Robotics: Science and Systems 2026
Sydney, Australia, July 13–17, 2026

$\pi^*_{0.6}$: a VLA That Learns From Experience

Physical Intelligence

Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alexander Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, Zhiyuan Zhou

https://pi.website/blog/pistar06

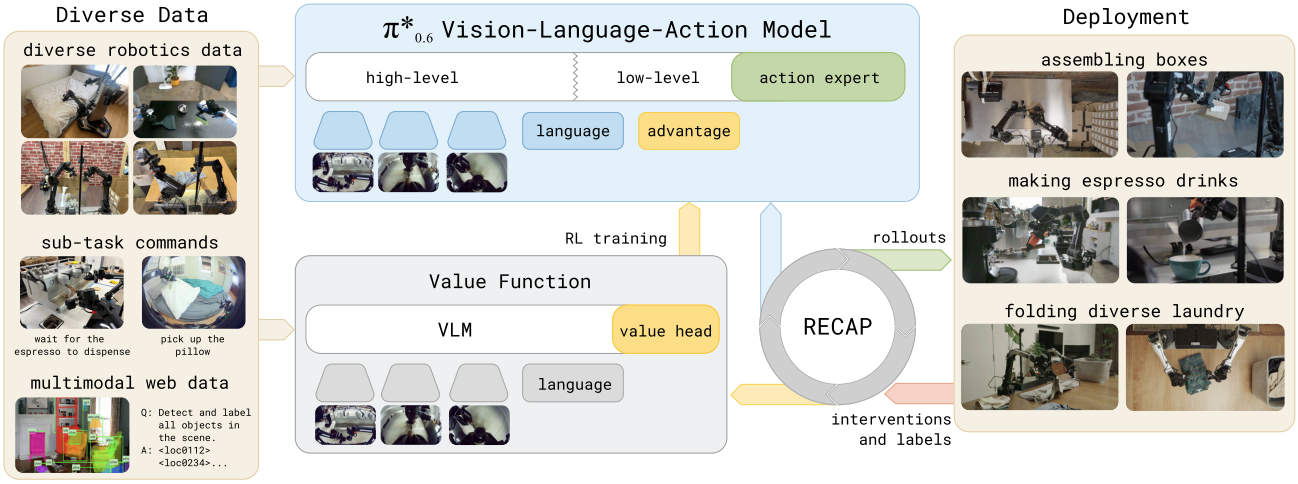


Fig. 1: **RECAP enables VLAs to improve performance during deployment.** Our model incorporates *advantage conditioning*, allowing it to ingest diverse types of real-world experience: human demonstrations, autonomous rollouts and online corrections. Learning from real-world interaction via advantage conditioning improves the model's understanding of how actions influence performance, leading to a faster and more successful policy.

***Abstract*—Vision–language–action (VLA) models offer a promising path toward general-purpose robots, but achieving the reliability and speed required for practical deployment remains challenging. We present a general-purpose method, RL with Experience and Corrections via Advantage-conditioned Policies (RECAP) that improves the efficiency and reliability of VLA policies by utilizing their real-world experience. Our method introduces value-based advantage conditioning during both pre-training and post-training phases, enabling VLA policies to ingest highly heterogeneous real-world experience, including human demonstrations, policy rollouts, and online correction data. We show that the $\pi^*_{0.6}$ model, trained with RECAP, achieves hours-long deployment of folding diverse laundry in real homes, reliably assembles boxes in a factory, and makes espresso drinks using a professional espresso machine. On some of the hardest tasks, RECAP more than doubles task throughput and roughly halves the task failure rate.**

I. INTRODUCTION

Practice makes perfect: while people are remarkably flexible in acquiring new skills, mastery invariably requires refinement through trial and error. General-purpose robotic foundation models, such as vision-language-action (VLA) models, now allow tasks to be specified flexibly via prompts. But just like people, they must *practice* to achieve mastery. This entails learning not only from demonstrations, but from autonomously collected experiential data that allows the policy to correct mistakes it makes during deployments, improve speed and robustness beyond human teleoperation, and adapt to new deployment conditions. Although the principles of learning through autonomous practice have been known for decades, as formalized with reinforcement learning (RL) [1], realizing them in a general and scalable robotic system remains challenging due to issues of scalability, data heterogeneity, and noisy reward signals.

We introduce RECAP, a training recipe that enables general-purpose vision–language–action (VLA) models to improve based on their own experiences and expert interventions. RECAP first pre-trains a VLA model on diverse, multi-task, multi-robot data via an offline RL step. During subsequent deployment, the VLA model is iteratively improved with policy rollouts, expert interventions, and sparse, binary reward feedback that reflects task success. Our recipe allows the model not only to imitate demonstrations, but to adapt to real deployment conditions, correct its own mistakes and improve efficiency beyond human teleoperation.

Training with RECAP follows an offline RL [2, 3] recipe (Figure 1): We first train a value function that estimates progress toward task completion from all (multi-task) data. This value function is then used to compute *advantages* for each action chunk in the data. We then obtain a pre-trained model that conditions the policy on a binarized improvement indicator based on this advantage [4]. After pre-training, RECAP fine-tunes the model on each downstream task with task-specific data, and then performs one or more rounds of

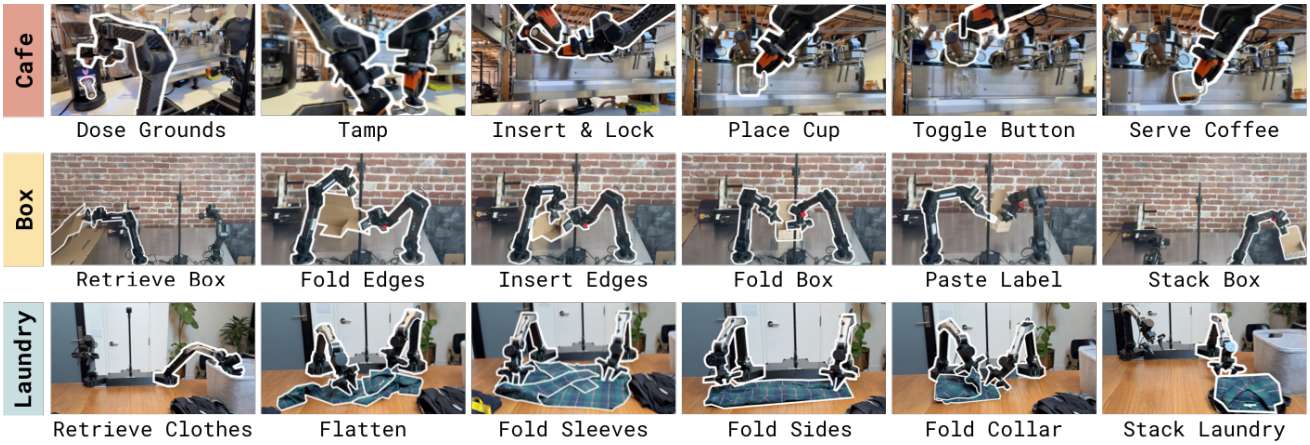


Fig. 2: **Real-world tasks improved by RECAP.** We evaluate RECAP on long-horizon manipulation tasks that require dexterous, multi-stage behavior: making espresso drinks, assembling cardboard boxes, and folding diverse laundry. These tasks include realistic sources of variability, such as deformable clothing, flattened boxes that stick together or bend, and liquid handling during espresso preparation.

on-robot data collection and offline RL to iteratively improve the policy.

We applied RECAP to challenging, long-horizon manipulation tasks such as folding diverse laundry, assembling boxes, and making espresso drinks (see Figure 2). Training with RECAP more than doubles the throughput on some of the hardest tasks, and can reduce failure rates by 2× or more, enabling levels of robustness required for real-world deployment: it ran continuously for 13 hours making espresso drinks, folded previously unseen laundry items in a new home for over two hours without interruption, and assembled boxes used in real factory packaging. While RECAP builds on algorithmic components explored in prior work, our system is novel in combining these elements, and the results show, for the first time, that a general-purpose RL recipe—combining autonomous deployment, sparse human reward feedback and expert interventions—can significantly improve both the robustness and throughput of VLA models at scale.

II. RELATED WORK

Policies trained with imitation learning (IL) are known to suffer from compounding errors [5] and, at best, can only be as performant as the demonstration data. The goal of this work is to improve the reliability and speed of VLA policies by going beyond IL from offline demonstrations. Prior works have used online interventions to improve robotic manipulation policies [6–9]. We adopt a form of such interventions, called human-gated DAgger [8, 10]. In contrast to these works, our method uses both expert interventions and fully autonomous experience, resulting in an RL-based framework that integrates multiple data sources. There is a large body of work on using RL for autonomous improvement of robotic manipulation policies [11–19], including methods using diffusion-based policies [20–22], in multi-task settings [23, 24], and using pre-trained multi-task policies [25–27]. Unlike these works, we study how to scale real-world RL to large VLA policies for long-horizon, fine-grained manipulation tasks.

