19/30) on the subset of tasks these methods were evaluated on. Comparing performance of the prior methods, the

Table 1: **Success rates across selected tasks comparing EXPO-FT to all points of comparison.** Performance is reported as successful trials out of 30. HIL-SERL (M) denotes HIL-SERL trained with additional samples, as the standard training budget is insufficient for it to begin learning on Cube Pick and Pool Shot. While HIL-SERL achieves highly reliable performance in its original evaluations, our experiments involve a substantially larger initial state space, under which HIL-SERL fails to achieve high performance consistently.

Task	Success Rate (x/30)					
	SFT	HG-DAgger	DSRL	HIL-SERL	HIL-SERL (M)	EXPO-FT
Egg Flip	16/30	18/30	15/30	13/30	-	**30/30**
Cube Pick	22/30	26/30	24/30	0/30	27/30	**30/30**
Pool Shot	23/30	14/30	25/30	1/30	13/30	**30/30**
Flower Insertion	14/30	24/30	12/30	8/30	-	**30/30**
Average	18.8/30	20.5/30	19/30	5.5/30	20/30	**30/30**

Table 2: **Comparison of success rates across all tasks** against SFT and HG-DAgger. Performance is reported as successful trials out of 30, with improvements over SFT shown in parentheses.

Task	Online data (min)	Success Rate (x/30)		
		SFT	HG-DAgger	EXPO-FT
Egg Flip	18	16/30	18/30	**30/30 (+88%)**
String Light Routing - Route I	18	23/30	18/30	**30/30 (+30%)**
String Light Routing - Route II	35	21/30	25/30	**30/30 (+43%)**
String Light Routing - Insert	16	23/30	24/30	**30/30 (+30%)**
Candy Scoop	20	22/30	28/30	**30/30 (+36%)**
Cube Pick	14	22/30	26/30	**30/30 (+36%)**
Flower Insert	14	14/30	24/30	**30/30 (+114%)**
Pool Shot	18	23/30	14/30	**30/30 (+30%)**
Average	19.1	20.5/30	22.1/30	**30/30 (+44%)**

performance gap compared to HG-DAgger is most pronounced on tasks that require dynamic action or high precision — notably Egg Flip and Pool Shot — where imitation learning even with human corrections struggles to recover from compounding errors and learn precise actions, while EXPO-FT learns through self-improvement to correct and refine its behavior over training. Comparing to other RL methods, both HIL-SERL and DSRL fall significantly behind on tasks that involve a wider initial state distribution, for example Egg Flip, where the rich semantic representations encoded in the VLA initialization allow the policy to generalize across diverse initial states. Although DSRL leverages the pretrained VLA as its starting point, its optimization is constrained to modes already covered by the offline data, limiting its ability to acquire different behavior. HIL-SERL additionally struggles on the Pool Shot task in our setting as it requires highly precise motions to strike the ball into the hole, and the base RLPD algorithm simply struggles to learn that given the number of samples. Our approach, EXPO-FT, combines the strengths of these approaches: it initializes from a pretrained VLA prior, and then applies RL with online human-in-the-loop feedback to directly optimize performance.

The training data column in Table 2 reports the total interaction time required to reach the reported performance for all methods. EXPO-FT reaches reliable performance within 19.1 minutes of real-world interaction on average across tasks, demonstrating the high sample-efficiency of our RL finetuning system. This efficiency arises from both a strong prior provided by the VLA, which reduces the need for extensive exploration in the early stages of training, and a highly sample efficient finetuning algorithm and procedure.

We also provide the training success rate curves and intervention rate during online training in Figure 4 and an additional average episode time in Figure 5 in Appendix Section A.1. As shown in the figures, EXPO-FT consistently improves throughout training while prior methods often do not reach a reliable performance.

Overall, our results demonstrate that EXPO-FT is both general and effective across tasks spanning a wide range of manipulation challenges, including dynamic interactions, precise object alignment, and deformable object handling. Using the same training approach across all tasks, EXPO-FT achieves the highest performance in every setting evaluated, validating the effectiveness and high sample efficiency of EXPO-FT for online RL finetuning of VLA models.

