MAKING EVERY INTERACTION COUNT: EFFICIENT REAL-WORLD RL FOR ROBOT POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-world reinforcement learning (RL) is crucial for adapting robotic control policies to physical environments. Yet, it faces several challenges: costly environment interactions and human supervision, sparse rewards in long-horizon tasks, and underutilized suboptimal experience from autonomous policy rollouts. To address this, we introduce *Detect, Evaluate, and Compose* (***DEC***), an RL framework that minimizes human intervention while enabling efficient autonomous policy improvement. *DEC* first trains a dynamics-aware discriminator to detect failures and determine when to trigger human takeover and resume autonomous execution. It then evaluates a robust value function to tackle the sparse reward challenge by adopting trajectory-held-out bootstrapping and discounted partial returns. For policy optimization, *DEC* learns complementary positive and negative sub-policies to exploit both successful and suboptimal experience, and compose them at inference time to enable controllable action generation. Across simulated manipulation environment and long-horizon, dexterous bimanual tasks in the real world, *DEC* consistently outperforms existing RL and policy fine-tuning methods with limited human intervention, demonstrating that effective real-world RL can make every interaction count. More results can be seen at project page.

1 INTRODUCTION

Recent robotic foundation models (Intelligence et al., 2025b; Zheng et al., 2026b; Yuan et al., 2026; NVIDIA et al., 2025; Team et al., 2025) have demonstrated great potential for generalist policies that understand language, perceive the environment, and perform diverse manipulation tasks. However, deploying these models in novel environments and tasks still requires post-training to achieve high precision and reliability, with reinforcement learning (RL) playing a central role in closing this last-mile deployment gap (Intelligence et al., 2025a; Xu et al., 2026). Although simulation-based RL offers scalable interaction, it cannot fully capture real-world dynamics, contact-rich interactions, sensing noise, and environmental variations. Real-world RL is therefore essential for adapting policies to the physical deployment setting, but nevertheless remains challenging: both policy rollouts and human interventions are costly, imposing stringent demands on sample efficiency; dense rewards are difficult to design for complex manipulation tasks, while sparse rewards and long horizons make reliable value estimation challenging; and suboptimal experiences generated during policy rollouts are often underutilized, further limiting learning efficiency.

Although RL offers potential to improve policies and even surpass human-level performance through autonomous environment interaction, this autonomy comes at a cost: RL remains notoriously sample-inefficient and difficult to optimize on real robots (Ibarz et al., 2021), especially for large generative policies. In practice, real-world RL can stall at some bottleneck states, fail to discover good behavior and make further progress. At such states, even a small amount of human supervision can offer valuable corrective signals and unlock further learning (Luo et al., 2025) (illustrated in Figure 1). This is where behavior cloning (BC) shines. By learning directly from demonstrated actions, BC-based methods such as supervised finetuning (SFT) (Kim et al., 2025) and DAgger (Ross et al., 2011) provide stable learning signals, avoiding the costly trial-and-error to discover these behaviors. However, the cost is unneglectable, requiring intensive human effort throughout the training process (Hoque et al., 2021). This motivates human-in-the-loop RL, which combines selective human correction with autonomous learning (Luo et al., 2024; 2025). Existing methods either regularize RL with naïve imitation objectives (Chen et al., 2025), learn auxiliary actor or residual policies that may

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

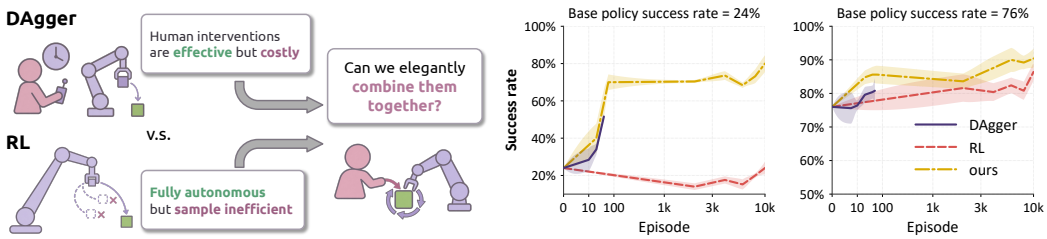

Figure 1: **Left: Comparison for DAgger and RL methods.** DAgger uses effective but costly human interventions, while RL is autonomous but sample-inefficient. We seek to combine the strengths of both paradigms. **Right: Methods comparison under different base-policy qualities.** Simulation task with base policy success rates of $24\%/76\%$ and intervention budgets of 5 min. RL uses the RL-Token (Xu et al., 2026) training scheme without interventions. Human guidance is especially helpful from weak initialization, while stronger policies benefit more from autonomous RL. Our method combines the merits of both paradigms effectively.