Many recent works have studied how to improve a base VLA model through RL. Several works directly apply the proximal policy optimization (PPO) algorithm and variations thereof to VLA fine-tuning [28–32], yielding approaches that are difficult to extend to real-world RL in an efficient and scalable fashion. Others have explored RL fine-tuning *on top of* pre-trained VLA models, either training a residual policy [33, 34], fine-tuning an action head network [35], selecting or refining actions proposed by the VLA [36–38], or optimizing the initial noise of a diffusion-based VLA [39]. Some have also explored ways to distill the learned behavior back into the VLA [33, 34, 36, 40]. These prior works generally use discrete actions or simple Gaussian continuous action distributions. A critical distinction is that we train an entire VLA end-to-end using iterated offline RL, with an expressive flow matching VLA model. This is made possible by a simple and scalable advantage-conditioned policy extraction method, which removes much of the complexity of using policy gradient style objectives with large VLA models. Our comparisons show that this significantly outperforms a more traditional policy gradient based extraction scheme.

More closely related to RECAP in terms of methodology, a number of prior works have integrated value functions and end-to-end RL training of VLAs on real robots [41–44]. For example, Huang et al. [41] apply calibrated Q-learning to an offline demonstration dataset for grasping tasks, without an online improvement phase. Zhang et al. [42] use direct preference optimization (DPO) to optimize pick-and-place skills from human preferences, using online rollouts from a VLA. Finally, Zhai et al. [43], Ghasemipour et al. [44] use PPO and REINFORCE, respectively, with time-to-completion value functions to train VLAs for tasks like moving a bowl, unfolding a mat, and pushing objects on a table. In contrast to these prior works, we describe an iterated offline RL framework for VLAs with multiple advantages. First, our method supports high-capacity diffusion and flow-based VLAs, unlike the discrete-action models studied in

prior works. Second, we avoid the need for on-policy PPO or REINFORCE by using an advantage conditioning strategy for policy extraction, which can utilize all prior (off-policy or offline) data. Lastly, our evaluation consists of complex, dexterous, and temporally extended tasks, where our method increases throughput by about 2× while handling deformable objects, liquids, and multi-stage tasks.

Prior works have explored the idea of conditioning the policy on rewards, values, and advantages [45–54], including methods that use classifier-free guidance [4]. We extend this approach to pre-train and fine-tune a large-scale generalist VLA policy [55], incorporating a variety of data sources (including demonstrations, interventions, and autonomous policy rollouts) to learn real robotic manipulation tasks. Recent research has also studied how to effectively train multi-task, language-conditioned reward functions [56–62] and value functions [43, 63, 64]. Building on these works, we also train a language-conditioned distributional value function, which allows us to estimate state-action advantages.

III. Preliminaries

Reinforcement learning. We consider the standard RL setting in which an agent, given by a policy $\pi(\mathbf{a}_t|\mathbf{o}_t)$, outputs actions $\mathbf{a}_t$ given an observation $\mathbf{o}_t \in \mathcal{O}$. We define a trajectory as $\tau = (\mathbf{o}_0, \mathbf{a}_0, \cdots, \mathbf{o}_T) \in \mathcal{O} \times \mathcal{A} \cdots \mathcal{O}$. A distribution over trajectories $\rho_\pi(\tau)$ is induced by the policy $\pi(\mathbf{a}_t|\mathbf{o}_t)$ and the stochastic dynamics $p(\mathbf{o}_{t+1}|\mathbf{o}_t, \mathbf{a}_t)$: $\rho_\pi(\tau) = p(\mathbf{o}_0)\prod_{t=0}^{T-1}\pi(\mathbf{a}_t|\mathbf{o}_t)p(\mathbf{o}_{t+1}|\mathbf{o}_t, \mathbf{a}_t)$. The reward function is given by $r(\mathbf{o}_t, \mathbf{a}_t)$, and we abbreviate it to r_t to shorten notation, where r_T is the terminal reward. The goal of RL is to maximize the cumulative reward, learning a policy that maximizes $\mathcal{J}(\pi) = \mathbb{E}_{\tau\sim\rho_\pi}[R(\tau)] = \mathbb{E}_{\tau\sim\rho_\pi}[\sum_{t=0}^{T} r_t]$. The value function for a policy π is then defined as $V^\pi(\mathbf{o}_t) = \mathbb{E}_{\tau_{t+1:T}}[\sum_{t=t}^{T} r_t]$. We can then calculate the advantage of an action $\mathbf{a}_t$ as $A^\pi(\mathbf{o}_t, \mathbf{a}_t) = \mathbb{E}_{\rho_\pi(\tau)}[\sum_{t'=t}^{t+N-1} r_{t'} + V^\pi(\mathbf{o}_{t+N})] - V^\pi(\mathbf{o}_t)$, corresponding to an n-step estimate.

Regularized reinforcement learning. Instead of maximizing $\mathcal{J}(\pi)$, it is common to use regularization in RL, optimizing for a policy that maximizes reward while remaining close to some reference policy π_{ref} [65–69]. This is important, for example, when we want to train for many gradient steps on the same data, in which case π_{ref} typically corresponds to the behavior policy that collected the training data. This can be formalized via the objective $\mathcal{J}(\pi, \pi_{\text{ref}}) = \mathbb{E}_{\tau\sim\rho_{\pi_\theta}}[\sum_{t=0}^{T}\gamma^t r_t] - \beta\mathbb{E}_{\mathbf{o}\sim\rho_{\pi_\theta}}[D(\pi(\cdot|\mathbf{o})\|\pi_{\text{ref}}(\cdot|\mathbf{o}))]$, where D denotes some divergence metric. For the case where D is the KL divergence, we have the well-known result that $\hat{\pi}(\mathbf{a}|\mathbf{o}) \propto \pi_{\text{ref}}(\mathbf{a}|\mathbf{o})\exp(A^{\pi_{\text{ref}}}(\mathbf{o}, \mathbf{a})/\beta)$ is the solution to $\max_\pi J(\pi, \pi_{\text{ref}})$, with Lagrange multiplier β [66–69]. Our advantage-conditioned policy extraction method is based on a closely related but less well-known result: if we define the policy $\hat{\pi}(\mathbf{a}|\mathbf{o}) \propto \pi_{\text{ref}}(\mathbf{a}|\mathbf{o})p(I|A^{\pi_{\text{ref}}}(\mathbf{o}, \mathbf{a}))^\beta$, where $p(I|A^{\pi_{\text{ref}}}(\mathbf{o}, \mathbf{a})) = g(A^{\pi_{\text{ref}}}(\mathbf{o}, \mathbf{a}))/\int g(A^{\pi_{\text{ref}}}(\mathbf{o}, \mathbf{a}'))\mathrm{d}\mathbf{a}'$ is the probability of any action $\mathbf{a}$ improving over π_{ref} as measured by a monotonically increasing function g, then $\hat{\pi}$ is guaranteed to improve over π_{ref}, i.e., $\mathcal{J}(\hat{\pi}) \geq \mathcal{J}(\pi_{\text{ref}})$ [4, 70]. We will use this property in deriving our policy extraction method in Section IV-C. Using this definition we can then obtain a parametric policy from the closed form definition of $\hat{\pi}$ by solving the following minimization problem: $\min_\theta \mathbb{E}_{s\sim\rho_{\pi_{\text{ref}}}}[KL(\hat{\pi}, \pi_\theta)]$.

Algorithm 1 Overview of RECAP

Require: multi-task demonstration dataset $\mathcal{D}_{\text{demo}}$

// Pre-train

Train V_{pre} on $\mathcal{D}_{\text{demo}}$ using Eq. 1

Train π_{pre} on $\mathcal{D}_{\text{demo}}$ using Eq. 3 and V_{pre}

// Post-train on a target task

Initialize $\mathcal{D}_\ell$ with demonstrations for target task ℓ

Train π_ℓ^0 from π_{pre} on $\mathcal{D}_\ell$ with $I_t = \text{True}$

// Deploy and iteratively improve performance

for $k = 1$ to K **do**

  Collect deployment data with π_ℓ^{k-1} and add it to $\mathcal{D}_\ell$

  Train V_ℓ^k from V_{pre} on $\mathcal{D}_\ell$ using Eq. 1

  Train π_ℓ^k from π_{pre} on $\mathcal{D}_\ell$ using Eq. 3 and V_ℓ^k

end for

IV. RL with Experience and Corrections via Advantage-Conditioned Policies (RECAP)

We present RECAP, an RL-based training framework that enables VLA models to improve during deployment (shown in Figure 1 and Alg. 1). We use RECAP to train a VLA model that we refer to as $\pi^*_{0.6}$.