6 Discussion & Conclusion

In this work, we present EXPO-FT, a system for sample-efficient, reliable reinforcement learning finetuning of vision-language-action (VLA) models. Across a diverse set of challenging robotic manipulation tasks, EXPO-FT substantially improves upon the pretrained VLA baseline and achieves highly reliable performance within an average of 19.1 minutes of online robot interaction data — demonstrating that RL finetuning of VLAs is both practical and broadly beneficial. Despite the promising results, EXPO-FT has limitations. Our current pipeline requires human environment resets between episodes, which can introduce operational burden at scale. Automating the reset process remains an important direction for future work. Because we finetune a pretrained VLA model, which typically comprises billions of parameters, running training and inference at high frequencies remains computationally challenging. Addressing this constraint is another direction we leave for future work.

7 Team Contribution

There is only one person on the final project.

References

[1] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

[2] G. R. Team, A. Abdolmaleki, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, A. Balakrishna, N. Batchelor, A. Bewley, J. Bingham, M. Bloesch, K. Bousmalis, P. Brakel, A. Brohan, T. Buschmann, A. Byravan, S. Cabi, K. Caluwaerts, F. Casarini, C. Chan, O. Chang, L. Chappellet-Volpini, J. E. Chen, X. Chen, H.-T. L. Chiang, K. Choromanski, A. Collister, D. B. D'Ambrosio, S. Dasari, T. Davchev, M. K. Dave, C. Devin, N. D. Palo, T. Ding, C. Doersch, A. Dostmohamed, Y. Du, D. Dwibedi, S. T. Egambaram, M. Elabd, T. Erez, X. Fang, C. Fantacci, C. Fong, E. Frey, C. Fu, R. Gao, M. Giustina, K. Gopalakrishnan, L. Graesser, O. Groth, A. Gupta, R. Hafner, S. Hansen, L. Hasenclever, S. Haves, N. Heess, B. Hernaez, A. Hofer, J. Hsu, L. Huang, S. H. Huang, A. Iscen, M. G. Jacob, D. Jain, S. Jesmonth, A. Jindal, R. Julian, D. Kalashnikov, M. E. Karagozler, S. Karp, M. Kecman, J. C. Kew, D. Kim, F. Kim, J. Kim, T. Kipf, S. Kirmani, K. Konyushkova, L. Y. Ku, Y. Kuang, T. Lampe, A. Laurens, T. A. Le, I. Leal, A. X. Lee, T.-W. E. Lee, G. Lever, J. Liang, L.-H. Lin, F. Liu, S. Long, C. Lu, S. Maddineni, A. Majumdar, K.-K. Maninis, A. Marmon, S. Martinez, A. H. Michaely, N. Milonopoulos, J. Moore, R. Moreno, M. Neunert, F. Nori, J. Ortiz, K. Oslund, C. Parada, E. Parisotto, A. Paryag, A. Pooley, T. Power, A. Quaglino, H. Qureshi, R. V. Raju, H. Ran, D. Rao, K. Rao, I. Reid, D. Rendleman, K. Reymann, M. Rivas, F. Romano, Y. Rubanova, P. P. Sampedro, P. R. Sanketi, D. Shah, M. Sharma, K. Shea, M. Shridhar, C. Shu, V. Sindhwani, S. Singh, R. Soricut, R. Sterneck, I. Storz, R. Surdulescu, J. Tan, J. Tompson, S. Tunyasuvunakool, J. Varley, G. Vesom, G. Vezzani, M. B. Villalonga, O. Vinyals, R. Wagner, A. Wahid, S. Welker, P. Wohlhart, C. Wu, M. Wulfmeier, F. Xia, T. Xiao, A. Xie, J. Xie, P. Xu, S. Xu, Y. Xu, Z. Xu, J. Yan, S. Yang, S. Yang, Y. Yang, H. H. Yu, W. Yu, W. Yuan, Y. Yuan, J. Zhang, T. Zhang, Z. Zhang, A. Zhou, G. Zhou, and Y. Zhou. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer, 2025. URL https://arxiv.org/abs/2510.03342.

[3] J. Luo, C. Xu, J. Wu, and S. Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning, 2025. URL https://arxiv.org/abs/2410.21845.

[4] 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, 2025. URL https://arxiv.org/abs/2401.16013.

[5] C. Xu, J. T. Springenberg, M. Equi, A. Amin, A. Esmail, S. Levine, and L. Ke. Rl token: Bootstrapping online rl with vision-language-action models, 2026. URL https://arxiv.org/abs/2604.23073.