override competent base actions (Xu et al., 2026; Dong et al., 2026; Liu et al., 2026; Zheng et al., 2026a), or derive binary advantage labels that discard fine-grained quality differences and amplify thresholding errors (Intelligence et al., 2025a; Xiao et al., 2026a). These limitations prevent human intervention and autonomous rollouts from being fully and reliably exploited.

The above observations suggest that effective real-world RL hinges on two questions: *when* should human supervision be introduced during interaction, and *how* can every collected transition—including suboptimal ones—contribute to policy improvement? To address both, we propose *Detect, Evaluate, and Compose* (***DEC***), a real-world RL framework that minimizes human intervention while fully exploiting past experiences. First, a dynamics-aware discriminator anticipates failures and enables timely bidirectional handoffs between autonomous execution and human control. Second, *DEC* evaluates a robust value function to tackle the sparse reward challenge by introducing a set of critical designs, including trajectory-held-out bootstrapping, discounted partial return estimates, and intrinsic reward signals from the learned discriminator. Finally, *DEC* trains complementary positive and negative sub-policies from preferred and suboptimal behaviors, allowing human interventions, successful rollouts, and suboptimal experience to jointly improve the base policy. These two sub-policies are composed through classifier-free guidance (CFG) style inference (Ho & Salimans, 2022; Frans et al., 2025; Liang et al., 2026), enabling controllable action generation. Together, these components reduce unnecessary human effort and extract useful learning signals from every real-world interaction. Extensive experiments on long-horizon dexterous real-world bimanual tasks and ROBOSUITE (Zhu et al., 2020) simulation tasks demonstrate the effectiveness of *DEC*, which consistently outperforms existing RL and DAgger-based policy finetuning methods while requiring minimal human intervention.

2 PRELIMINARIES

Notations. We formulate the robotic control problem as a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and primitive action spaces, $\mathcal{P}(\cdot \mid \mathbf{s}, \mathbf{a})$ denotes the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in (0, 1)$ is the discount factor. We treat the robot's visual and proprioceptive observation context as the Markov state, with task-language conditioning left implicit. During training, we allow a human operator to temporarily take over control and provide corrective actions (Luo et al., 2025; Xu et al., 2026). Our goal is to improve a pretrained policy to maximize its expected discounted return $J(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)]$.

Reinforcement learning with action chunking. Modern robot policies often predict a sequence of future actions at each decision step, motivating action chunking as the basic decision unit for RL finetuning. We denote an action chunk of horizon H as $\mathbf{a}_t = (a_t, \ldots, a_{t+H-1}) \in \mathbb{R}^{H \times d}$. Executing the chunk induces an H-step transition from $\mathbf{s}_t$ to $\mathbf{s}_{t+H}$, accumulating discounted reward $\sum_{i=0}^{H-1} \gamma^i r(\mathbf{s}_{t+i}, \mathbf{a}_{t+i})$ with continuation discount γ^H (Li et al., 2026). Let $V^\pi(\mathbf{s})$ and $Q^\pi(\mathbf{s}, \mathbf{a})$ denote the state- and action-value functions when following π from a chunk decision boundary. We extend the standard advantage definition $A^\pi = Q^\pi - V^\pi$ to the action-chunk setting as:

$$A^\pi_H(\mathbf{s}_t, \mathbf{a}_{t:t+H-1}) = \mathbb{E}_{\mathbf{s}_{t+H} \sim \mathcal{P}} \left[\sum_{i=0}^{H-1} \gamma^i r(\mathbf{s}_{t+i}, \mathbf{a}_{t+i}) + \gamma^H V^\pi(\mathbf{s}_{t+H}) \right] - V^\pi(\mathbf{s}_t), \tag{1}$$