A. Overview

RECAP starts with a data set of diverse human-collected *demonstrations*, and iterates through three subroutines:

1) **Value function training.** We use all of the data collected to train a large, multi-task value function, $V^{\pi_{\text{ref}}}$, which models the expected time to task completion, thereby detecting failures or suboptimalities.
2) **Advantage conditioned training.** To improve the VLA policy, we include an optimality indicator based on advantage values derived from this value function in the VLA prefix. This "advantage conditioned" recipe provides a simple and effective way to extract improved policies from deployment data.
3) **Deployment data collection.** We run the VLA on a task, labeling each episode with task outcome labels (which determine the reward), and optionally providing human interventions that demonstrate corrections for mistakes in earlier iterations.

Our pre-training phase consists of performing steps (1) and (2) above on our entire pre-training dataset, which contains tens of thousands of hours of demonstrations from numerous tasks and a variety of different robots. Then, for each task, we perform steps (1), (2), and (3) one or more times to further improve the VLA. In practice, the specialists are each fine-tuned independently from the pre-trained model. We describe

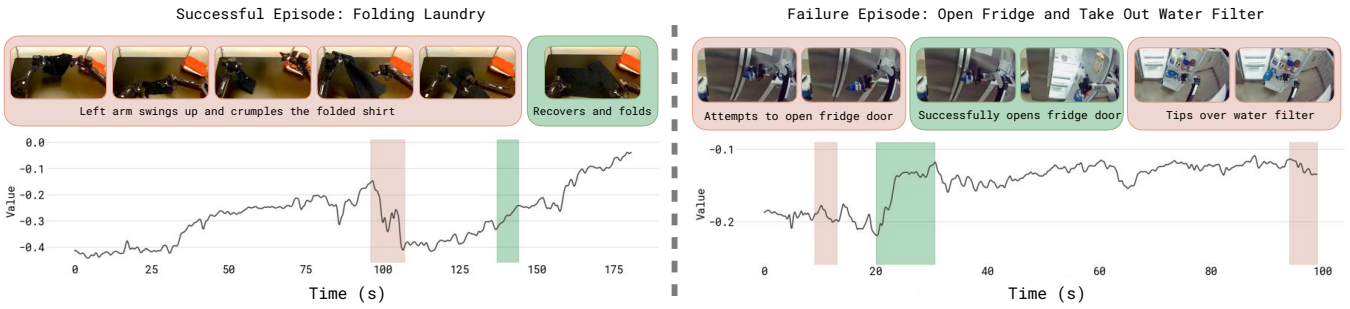


Fig. 3: **Value function visualizations.** We visualize our learned value function on a laundry folding episode that finished successfully (left), and a refrigerator manipulation episode that failed (right). The segments highlighted in red show examples of negative advantage, and green parts highlight areas of relatively high advantage, with the images on top showing the corresponding frames of the episode. These visualizations show that the value function correctly identifies mistakes in the episode as having negative advantage and models the speed of progress being made.

the value function training and policy training steps below, and then present our specific instantiation of this approach for training $\pi^*_{0.6}$ in Section V.

B. Distributional value function training

To train a value function that can act as a reliable critic for arbitrary tasks, we represent $V^{\pi_{\text{ref}}}$ with a multi-task distributional value function $p_\phi(V|\mathbf{o}_t, \ell) \in \Delta_B$ [71], mapping the observations $\mathbf{o}_t$ and language command ℓ to a distribution over B discrete bins. In our implementation, this value function uses the same architecture as the VLA policy but with a smaller VLM backbone. Using $R_t(\tau) = \sum_{t'=t}^{T} r_{t'}$ to denote the empirical return of a trajectory τ from time step t until the end, we train $p_\phi(V|\mathbf{o}_t, \ell)$ by first discretizing the empirical return value $R_t(\tau)$ into $B = 201$ bins (using R_t^B to denote the discretized returns), and then minimizing the cross-entropy H over the trajectories in the dataset $\mathcal{D}$:

$$\min_\phi \mathbb{E}_{\tau \in \mathcal{D}} \left[\sum_{\mathbf{o}_t \in \tau} H(R_t^B(\tau), p_\phi(V|\mathbf{o}_t, \ell)) \right]. \quad (1)$$

This is a Monte Carlo estimator for the value function of the policy represented by the dataset $\mathcal{D}$ (i.e., the behavior policy π_{ref}). We can extract a continuous value function, and thus an advantage, from the learned value distribution using $V^{\pi_{\text{ref}}}(o_t, \ell) = \sum_{b \in [0,B]} p_\phi(V = b|\mathbf{o}_t) v(b)$, where $v(b)$ denotes the value corresponding to bin b. During the pre-training phase, the dataset $\mathcal{D}$ corresponds to the human demonstrations, and the value function captures the expected return for the task and metadata we condition on. In subsequent iterations, it skews toward a weighted combination of the return of the demonstrations and the learned policy.

While this value function can be suboptimal compared to a Q-function estimator trained via temporal difference learning, we found it to be simple and highly reliable, while still allowing for substantial improvement over imitation learning. Our method could be extended to accommodate off-policy estimators in future work.

C. Policy extraction via advantage conditioning

Once we have the value function $V^{\pi_{\text{ref}}}$, we need an effective policy extraction method that satisfies several criteria. First, it needs to be able to handle diverse off-policy data, comprising the initial demonstrations, the expert interventions, and autonomous episodes from both the latest policy and older policies. Second, it needs to be applicable to large VLA models, including models that use flow matching or diffusion to generate actions. Third, to improve the policy with deployment data, it needs to learn from both good (near-optimal) and bad (suboptimal) data.

Among the existing methods for policy extraction, policy gradient methods (including regularized and reparameterized policy gradients) are perhaps the most widely used [65, 72]. However, these methods are difficult to apply to flow matching models, which do not readily provide a tractable log-likelihood, making them hard to scale up to modern VLA architectures (see comparison in Section VI). An alternative is to use weighted regression methods, such as AWR [67, 73, 74], which implicitly provide for regularization to the behavior policy and use a simple, importance-weighted supervised learning objective. However, these methods discard or significantly downweight a sizable portion of the data, effectively implementing a kind of filtered imitation technique.

Instead, we use a variant of *advantage conditioning* [46], where the policy is trained on all of the data with an additional input indicating the *optimality* of the action. This is closely related to a variety of methods that condition the policy on some function of the resulting trajectory [45, 48]. The specific formulation in our method is most similar to CFGRL [4]. Building on the formulation in Section III, we can apply Bayes' rule to rewrite the probability of policy improvement as $p(I|A^{\pi_{\text{ref}}}(\mathbf{o}, \mathbf{a})) = \pi_{\text{ref}}(\mathbf{a}|I, \mathbf{o}) / \pi_{\text{ref}}(\mathbf{a}|\mathbf{o})$. Applying this to our setting and including language conditioning, we can obtain an alternative closed form for the improved regularized policy described in Section III as

$$\hat{\pi}(\mathbf{a}, |\mathbf{o}, \ell) \propto \pi_{\text{ref}}(\mathbf{a}|\mathbf{o}, \ell) \left(\frac{\pi_{\text{ref}}(\mathbf{a}|I, \mathbf{o}, \ell)}{\pi_{\text{ref}}(\mathbf{a}|\mathbf{o}, \ell)} \right)^\beta. \quad (2)$$

For the special case $\beta = 1$, we get $\hat{\pi}(\mathbf{a}, |\mathbf{o}, \ell) = \pi_{\text{ref}}(\mathbf{a}|I, \mathbf{o}, \ell)$.

We can therefore represent $\hat{\pi}$ without needing to explicitly represent the improvement probability $p(I|A^{\pi_{\text{ref}}}(\mathbf{o}, \mathbf{a}))$, if we train the policy so that it can represent both $\pi_{\text{ref}}(\mathbf{a}|\mathbf{o}, \ell)$ and

$\pi_{\text{ref}}(\mathbf{a}|I, \mathbf{o}, \ell)$. This principle is similar to the approach in classifier-free guidance (CFG), where a diffusion model is trained to model the data both with and without a conditioning variable [4]. We assume the improvement indicator I follows a delta distribution

$$p(I|A^{\pi_{\text{ref}}}(o, a, \ell)) = \delta(A^{\pi_{\text{ref}}}(o, a, \ell) > \epsilon_\ell),$$

with a task dependent improvement threshold ϵ_ℓ. This threshold allows us to control the optimality indicator, and minimizes the need for finding an attenuation factor β to sharpen the improvement conditioned distribution after training.[1] The policy objective then corresponds to minimizing the following negative log-likelihood:

$$\min_\theta \mathbb{E}_{\mathcal{D}_{\pi_{\text{ref}}}}\Big[-\log \pi_\theta(\mathbf{a}_t|\mathbf{o}_t, \ell) - \alpha \log \pi_\theta(\mathbf{a}_t|I_t, \mathbf{o}_t, \ell)\Big], \quad \text{where } I_t = \mathbb{1}\big(A^{\pi_{\text{ref}}}(\mathbf{o}_t, \mathbf{a}_t, \ell) > \epsilon_\ell\big). \tag{3}$$

The advantage values $A^{\pi_{\text{ref}}}(\mathbf{o}_t, \mathbf{a}_t, \ell)$ are obtained from the value function in the previous section, and α is a trade-off hyperparameter.