[6] K. Lei, H. Li, D. Yu, Z. Wei, L. Guo, Z. Jiang, Z. Wang, S. Liang, and H. Xu. Rl-100: Performant robotic manipulation with real-world reinforcement learning, 2026. URL https://arxiv.org/abs/2510.14830.

[7] 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. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mEpqHvbD2h.

[8] 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, 2025. URL https://arxiv.org/abs/2506.15799.

[9] Y. Li, X. Ma, J. Xu, Y. Cui, Z. Cui, Z. Han, L. Huang, T. Kong, Y. Liu, H. Niu, W. Peng, J. Qiao, Z. Ren, H. Shi, Z. Su, J. Tian, Y. Xiao, S. Zhang, L. Zheng, H. Li, and Y. Wu. Gr-rl: Going dexterous and precise for long-horizon robotic manipulation, 2025. URL https://arxiv.org/abs/2512.01801.

[10] H. Li, Y. Zuo, J. Yu, Y. Zhang, Y. Zhaohui, K. Zhang, X. Zhu, Y. Zhang, T. Chen, G. Cui, D. Wang, D. Luo, Y. Fan, Y. Sun, J. Zeng, J. Pang, S. Zhang, Y. Wang, Y. Mu, B. Zhou, and N. Ding. SimpleVLA-RL: Scaling VLA training via reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=TQhSodCM4r.

[11] K. Chen, Z. Liu, T. Zhang, Z. Guo, S. Xu, H. Lin, H. Zang, X. Li, Q. Zhang, Z. Yu, G. Fan, T. Huang, Y. Wang, and C. Yu. π_{RL}: Online rl fine-tuning for flow-based vision-language-action models, 2026. URL https://arxiv.org/abs/2510.25889.

[12] P. Dong, Q. Li, D. Sadigh, and C. Finn. EXPO: Stable reinforcement learning with expressive policies. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=aFjSjkB6CV.

[13] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, page 8077–8083. IEEE Press, 2019. doi:10.1109/ICRA.2019.8793698. URL https://doi.org/10.1109/ICRA.2019.8793698.

[14] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[15] 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*, pages 651–673. PMLR, 2018.

[16] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016. URL http://jmlr.org/papers/v17/15-522.html.

[17] M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal. Learning force control policies for compliant manipulation. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4639–4644, 2011. doi:10.1109/IROS.2011.6095096.

[18] M. P. Deisenroth, C. E. Rasmussen, and D. Fox. Learning to control a low-cost manipulator using data-efficient reinforcement learning. In H. Durrant-Whyte, N. Roy, and P. Abbeel, editors, *Robotics: Science and Systems VII*. The MIT Press, 06 2012. ISBN 9780262305969. doi:10.7551/mitpress/9481.003.0013. URL https://doi.org/10.7551/mitpress/9481.003.0013.

[19] T. C. Kietzmann and M. A. Riedmiller. The neuro slot car racer: Reinforcement learning in a real world setting. *2009 International Conference on Machine Learning and Applications*, pages 311–316, 2009. URL https://api.semanticscholar.org/CorpusID:17199272.

[20] J. Kober, K. Mülling, O. Krömer, C. H. Lampert, B. Schölkopf, and J. Peters. Movement templates for learning of hitting and batting. In *2010 IEEE International Conference on Robotics and Automation*, pages 853–858, 2010. doi:10.1109/ROBOT.2010.5509672.

[21] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 2008. ISSN 0893-6080. doi:https://doi.org/10.1016/j.neunet.2008.02.003. URL https://www.sciencedirect.com/science/article/pii/S0893608008000701. Robotics and Neuroscience.

[22] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine. Efficient online reinforcement learning with offline data. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[23] P. Dong, A. M. Lessing, A. S. Chen, and C. Finn. Reinforcement learning via implicit imitation guidance, 2026. URL https://openreview.net/forum?id=CgupPwA40q.

[24] X. Chen, C. Wang, Z. Zhou, and K. W. Ross. Randomized ensembled double q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=AY8zfZm0tDd.

[25] M. Nauman, M. Ostaszewski, K. Jankowski, P. Miłoś, and M. Cygan. Bigger, regularized, optimistic: scaling for compute and sample-efficient continuous control, 2024. URL https://arxiv.org/abs/2405.16158.

[26] L. Ankile, Z. Jiang, R. Duan, G. Shi, P. Abbeel, and A. Nagabandi. Residual off-policy rl for finetuning behavior cloning policies, 2025. URL https://arxiv.org/abs/2509.19301.

[27] J. Luo, P. Dong, Y. Zhai, Y. Ma, and S. Levine. Rlif: Interactive imitation learning as reinforcement learning. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 36329–36351, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/9c537882044c8b5352c363e840872ddb-Paper-Conference.pdf.

[28] P. Dong, S. Mirchandani, D. Sadigh, and C. Finn. What matters for batch online reinforcement learning in robotics? In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=usw1NVkczu.

[29] L. Yang, Z. Huang, F. Lei, Y. Zhong, Y. Yang, C. Fang, S. Wen, B. Zhou, and Z. Lin. Policy representation via diffusion probability model for reinforcement learning, 2023. URL https://arxiv.org/abs/2305.13122.

[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, 2024. URL https://arxiv.org/abs/2412.06685.

[31] P. Dong, A. Swerdlow, D. Sadigh, and C. Finn. Faster: Value-guided sampling for fast rl, 2026. URL https://arxiv.org/abs/2604.19730.

[32] M. Psenka, A. Escontrela, P. Abbeel, and Y. Ma. Learning a diffusion model policy from rewards via q-score matching, 2024. URL https://openreview.net/forum?id=StkLULT1i1.

[33] Q. Li and S. Levine. Q-learning with adjoint matching. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=vd4eNAdtO6.

[34] P. Intelligence, A. Amin, R. Aniceto, A. Balakrishna, K. Black, K. Conley, G. Connors, J. Darpinian, K. Dhabalia, J. DiCarlo, D. Driess, M. Equi, A. Esmail, Y. Fang, C. Finn, C. Glossop, T. Godden, I. Goryachev, L. Groom, H. Hancock, K. Hausman, G. Hussein, B. Ichter, S. Jakubczak, R. Jen, T. Jones, B. Katz, L. Ke, C. Kuchi, M. Lamb, D. LeBlanc, S. Levine, A. Li-Bell, Y. Lu, V. Mano, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, C. Sharma, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, W. Stoeckle, A. Swerdlow, J. Tanner, M. Torne, Q. Vuong, A. Walling, H. Wang, B. Williams, S. Yoo, L. Yu, U. Zhilinsky, and Z. Zhou. $\pi^*_{0.6}$: a vla that learns from experience, 2025. URL https://arxiv.org/abs/2511.14759.

[35] J. Liu, F. Gao, B. Wei, X. Chen, Q. Liao, Y. Wu, C. Yu, and Y. Wang. What can RL bring to VLA generalization? an empirical study. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=qmBMPInbZC.

[36] S. Tan, K. Dou, Y. Zhao, and P. Krähenbühl. Interactive post-training for vision-language-action models, 2025. URL https://arxiv.org/abs/2505.17016.

[37] G. Lu, W. Guo, C. Zhang, Y. Zhou, H. Jiang, Z. Gao, Y. Tang, and Z. Wang. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning, 2025. URL https://arxiv.org/abs/2505.18719.

[38] Y. Chen, S. Tian, S. Liu, Y. Zhou, H. Li, and D. Zhao. Conrft: A reinforced fine-tuning method for vla models via consistency policy, 2025. URL https://arxiv.org/abs/2502.05450.

[39] Y. Guo, J. Zhang, X. Chen, X. Ji, Y.-J. Wang, Y. Hu, and J. Chen. Improving vision-language-action model with online reinforcement learning. *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15665–15672, 2025. URL https://api.semanticscholar.org/CorpusID:275932066.

[40] W. Xiao, H. Lin, A. Peng, H. Xue, T. He, Z. Luo, Y. Xie, F. Hu, L. Fan, G. Shi, and Y. Zhu. Self-improving vision-language-action models with data generation via residual RL. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=eUGoqrZ6Ea.

[41] X. Yuan, T. Mu, S. Tao, Y. Fang, M. Zhang, and H. Su. Policy decorator: Model-agnostic online refinement for large policy model. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=e5jGTEiJMT.