In practice, $\mathcal{D}_{\pi_{\text{ref}}}$ contains data from a mixture of sources, including demonstrations and autonomous task attempts collected by previously deployed policies. The policy extraction objective applies to all such data once each action has been assigned an indicator I_t from its estimated advantage. Section V-C describes how these indicators are assigned in our full training pipeline, including the treatment of teleoperated interventions. As we will discuss in Section V, our VLA model produces both discrete and continuous outputs, with the continuous distribution represented via flow matching. Therefore, the real training objective combines likelihoods for the discrete values with the flow matching objective for the continuous values.

V. Implementation, Model, and System Details

We instantiate RECAP with a VLA model, based on the $\pi_{0.6}$ VLA [76], which improves upon $\pi_{0.5}$ [55] by leveraging a larger backbone and more diverse conditioning. As shown in Figure 4, $\pi^*_{0.6}$ modifies $\pi_{0.6}$ by introducing conditioning on the binarized advantage indicator I_t, making it suitable for RL. To estimate advantage during VLA training, we also train a VLM-initialized value function. In this section, we first explain how we extend $\pi_{0.6}$ to incorporate advantages, then describe the reward and value function, and then elaborate on our training and data collection processes.

A. From $\pi_{0.6}$ to $\pi^*_{0.6}$ with advantage conditioning

$\pi^*_{0.6}$ uses the same base VLA architecture as $\pi_{0.6}$, but adds advantage conditioning for RL by including a binarized improvement indicator as an additional text input. Concretely, we input "Advantage: positive" when I_t = True and "Advantage: negative" otherwise. The advantage indicator appears in the training sequence after $\hat{\ell}$ but before the discretized and continuous actions, such that only the action log-likelihoods are affected. Apart from this additional conditioning input, the VLA model is the same as detailed in the $\pi_{0.6}$ model card [76]. Following $\pi_{0.5}$ we train the continuous actions using the flow matching loss [77] and the discrete actions with autoregressive cross-entropy loss. It is possible to draw a close parallel between flow matching and diffusion (under some assumptions), and the latter in turn can be interpreted as a lower bound on the log-likelihood [78], so we can roughly motivate the sum of the log-likelihood of the discrete actions and the flow matching loss on the continuous actions as a lower bound on the overall action likelihood:

$$\log \pi_\theta(\mathbf{a}_{t:t+H}, a^\ell_{t:t+H}|I_t, \mathbf{o}_t, \ell, \hat{\ell}) \geq \mathbb{E}_{\eta,\omega}\Big[\log p_\theta(a^\ell_{t:t+H}|I_t, \mathbf{o}_t, \ell, \hat{\ell}) - \alpha_\eta \left\|\omega - \mathbf{a}_{t:t+H} - f_\theta(\mathbf{a}^{\eta,\omega}_{t:t+H}, I_t, \mathbf{o}_t, \ell, \hat{\ell})\right\|^2\Big], \tag{4}$$

with $\mathbf{a}^{\eta,\omega}_{t:t+H} = \eta\mathbf{a}_{t:t+H} + (1-\eta)\omega$, $\omega \sim \mathcal{N}(0, \mathbf{I})$ denoting the noised action, where $\eta \in [0, 1]$ is the flow matching time index and f_θ denotes the continuous outputs of the diffusion expert. α_η is a loss weighting term. Full details for the loss are provided in Appendix C. During training, we randomly omit the indicator I_t instead of tuning the loss multiplier α to allow us to either directly sample from the policy with I_t = True (which corresponds to setting $\beta = 1$ in Equation (2)), or to use both a conditional and unconditional model to implement classifier-free guidance (CFG). See Appendix E for details.

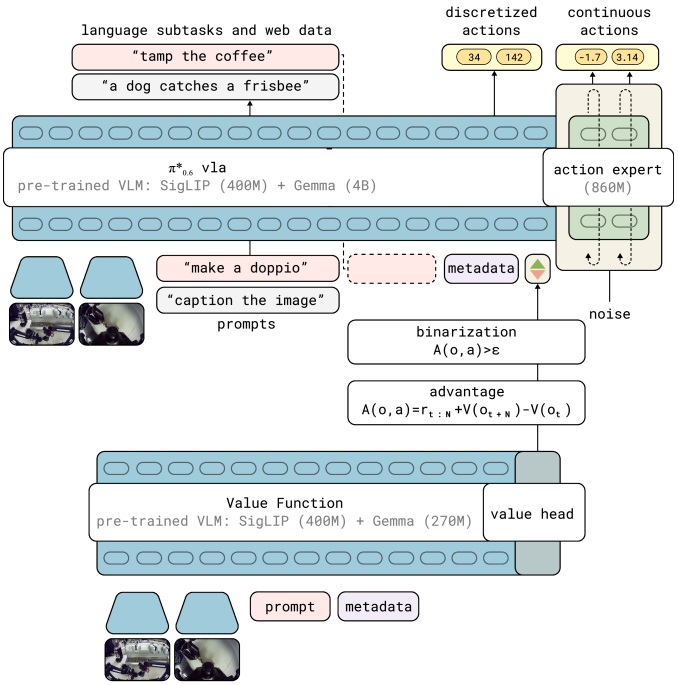


Fig. 4: **Interaction between $\pi^*_{0.6}$ and value function in RECAP.** The $\pi^*_{0.6}$ VLA uses a pre-trained VLM backbone. Training follows the KI recipe [75], with next-token prediction on many data sources in pre-training, and a flow-matching action-expert with stop gradient. Uniquely, the $\pi^*_{0.6}$ VLA is conditioned on a binarized advantage indicator, obtained from a separate value function initialized from a pre-trained but smaller VLM.

[1] Prior work [4] instead uniformly chose $\epsilon = 0$ and tuned β at test time, as in CFG. However, high CFG weights can drive the action distribution to the corners of its support (leading to aggressive behavior) and would not affect the autoregressive part of the model. We found it easier to obtain good results by instead using the threshold ϵ_ℓ to trade off regularization and optimality.

B. Reward definition and value function training

Since our aim is to develop a general and broadly applicable method for improving the reliability and speed of VLAs from real-world experience, we use a generic reward definition based on episode-level success and time to completion. For successful episodes, we have a constant time penalty for every step (encouraging the robot to perform the task as quickly as possible). For failed episodes, we provide a large penalty C_{fail} (chosen so as to ensure that failed episodes have low returns) while successes end with reward 0. Formally, for an episode of length T, we define the reward for timestep t as:

$$r_t = \begin{cases} 0 & \text{if t = T and success} \\ -C_{\text{fail}} & \text{if t = T and failure} \\ -1 & \text{otherwise.} \end{cases} \tag{5}$$

Thus, the value function learns to predict the (negative of the) number of remaining steps until success for successful episodes, and a large negative value for failed episodes. In practice, we normalize the predicted values to be between $(-1, 0)$. Since we train on diverse tasks that have very different typical lengths, we normalize the values per task based on the maximum episode length of the task.

The value function takes as input the same language inputs as the $\pi^*_{0.6}$ VLA, and uses the same architecture design, with a smaller 670M parameter VLM backbone that is also initialized from Gemma 3 (see Figure 4). To prevent overfitting, we also co-train the value function on a small mixture of multi-modal web data. Figure 3 shows visualizations of the value function on some examples of successful and failure episodes, with additional visualizations in Figure 11 in Appendix B.

C. Pre-training and learning from experience

The data mixture used in the pre-training phase of our model largely follows the recipe used by $\pi_{0.5}$ [55], with vision-language data from the web, prediction of subtasks $\hat{\ell}$, and prediction of low-level actions on a variety of tasks from many different robots. During pre-training, we first train the value function on the dataset from Section V-B and compute the per-task improvement threshold ϵ_ℓ as the 30th percentile of predicted values for task ℓ. During VLA training, the value function is run on-the-fly to estimate $A^{\pi_{\text{ref}}}(\mathbf{o}_t, \mathbf{a}_t, \ell)$ for each example, which is then used to compute I_t based on ϵ_ℓ. As described in Section V-A, I_t is provided as input to $\pi^*_{0.6}$, yielding a single pre-trained model that represents $\pi_\theta(\mathbf{a}_t | I_t, \mathbf{o}_t, \ell)$ across the full pre-training mixture. Using a relatively small VLM backbone (670M) keeps on-the-fly value inference relatively inexpensive, adding approximately 5–10% wall-clock overhead during training.