[42] Y. Zhang, C. Wang, ouyang lu, Y. Zhao, Y. Ge, Z. Sun, X. Li, C. Zhang, C. Bai, and X. Li. Align-then-steer: Adapting the vision-language action models through unified latent guidance. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=T3i7Ifeatk.

[43] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

[44] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In G. Gordon, D. Dunson, and M. Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/ross11a.html.

[45] P. Dong, K.-H. Hung, A. Swerdlow, D. Sadigh, and C. Finn. Tql: Scaling q-functions with transformers by preventing attention collapse, 2026. URL https://arxiv.org/abs/2602.01439.

[46] P. Dong, C. Zheng, C. Finn, D. Sadigh, and B. Eysenbach. Value flows, 2026. URL https://arxiv.org/abs/2510.07650.

[47] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

A Additional Experiment Results

A.1 Training Episode Time

In addition, we provide training episode time plots throughout online training. As shown in the plots, EXPO-FT consistently reduces the episode completion time as training progresses, indicating improved task efficiency and smoother policy execution during online adaptation.

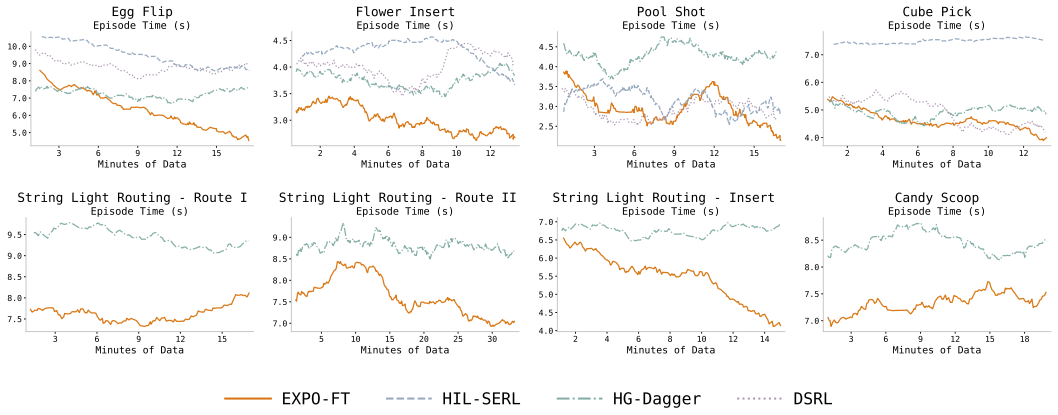

Figure 5: **Episode Time across all tasks. Top row:** Egg Flip, Flower Insert, Pool Shot, Cube Pick. **Bottom row:** String Light Routing - Route I, String Light Routing - Route II, String Light Routing - Insert, Candy Scoop.

B Detailed Task Setting

B.1 Task Setting Description

Here, we provide detailed descriptions of the data collection process, reward specification, task success detector, reset mechanism and task randomization for each task. We also provide detailed task completion strips in Figure 6.

Egg Flip. We pre-collect 25 demonstrations for this task. The task is counted as successful when the egg has been flipped to the opposite side while remaining inside the pan. To determine this, we record the egg's initial orientation and detect yolk pixels within a cropped region corresponding to the pan. The spatula remains grasped throughout training, and the robot returns to its initial position on each reset. A human reset is performed only if the egg falls outside the pan. The initial positions of the egg and spatula are randomized without constraints within approximately two thirds and one third of the pan, respectively.

String Light Routing - Route I. We pre-collect 40 demonstrations for this task. The task is counted as successful when the lightbulb's vertical position exceeds a fixed threshold and the gripper is open. The lightbulb position is tracked via pixel color detection. Human resets are performed between episodes. The lightbulb's initial position is randomized.

String Light Routing - Route II. We pre-collect 30 demonstrations for this task. The task is counted as successful when the lightbulb's vertical position exceeds a fixed threshold and the gripper is open. The lightbulb position is tracked via pixel color detection. Human resets are performed between episodes. The lightbulb's initial position is randomized.

String Light Routing - Insert. We pre-collect 25 demonstrations for this task. The task is counted as successful when the light turns on. During reset, the plug is dropped at a random location, and the robot automatically moves to a random reset position. The plug's initial position is randomized.

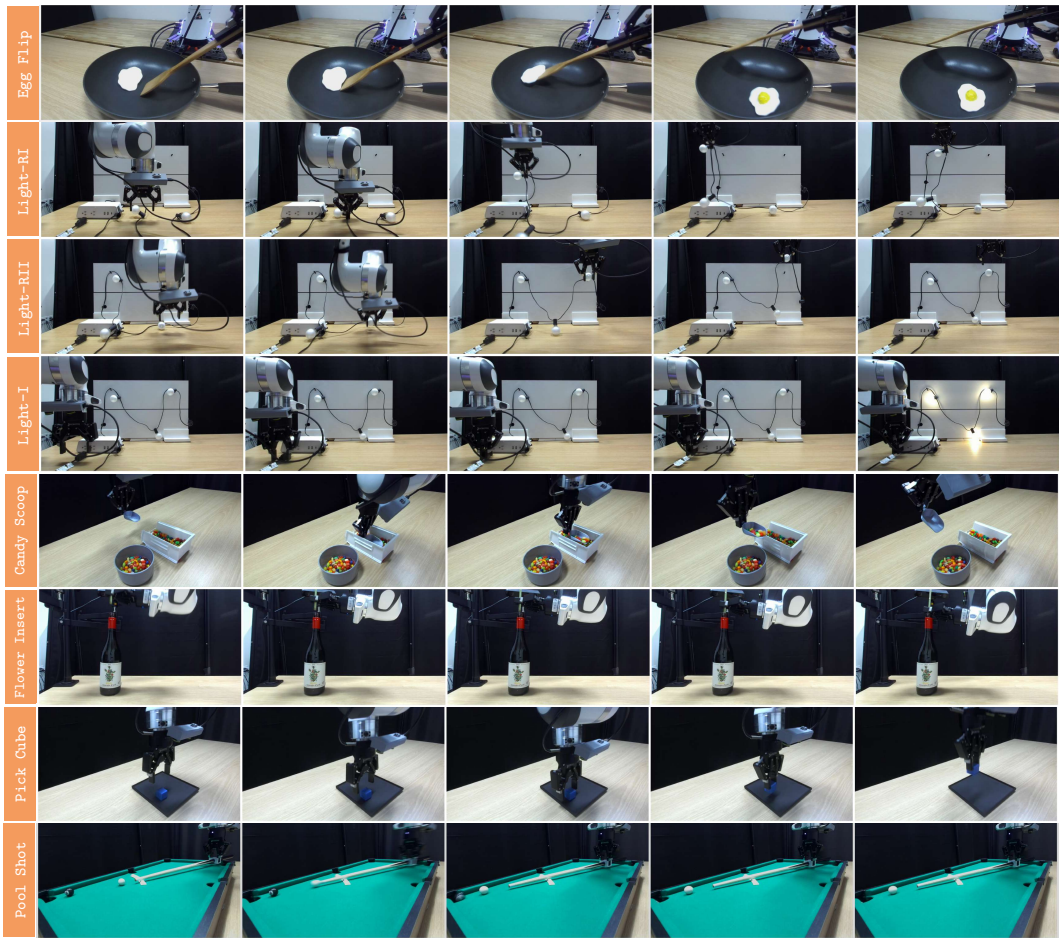

Figure 6: **Task strips demonstrating successful completion of each task.**

Candy Scoop. We pre-collect 20 demonstrations for this task. The reward classification for this task is split into two parts, both of which must succeed for the episode to be counted as successful. In the first part, we verify that candies are present in the scoop once the scoop is raised above a height threshold. In the second part, we check whether the candy count decreases to zero within a specified xyz region corresponding to the target container, indicating a successful pour (minor spillage is still considered successful). Candies are detected via pixel-based color matching. The scoop remains grasped throughout training, and the robot returns to its initial pose on each reset. A human reset is performed if there are too few candies remaining in the source container or if too many candies spill onto the table. The robot's initial position is randomized.

Cube Pick. We pre-collect 10 demonstrations for this task. The task is counted as successful when the gripper rises above a height threshold while performing a partial closing motion, indicating that an object is being grasped. During reset, the cube is moved to a random location and the robot moves to a random initial position. The cube's position is randomized across the entire black workspace plane.