For each target task, we then perform one or more iterations of RECAP independently. To start this policy improvement loop, we first warm-start the policy by fine-tuning $\pi^*_{0.6}$ with demonstration data $\mathcal{D}_\ell$ for the target task ℓ. In this one-time initialization stage, we fix the indicator I_t to True for all training examples, such that this stage corresponds to supervised fine-tuning (SFT). The goal of this stage is to obtain a sufficiently capable initial policy π^0_ℓ for collecting useful autonomous deployment data.

Thereafter, each policy improvement iteration consists of collecting additional deployment data, fine-tuning the value function on all data collected for the task so far, and then fine-tuning the policy with updated indicators I_t. During deployment data collection, we always collect some fully autonomous rollouts. We may additionally collect rollouts monitored by an expert teleoperator, who can intervene and provide demonstrations of corrective behavior. For autonomously collected data, $I_t \in \{\text{True}, \text{False}\}$ depends on the estimated advantage as described in Section IV-C. For interventions saved by the teleoperator, we set $I_t = \text{True}$, performing "teacher forcing" analogous to the standard DAgger treatment of interventions as expert behavior [5]. Although I_t could in principle be derived from the learned advantages for these segments as well, we deliberately assign positive labels to these corrections so that the policy treats them as positive-advantage examples during training. This choice has limitations: human operators cannot guarantee globally optimal behavior, and intervening during autonomous execution can be disruptive. Thus, corrections are useful for fixing large mistakes and overcoming challenges with exploration, while autonomous rollouts remain important for learning from the policy's own successes and failures. Generalizing this step is an interesting avenue for future work.

Both the value function and policy are fine-tuned from their respective pre-trained checkpoints, rather than their checkpoints from the last iteration. In practice, we found this to be useful for avoiding drift over multiple iterations. We can repeat this process for several iterations as needed, though in practice we found that even one iteration often leads to significantly improved results. The computational cost of each fine-tuning iteration varies with task and dataset size; in our experiments, each iteration averages roughly 200 H100 GPU-hours, with runs ranging from 20 to 520 H100 GPU-hours. At test time, RECAP introduces no additional inference overhead relative to the base VLA, since the deployed policy conditions directly on the positive advantage indicator rather than querying the value function online. On an H100 machine, end-to-end policy inference latency is approximately 100-150 ms.

VI. EXPERIMENTAL EVALUATION

Our experiments focus on the following questions:

(1) How much does RECAP improve the policy?

(2) How does RECAP improve $\pi_{0.6}$ over multiple iterations?

(3) How does the advantage-conditioned policy extraction method in RECAP compare to alternatives, such as AWR [67] or PPO-based diffusion training [21, 65]?

(4) Does using autonomous rollout data help compared to just using demonstrations and corrections?

(5) Can RECAP significantly alter policy behavior with relatively little data and remove a failure mode?

We use a two-arm robot platform shown in Figure 6 and consider a set of realistic tasks depicted in Figure 2: making espresso drinks, folding diverse laundry, and assembling boxes. Each task requires multiple steps, takes 5 to 15 minutes

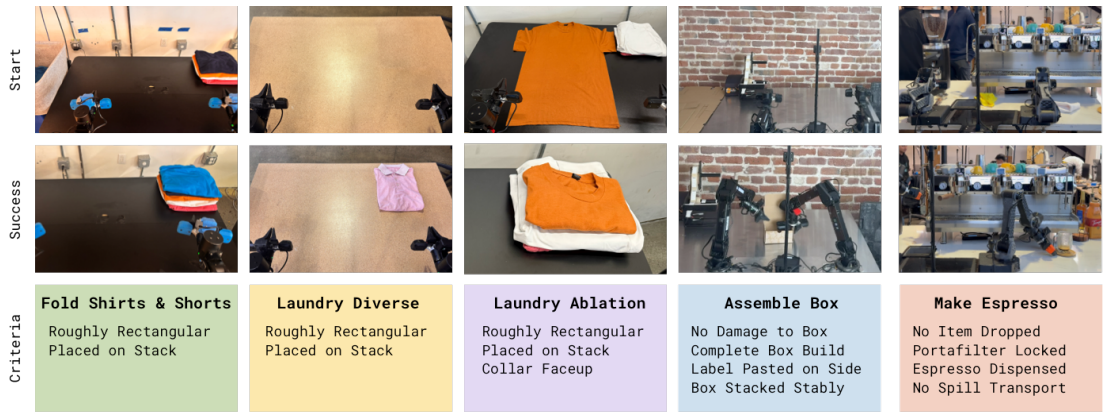


Fig. 5: **Evaluation tasks and success criteria.** We evaluate three laundry variants, box assembly, and making espresso drinks. For each rollout, human annotators assess task completion using the task-specific quality criteria shown here, such as fold quality and placement for laundry or damage and stability of stacking for box assembly. These criteria are aggregated into a binary success label for reporting success rate.

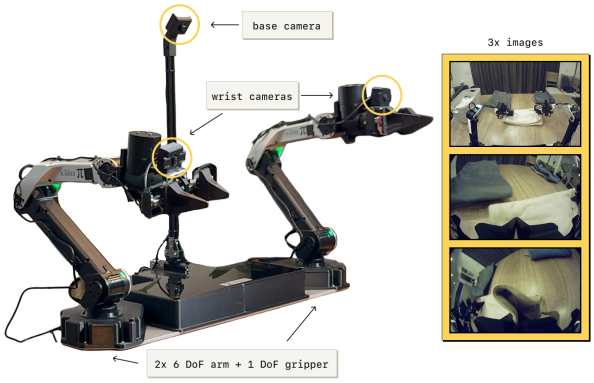


Fig. 6: **Our robot setup.** For experiments, we use a static bimanual system with two 6 DoF arms with parallel jaw grippers. The arms are controlled at 50 Hz with joint positions. Observations consist of joint and gripper positions, as well as images from three cameras: a base camera and wrist-mounted cameras.

to complete, and involves complex manipulation behaviors such as constrained forceful manipulation, pouring liquids, and manipulating cloth or cardboard. For each task, RECAP improves the policy using additional rollouts and interventions collected in the target deployment setting, and we evaluate the resulting policy in that same setting. We report *success rate* and *throughput*, where throughput is the expected number of successful task executions per hour. The evaluation tasks and success criteria are shown in Figure 5, with additional details in Appendix A.

A. Comparisons

We compare the following models:

- **Pre-trained** $\pi_{0.5}$ [55] is a baseline that represents a state-of-the-art pre-trained VLA that does not use RL.
- **Pre-trained** $\pi_{0.6}$ [76] is a baseline that uses the same architecture as $\pi^*_{0.6}$ but omits the advantage indicator I_t, and therefore does not use offline RL pre-training.
- **RL pre-trained** $\pi^*_{0.6}$ is our pre-trained model, pretrained with offline RL as described in Section V-C.
- $\pi^*_{0.6}$ **offline RL + SFT** is trained by fine-tuning the pre-trained $\pi^*_{0.6}$ model with hand-curated data. It represents the classic single-task fine-tuning recipe. During fine-tuning, the advantage values are fixed to True.
- $\pi^*_{0.6}$ **(Ours)** is our final model obtained by RECAP on the target task, fine-tuned on deployment data.
- $\pi^*_{0.6}$ **- Auton.** is an ablation model that is fine-tuned on the demonstration and intervention data only, removing autonomous deployment data.

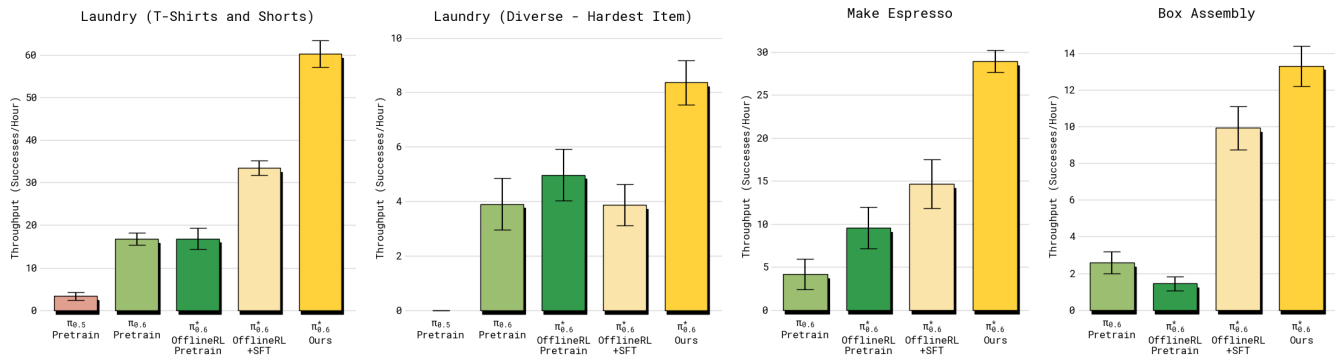
We also consider two alternative policy extraction methods from the literature as baselines, both of which use the same on-robot data as RECAP but a different policy learning method:

- **AWR**. Starting from a pre-trained $\pi_{0.6}$ model (without advantage conditioning), we fine-tune wth advantage weighted regression [67] using the advantages extracted from our value function.
- **PPO**. We implement a variant of DPPO/FPO [21, 79] in which we calculate likelihoods based on the single step diffusion objective and use an alternative definition of the PPO constraint following SPO [80] (see Appendix D).

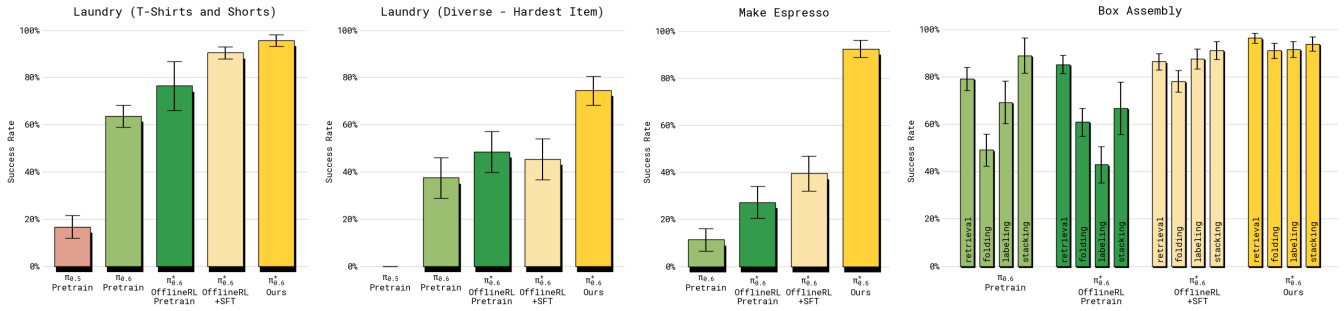
B. Quantitative results

(1) How much does RECAP *improve the policy?* We report throughput and success rates for each of the model variants on all tasks in Figures 7a and 7b. Across all tasks, the final $\pi^*_{0.6}$ model significantly improves over the base (supervised) $\pi_{0.6}$ model, the RL pre-trained $\pi^*_{0.6}$ model, and the **offline RL + SFT** $\pi^*_{0.6}$ model. Throughput more than doubles on the diverse laundry folding and espresso tasks from including on-robot data (the improvement from **offline RL + SFT** to the final $\pi^*_{0.6}$ model), and the rate of failure reduces by about a factor of two. On the T-shirts and shorts task, the success rate is already close to the maximum after the SFT phase, but throughput still increases by a significant margin.

On all tasks except diverse laundry, the success rate of the final $\pi^*_{0.6}$ model is in the 90%+ range, making it feasible to use in practice: making espresso drinks at an office or assembling boxes in a factory. For the box assembly task, Figure 7b (right) contains a breakdown of the task success over its four stages: picking up a box sheet, building the box, labeling the box,

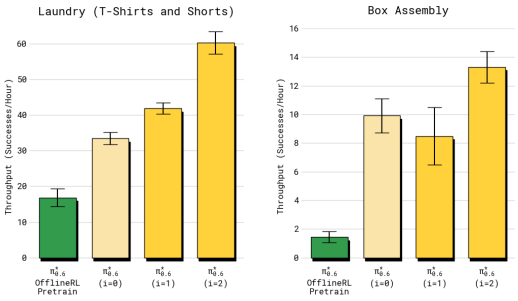


(a) **Throughput.** We report the number of successfully completed trials per hour for all tasks, with error bars denoting standard error. RECAP applied to $\pi^*_{0.6}$ yields substantial throughput gains across all tasks. We see the largest improvements on diverse laundry and espresso, where RECAP more than doubles the successful completions per hour.

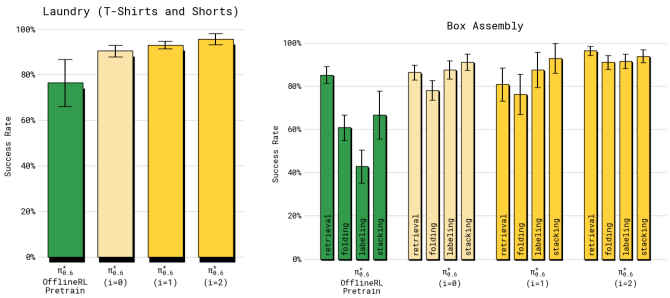


(b) **Success rates.** We report success rates with standard error for all tasks, finding that each component of RECAP consistently improves performance. The most challenging diverse laundry and espresso tasks exhibit the largest gains, with more than a $2\times$ reduction in failure rates. For box assembly, we further break down success rates per individual subtasks; RECAP achieves the most consistent and highest success across all subtasks.

Fig. 7: **Quantitative results** We show the efficacy of RECAP in improving policy performance with deployment data in terms of throughput and success rates.



(a) **Throughput.** Both tasks improve significantly in throughput as we take more iterations of RECAP, with box assembling first dropping and then improving significantly.



(b) **Success rate.** The laundry task quickly reaches the maximum success rate, while box assembly continues to improve.

Fig. 8: **Policy improvement over multiple iterations.** RECAP can be used to iteratively improve policy performance.

and placing it at an available spot in a crate. $\pi^*_{0.6}$ attains higher success rates for all of the stages compared to the other models. The majority of failures on these stages happen because the policy runs out of time.

(2) How much does RECAP *improve* $\pi^*_{0.6}$ *over multiple iterations?* We next study how RECAP improves policies through multiple iterations. For the T-shirt folding task, we collect data using only autonomous evaluations (without human corrections) to perform policy improvement over two iterations, in order to evaluate how well our method can improve the policy completely autonomously. We collect 300 trajectories on four robots in each iteration. Box assembly uses both autonomous and teleoperated intervention data, with 600 autonomous trials and 360 trials with interventions per iteration.

We see the impact on throughput and success rate over iterations in Figure 8, comparing two iterations of RECAP, denoted by $i = 1$, $i = 2$ respectively. The final iteration, labeled (Ours), corresponds to the overall best result for these tasks presented in the previous section. We also compare the initial data collection policy, which uses the offline RL pre-trained $\pi^*_{0.6}$ model with SFT fine-tuning.

For both tasks, $\pi^*_{0.6}$ improves throughput over the two iterations. In the laundry task, we see steady improvement yielding an overall 50% improvement in throughput. Here, the first iteration already raises the success rate to over 90%, while the second iteration mainly improves throughput. For the box assembly task, we see clear improvements in the success rate over both iterations. While there are still some failures (especially when placing the box on the stack at the end), the final policy achieves a success rate of about 90% both for folding the box and labeling it in the allocated time limit of 600 seconds. Here we see a $2\times$ improvement in throughput during the second iteration.

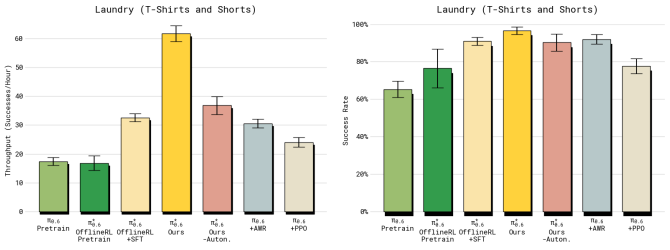


Fig. 9: **Comparison of different policy extraction methods.** RECAP achieves the highest throughput for the laundry task compared to alternative policy extraction methods like AWR and PPO. The full RECAP recipe (Ours) is better than training without autonomous data (Ours-Auton.).

Fig. 10: **Failure mode removal.** RECAP is effective at removing failure modes that would be considered non successful under the strict criteria.

(3) How does the advantage-conditioned policy extraction method in RECAP *compare to other methods?* To answer this question, we use the T-shirts and Shorts task, and to ensure a controlled comparison, we use the same data that was used to train our final model for all methods. This provides a slight advantage to the baselines, since they have access to better data that was collected while running RECAP. As shown in Figure 9, while both AWR and PPO can attain reasonable results, they both fall far short of our method, and struggle to improve over the offline RL + SFT $\pi^*_{0.6}$ model. For PPO, we had to use a small trust-region constraint ($\eta = 0.01$) to stabilize training in this off-policy setting, and while this makes training stable, the method does not achieve good performance. AWR can achieve a reasonable success rate, but leads to much slower policies with lower throughput.