Flower Insert. We pre-collect 25 demonstrations for this task. The task is counted as successful when the gripper is within a target region and no camera detects any portion of the flower stem below the mouth of the bottle. The stem is detected via pixel color matching. The gripper remains closed throughout training, and the robot moves to a random position on each reset. The robot's initial position is randomized.

Pool Shot. We pre-collect 30 demonstrations for this task. The task is counted as successful when three conditions are simultaneously satisfied: (1) the cue stick does not contact the black ball at any point during execution, (2) the black ball ends up in a pocket, and (3) the white ball does not end up in a pocket. Human resets are performed between episodes. The black ball's initial position is randomized along a fixed line.

B.2 Task Initial State Randomization Space

Here, we provide visualizations of the randomized initial state space for each task, as shown in Figure 7. The randomized regions are highlighted in orange.

For Egg Flip, both the egg position and scoop position are randomized. For String Light Routing, the cable positions are randomized. For Candy Scoop, the gripper position is randomized. For Cube Pick, the gripper position is randomized. For Flower Insert, the gripper position is randomized. For Pool Shot, the position of the black ball is randomized.

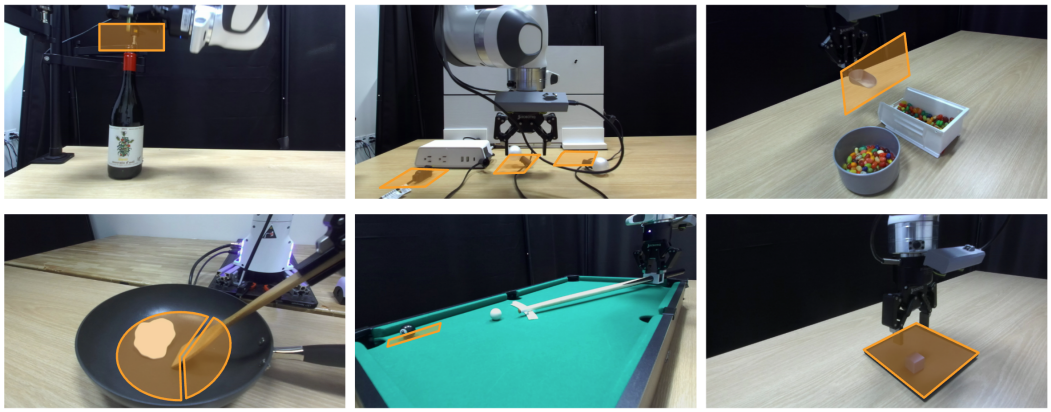

Figure 7: **Visualization of randomized initial state spaces for all tasks.** The orange regions indicate the randomized initialization areas used during training.

C Detailed Training Setting

C.1 Model Structure/Training Detailed

We instantiate EXPO-FT with $\pi 0.5$ [1] as the base policy, initialized from a task-specific LoRA [47] supervised-finetuning checkpoint and the matching normalization statistics for the robot setup. Input images are resized to 224×224 before encoding.

The value function is implemented as a RedQ-style [24] ensemble of 10 Q-networks with layer normalization. For each target value, we draw 2 networks at random from the ensemble, take their minimum to reduce overestimation, and apply slow Polyak averaging (standard soft update rate) toward a target ensemble. Discounting for multi-step action chunks is applied consistently with using 4 or 8 executed steps per replan (see Optimization).

Action sampling combines the base model with a small edit corrector. The base model draws 8 stochastic short-horizon action chunks per decision. edits are scaled by 0.05, 0.1 or 0.2 before being added to the base. Among the 16 candidates (8 base + 8 edit), the policy executes the chunk with the highest estimated Q-value (deterministic top-Q selection, not a softmax).

The critic's visual backbone is a ResNet-50 [43] with stage depths (3, 4, 6, 3) and 64 filters per stage, producing a 512-dimensional image embedding; proprio is embedded to 64 dimensions and fused with the image representation for Q. Hidden MLP layers for the critic and edit modules use three layers of width 256.

VLA's image encoder weights are frozen during RL; critic image features are still learned. During updates we apply OpenPI-style [1] augmentation to both camera views: a 95% random crop with resize to 224×224, rotation in $[-5°, 5°]$, and color jitter (brightness, contrast, saturation ±0.1). Each view and each of the current/next observations is augmented with an independent random draw.