(4) Does using autonomous rollout data help compared to just using demonstrations and corrections? As shown in Figure 9 we find that RECAP without autonomous data (- Auton.) performs worse than using the full combination of demonstration, corrections and autonomous data for the laundry task (T-shirts and Shorts). We find that the relative usefulness of correction vs. autonomous data is task dependent but generally observe best results when using both.

(5) Can RECAP *significantly alter policy behavior with relatively little data and remove a failure mode?* While the preceding experiments have focused on holistic end-to-end evaluations of policy performance, here we zoom in on a specific failure mode to examine whether RL training with RECAP can remove a specific mistake from the policy. For this experiment, we use a version of the laundry task with a strict success criterion, which requires the policy to fold a T-shirt with the collar centered and facing up. Each episode is initialized with a specific adversarial condition in which the shirt is placed flat on the table in such a way that the baseline offline RL + SFT policy often fails to fold it correctly. As shown in Figure 10, applying RECAP in this setting for two iterations (collecting 600 trajectories in each iteration) results in a policy that succeeds 97% of the time, and with high speed. Thus we conclude that RECAP can be effective at removing specific failure modes, even when learning entirely from autonomously collected deployment data.

VII. CONCLUSION

Training policies that can achieve human-like robustness, speed, and fluency on real-world tasks presents a major challenge in robotic learning. This paper explores how learning from experience, through a combination of DAgger-style corrections and RL, can help address this challenge. We introduce RECAP, a scalable RL-based method for training VLAs that uses advantage conditioning for policy extraction, learning from autonomous trials, reward feedback, and targeted human interventions. We evaluate a model trained with RECAP, $\pi^*_{0.6}$, on realistic tasks including making espresso drinks, folding diverse laundry, and assembling boxes. Our experiments show that RECAP improves both success rate and throughput, more than doubling throughput on some of the harder tasks while reducing failures by approximately $2\times$.

There are several directions for improvement with RECAP. First, our system is not fully autonomous: it relies on human labeling and effort for reward feedback, interventions, and episode resets. Second, our system is relatively naïve in how it approaches exploration: we rely on the stochasticity of the policy and human interventions to explore new solutions. Third, several implementation choices in RECAP—including the value discretization, advantage threshold, and intervention labeling scheme—are practical design choices that worked well empirically, but may not be optimal; more systematic sensitivity studies would be valuable in future work. Lastly, RECAP performs iterated "offline" updates (i.e., it collects a batch of data, retrains the model, and repeats), rather than running a fully online RL loop where the model is updated while data is collected. More broadly, RL with VLAs presents a number of unique challenges, from the difficulty of large-scale RL training of high capacity models to sample complexity, autonomy, and delayed feedback. While existing RL frameworks designed for smaller-scale systems or LLMs can provide a good starting point, more research will be needed to make RL a practical tool for VLA training. We hope that our work represents a meaningful step in this direction.

REFERENCES

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 1

[2] Sascha Lange, Thomas Gabel, and Martin A. Riedmiller. Batch reinforcement learning. In Marco A. Wiering and Martijn van Otterlo, editors, *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pages 45–73. Springer, 2012. doi: 10.1007/978-3-642-27645-3_2. 1

[3] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020. 1

[4] Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Diffusion guidance is a controllable policy improvement operator. *arXiv preprint*, arXiv:2505.23458, 2025. 1, 3, 4, 5, 15

[5] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, pages 627–635, 2011. 2, 6

[6] Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. Shiv: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces. In *Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 462–469, 2016. doi: 10.1109/ICRA.2016.7487175. 2

[7] Michael Laskey, Jonathan Lee, Roy Fox, Anca D. Dragan, and Ken Goldberg. Dart: Noise injection for robust imitation learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 1989–1998. PMLR, 2017.

[8] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022. 2

[9] Zheyuan Hu, Robyn Wu, Naveen Enock, Jasmine Li, Riya Kadakia, Zackory Erickson, and Aviral Kumar. Rac: Robot learning for long-horizon tasks by scaling recovery and correction. *arXiv preprint*, arXiv:2509.07953, 2025. 2

[10] Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *ICRA*, 2019. 2

[11] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016. 2

[12] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

[13] Ajay Mandlekar, Fabio Ramos, Byron Boots, Li Fei-Fei, Animesh Garg, and Dieter Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. *ICRA*, 2020.

[14] Archit Sharma, M. Ahmed Ahmed Rehaan Ahmad, and Chelsea Finn. Self-improving robots: End-to-end autonomous visuomotor reinforcement learning. In *Proceedings of the 7th Conference on Robot Learning (CoRL)*, volume 229, pages 3292–3308. PMLR, 2023.

[15] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Alan: Autonomously exploring robotic agents in the real world. In *Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3044–3050, 2023. doi: 10.1109/ICRA48891.2023.10013321.

[16] Russell Mendonca, Emmanuel Panov, Bernadette Bucher, Jiuguang Wang, and Deepak Pathak. Continuously improving mobile manipulation with autonomous real-world rl. In *Proceedings of the 8th Conference on Robot Learning (CoRL)*, pages 5204–5219, 2024.

[17] Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. Serl: A software suite for sample-efficient robotic reinforcement learning, 2024.

[18] Lars Ankile, Zhenyu Jiang, Rocky Duan, Guanya Shi, Pieter Abbeel, and Anusha Nagabandi. Residual off-policy rl for finetuning behavior cloning policies. *arXiv preprint arXiv:2509.19301*, 2025.

[19] Thomas Lampe, Abbas Abdolmaleki, Sarah Bechtle, Sandy H. Huang, Jost Tobias Springenberg, Michael Bloesch, Oliver Groth, Roland Hafner, Tim Hertweck, Michael Neunert, Markus Wulfmeier, Jingwei Zhang, Francesco Nori, Nicolas Heess, and Martin Riedmiller. Mastering stacking of diverse shapes with large-scale iterative reinforcement learning on real robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7772–7779, 2024. doi: 10.1109/ICRA57147.2024.10610297. 2

[20] Perry Dong, Suvir Mirchandani, Dorsa Sadigh, and Chelsea Finn. What matters for batch online reinforcement learning in robotics? *arXiv preprint*, arXiv:2505.08078, 2025. 2

[21] Allen Z. Ren, Justin Lidard, Lars Lien Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion Policy Policy Optimization. In *Proceedings of the 2025 International Conference on Learning Representations (ICLR)*, 2025. 6, 7, 15

[22] Kun Lei, Huanyu Li, Dongjie Yu, Zhenyu Wei, Lingxiao Guo, Zhennan Jiang, Ziyu Wang, Shiyu Liang, and Huazhe Xu. Rl-100: Performant robotic manipulation with real-world reinforcement learning. *arXiv preprint*, arXiv:2510.14830, 2025. 2

[23] Dmitry Kalashnkov, Jake Varley, Yevgen Chebotar, Ben Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv*, 2021. 2

[24] Abhishek Gupta, Justin Yu, Tony Z. Zhao, Vikash Kumar, Aaron Rovinsky, Kelvin Xu, Thomas Devlin, and Sergey Levine. Reset-free reinforcement learning via multi-

task learning: Learning dexterous manipulation behaviors without human intervention. In *Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6664–6671, 2021. 2

[25] Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. Robocat: A self-improving foundation agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023. 2

[26] Aviral Kumar, Anikait Singh, Frederik Ebert, Mitsuhiko Nakamoto, Yanlai Yang, Chelsea Finn, and Sergey Levine. Pre-training for robots: Offline reinforcement learning enables learning new tasks from a handful of trials. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023. doi: 10.15607/RSS.2023.XIX.019.

[27] Jingyun Yang, Max Sobol Mark, Brandon Vu, Archit Sharma, Jeannette Bohg, and Chelsea Finn. Robot fine-tuning made easy: Pre-training rewards and policies for autonomous real-world reinforcement learning. In *Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024. doi: 10.1109/ICRA57147.2024.10610421. 2

[28] Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive post-training for vision-language-action models. *arXiv preprint*, arXiv:2505.17016, 2025. 2

[29] Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning. *arXiv preprint*, arXiv:2505.18719, 2025.

[30] Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What can rl bring to vla generalization? an empirical study. *arXiv preprint*, arXiv:2505.19789, 2025.

[31] Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, Tiejun Huang, Yu Wang, and Chao Yu. $\pi_{\texttt{rl}}$: Online rl fine-tuning for flow-based vision-language-action models. *arXiv preprint*, arXiv:2510.25889, 2025.

[32] Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, Dehui Wang, Dingxiang Luo, Yuchen Fan, Youbang Sun, Jia Zeng, Jiangmiao Pang, Shanghang Zhang, Yu Wang, Yao Mu, Bowen Zhou, and Ning Ding. Simplevla-rl: Scaling vla training via reinforcement learning. *arXiv preprint*, arXiv:2509.09674, 2025. 2

[33] Yanjiang Guo, Jianke Zhang, Xiaoyu Chen, Xiang Ji, Yen-Jen Wang, Yucheng Hu, and Jianyu Chen. Improving vision-language-action model with online reinforcement learning. *arXiv preprint*, arXiv:2501.16664, 2025. 2

[34] Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Yuqi Xie, Fengyuan Hu, Jimmy Wu, Zhengyi Luo, Linxi "Jim" Fan, Guanya Shi, and Yuke Zhu. Self-improving vision-language-action models with data generation via residual rl, 2025. 2

[35] Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. Conrft: A reinforced fine-tuning method for vla models via consistency policy. *arXiv preprint arXiv:2502.05450*, 2025. 2

[36] Max Sobol Mark, Tian Gao, Georgia Gabriela Sampaio, Mohan Kumar Srirama, Archit Sharma, Chelsea Finn, and Aviral Kumar. Policy-agnostic rl: Offline rl and online rl fine-tuning of any class and backbone. *arXiv preprint*, arXiv:2412.06685, 2024. 2

[37] Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. In *Conference on Robot Learning*, pages 4996–5013. PMLR, 2025.

[38] Yang Zhang, Chenwei Wang, Ouyang Lu, Yuan Zhao, Yunfei Ge, Zhenglong Sun, Xiu Li, Chi Zhang, Chenjia Bai, and Xuelong Li. Align-then-steer: Adapting the vision-language action models through unified latent guidance. *arXiv preprint arXiv:2509.02055*, 2025. 2

[39] Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In *Proceedings of the 9th Conference on Robot Learning (CoRL)*, 2025. 2

[40] Charles Xu, Qiyang Li, Jianlan Luo, and Sergey Levine. Rldg: Robotic generalist policy distillation via reinforcement learning. *arXiv preprint arXiv:2412.09858*, 2024. 2

[41] Dongchi Huang, Zhirui Fang, Tianle Zhang, Yihang Li, Lin Zhao, and Chunhe Xia. Co-rft: Efficient fine-tuning of vision-language-action models through chunked offline reinforcement learning. *arXiv preprint*, arXiv:2508.02219, 2025. 2

[42] Zijian Zhang, Kaiyuan Zheng, Zhaorun Chen, Joel Jang, Yi Li, Siwei Han, Chaoqi Wang, Mingyu Ding, Dieter Fox, and Huaxiu Yao. Grape: Generalizing robot policy via preference alignment. *arXiv preprint*, arXiv:2411.19309, 2024. 2

[43] Shaopeng Zhai, Qi Zhang, Tianyi Zhang, Fuxian Huang, Haoran Zhang, Ming Zhou, Shengzhe Zhang, Litao Liu, Sixu Lin, and Jiangmiao Pang. A vision-language-action-critic model for robotic real-world reinforcement learning. *arXiv preprint*, arXiv:2509.15937, 2025. 2, 3

[44] Seyed Kamyar Ghasemipour, Ayzaan Wahid, Jonathan Tompson, Pannag Sanketi, and Igor Mordatch. Self-improving embodied foundation models. *arXiv preprint*, arXiv:2509.15155, 2025. 2

[45] Jürgen Schmidhuber. Reinforcement learning upside down: Don't predict rewards — just map them to actions. *arXiv preprint*, arXiv:1912.02875, 2019. 3, 4

[46] Aviral Kumar, Xue Bin Peng, and Sergey Levine. Reward-conditioned policies. *CoRR*, abs/1912.13465, 2019. 4

[47] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems (NeurIPS) 34*, 2021.

[48] David Brandfonbrener, Alberto Bietti, Jacob Buckman, Romain Laroche, and Joan Bruna. When does return-conditioned supervised learning work for offline reinforcement learning? In *Advances in Neural Information Processing Systems (NeurIPS) 35*, 2022. 4

[49] Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning? In *Proceedings of the 10th International Conference on Learning Representations (ICLR)*, 2022.

[50] Hiroki Furuta, Yusuke Matsuo, and Shixiang Shane Gu. Generalized decision transformer for offline hindsight information matching. In *Proceedings of the 10th International Conference on Learning Representations (ICLR)*, 2022.

[51] Taku Yamagata, Ahmed Khalil, and Raúl Santos-Rodríguez. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline rl. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pages 38989–39007. PMLR, 2023.

[52] Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pages 27042–27059. PMLR, 2022.

[53] Jakub Grudzien Kuba, Pieter Abbeel, and Sergey Levine. Advantage-conditioned diffusion: Offline rl via generalization. 2023.

[54] Yueh-Hua Wu, Xiaolong Wang, and Masashi Hamaya. Elastic decision transformer. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*, 2023. doi: 10.5555/3666122.3666936. 3

[55] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y Galliker, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *9th Annual Conference on Robot Learning*, 2025. 3, 5, 6, 7

[56] Lin Shao, Toki Migimatsu, Qiang Zhang, Kaiyuan Yang, and Jeannette Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. In *Proceedings of Robotics: Science & Systems (RSS)*, 2020. doi: 10.15607/RSS.2020.XVI.082. 3

[57] Annie S. Chen, Suraj Nair, and Chelsea Finn. Learning generalizable robotic reward functions from "in-the-wild" human videos. In *Proceedings of Robotics: Science & Systems (RSS) 2021*, 2021.

[58] Suraj Nair, Eric Mitchell, Kevin Chen, Brian Ichter, Silvio Savarese, and Chelsea Finn. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, volume 164 of *Proceedings of Machine Learning Research*, pages 1303–1315. PMLR, 2022.

[59] Sumedh A. Sontakke, Jesse Zhang, Sébastien M.R. Arnold, Karl Pertsch, Erdem Bıyık, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. Roboclip: One demonstration is enough to learn robot policies. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[60] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, Brian Ichter, Ted Xiao, Peng Xu, Andy Zeng, Tingnan Zhang, Nicolas Heess, Dorsa Sadigh, Jie Tan, Yuval Tassa, and Fei Xia. Language to rewards for robotic skill synthesis. In *Proceedings of the 7th Conference on Robot Learning (CoRL)*, volume 229 of *Proceedings of Machine Learning Research*, pages 374–404. PMLR, 2023.

[61] Jiahui Zhang, Yusen Luo, Abrar Anwar, Sumedh Anand Sontakke, Joseph J. Lim, Jesse Thomason, Erdem Bıyık, and Jesse Zhang. Rewind: Language-guided rewards teach robot policies without new demonstrations. In *Proceedings of the 9th Conference on Robot Learning (CoRL)*, 2025.

[62] Minttu Alakuijala, Reginald McLean, Isaac Woungang, Nariman Farsad, Samuel Kaski, Pekka Marttinen, and Kai Yuan. Video-language critic: Transferable reward functions for language-conditioned robotics. *Transactions on Machine Learning Research*, 2025:1–22, 2025. 3

[63] Yecheng Jason Ma, William Liang, Vaidehi Som, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. Liv: Language-image representations and rewards for robotic control. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023. 3

[64] Yecheng Jason Ma, Joey Hejna, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, Osbert Bastani, Dinesh Jayaraman, Wenhao Yu, Tingnan Zhang, Dorsa Sadigh, and Fei Xia. Vision language models are in-context value learners. In *Proceedings of the 13th International Conference on Learning Representations (ICLR)*, 2025. 3

[65] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 3, 4, 6, 15

[66] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*,

2018. 3

[67] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019. 4, 6, 7

[68] Peter Dayan and Geoffrey E. Hinton. Using expectation-maximization for reinforcement learning. *Neural Computation*, 9(2):271–278, 1997. doi: 10.1162/neco.1997.9.2.271.

[69] Jan Peters, Katharina Mülling, and Yasemin Altün. Relative entropy policy search. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, page 1607–1612. AAAI Press, 2010. 3

[70] Qing Wang, Jiechao Xiong, Lei Han, peng sun, Han Liu, and Tong Zhang. Exponentially weighted imitation learning for batched historical data. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, 2018. 3

[71] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pages 449–458. PMLR, 2017. 4

[72] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *ICML*, 2018. 4

[73] Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, and Nando de Freitas. Critic regularized regression. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7768–7778, 2020. 4

[74] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. 4

[75] Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*, 2025. 5

[76] Physical Intelligence Team. $\pi_{0.6}$ model card. Physical Intelligence model card PDF, 2025. 5, 7

[77] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. 5, 14

[78] Diederik Kingma and Ruiqi Gao. Understanding diffusion objectives as the elbo with simple data augmentation. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 65484–65516, 2023. 5, 15

[79] David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients, 2025. 7, 14, 15

[80] Zhengpeng Xie, Qiang Zhang, Fan Yang, Marco Hutter, and Renjing Xu. Simple policy optimization. In *Forty-second International Conference on Machine Learning (ICML)*, 2025. 7, 15

[81] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. 14