C.2 Optimization Hyperparameters

In this section, we summarize the optimization hyperparameters used for EXPO-FT. We separate the hyperparameters into two groups: (1) global hyperparameters shared across all experiments, and (2) task-specific hyperparameters that are adjusted based on the manipulation difficulty and control requirements of each task.

All differentiable components are optimized using Adam with a learning rate of 3×10^{-4}. This includes the Q-ensemble, the tanh-bounded edit policy used to generate corrective actions, and the learnable temperature parameter in the maximum-entropy objective. The temperature parameter is initialized as $\alpha_0 = 1$.

We use a discount factor of $\gamma = 0.99$. The target Q-networks are updated using Polyak averaging with coefficient $\tau_Q = 5 \times 10^{-3}$. The target parameters of the base $\pi_{0.5}$ policy are updated using a faster Polyak averaging coefficient $\tau_\pi = 10^{-3}$.

Training uses replay minibatches of size $B = 64$. We employ a relatively high update-to-data ratio of $\mathrm{UTD} = 20$, corresponding to 20 gradient updates per environment-step-aligned batch of collected experience.

Table 3: Shared optimization hyperparameters used across all experiments.

Hyperparameter	Value
Optimizer	Adam
Learning rate	3×10^{-4}
Discount factor γ	0.99
Initial temperature α_0	1.0
Critic target update τ_Q	5×10^{-3}
Policy target update τ_π	10^{-3}
Batch size	64
Update-to-data ratio (UTD)	20

Table 4 summarizes the task-specific hyperparameters. The edit scale controls the magnitude of corrective edit actions added to the base policy. Tasks requiring precise manipulation generally use a smaller edit scale to avoid unstable corrections.

The replanning interval C determines how frequently the policy replans actions during execution. For tasks requiring higher precision or tighter feedback control, we use smaller replanning intervals to increase closed-loop responsiveness.

Finally, the number of updates per episode is selected such that the effective training ratio remains approximately close to one update per 40 environment steps across tasks with different episode lengths.

C.3 Baseline implementations

HG-DAgger [13] We train $\pi_{0.5}$ [1] with imitation learning only, where updates use supervised losses on human interventions together with offline data. Optimization uses Adam at 3×10^{-4}, a batch size of 64, and the same update per episode as EXPO-FT. The per-episode update schedule and replan horizon C are set identically to those of EXPO-FT on each task. We use *synchronous* training (the learner runs in the main environment loop).

Table 4: Task-specific hyperparameters.

Task	Edit Scale	Replan Steps C	Updates per Episode	Training Steps
Egg Flip	0.2	8	4	10000
String Light Routing – Route I	0.05	8	4	10000
String Light Routing – Route II	0.05	8	4	20000
String Light Routing – Insert	0.05	4	3	9000
Candy Scoop	0.2	8	6	12000
Cube Pick	0.2	8	3	8000
Flower Insert	0.05	4	1	8000
Pool Shot	0.05	8	1	10000

HIL-SERL [3] We follow the HIL-SERL [3] training setup using the RLPD algorithm [22] with the same $\pi_{0.5}$-style observation pipeline as our real-robot stack. The hyperparameters are: hidden sizes $(256, 256, 256)$, Adam with a learning rate of 3×10^{-4}, discount factor $\gamma = 0.99$, batch size 64, UTD ratio 10 (we also experimented with a UTD ratio of 20 as used in EXPO-FT, but found that UTD 10 performs better under asynchronous training), initial temperature 0.1, and Bellman backups *without* an entropy term. The critic uses the same REDQ-style [24] ensemble as EXPO-FT (10 Q-networks, with 2 subsampled for the target). Training runs in an *asynchronous* thread; the learner runs continuously and, in our setup, performs roughly *one* full optimizer step per environment step on average.

DSRL [8] We use the DSRL built on frozen $\pi_{0.5}$ VLA. The hyperparameters are: hidden sizes $(256, 256, 256)$, Adam at 3×10^{-4}, $\gamma = 0.99$, batch size 64, UTD ratio 10, initial temperature 0.1, no entropy term in the backup, a REDQ-style critic (10 networks, with two used for the target), a 512-D image latent and a 64-D state latent with a ResNet [43] $(3, 4, 6, 3)$ of width 64, state included in Q, and full image augmentation. Training is *asynchronous*, with the same approximate effective rate of one update per environment step.