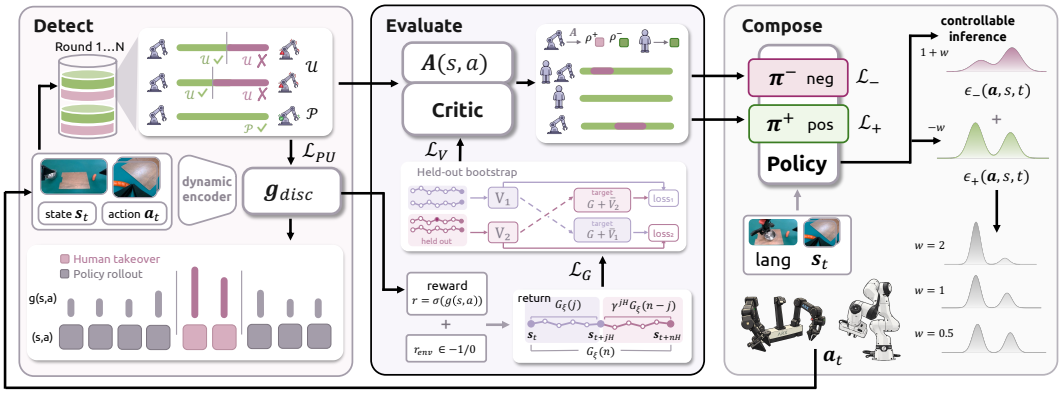

Figure 2: Illustration of ***Detect, Evaluate, and Compose*** (*DEC*) framework. In **Detect** stage, a discriminator trained under a non-negative PU-learning objective with dynamics-aware representations anticipates the eventual success likelihood of each $(\mathbf{s}, \mathbf{a})$ pair and triggers human intervention when success is unlikely. In **Evaluate** stage, we augment sparse outcome reward with discriminator feedback, fit horizon-conditioned partial returns, and learn values with trajectory-held-out bootstrapping. In **Compose** stage, score functions are learned separately from desirable and suboptimal behaviors and flexibly combined at inference time. The updated policy then collects new trajectories, which are used to refine all three stages in the next iteration.

where the expectation is over the dynamics $\mathcal{P}$ induced by executing the action chunk from $\mathbf{s}_t$.

Diffusion/flow-based robotic foundation models. Diffusion and flow matching have become popular generative policy classes for robotic control due to their strong expressiveness in modeling complex action distributions (Chi et al., 2025; Intelligence et al., 2025b; Zheng et al., 2025; Tan et al., 2026). Action generation can be viewed as a state-conditional generation process that transforms a simple source distribution, typically a standard Gaussian, into the target action distribution. For concreteness, we use a diffusion policy with noise prediction as an example. Given $(\mathbf{s}, \mathbf{a}) \sim \mathcal{D}$ and Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, a noisy action is constructed as $\mathbf{a}_\tau = \alpha_\tau \mathbf{a} + \sigma_\tau \boldsymbol{\epsilon}$, where τ denotes the generative timestep and α_τ, σ_τ specify the noise schedule. The policy is trained to predict the injected noise via

$$\mathcal{L}_{\mathrm{BC}}(\theta) = \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D},\, \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I}),\, \tau} \left[\left\| \boldsymbol{\epsilon} - \epsilon_\theta(\mathbf{a}_\tau, \mathbf{s}, \tau) \right\|_2^2 \right]. \tag{2}$$

Flow matching and other generative parameterizations follow an analogous formulation with different probability paths and prediction targets.

3 Methodology

Figure 2 provides an overview illustration of our proposed *DEC* framework, which contains three stages of algorithmic designs—**Detect**, **Evaluate**, and **Compose**. In the **Detect** stage, a learned discriminator identifies the right moments for human intervention, allowing limited human supervision to be allocated where it is most valuable and yielding higher-quality training data. The **Evaluate** stage then learns a robust critic that can reliably assess sample quality under long-horizon, sparse-reward settings, providing accurate signals for policy improvement. Finally, the **Compose** stage trains a pair of positive and negative sub-policies on desirable and suboptimal samples, and composes them into an execution-time policy to achieve controllable action generation. Together, these designs provide a practical path toward efficient real-world robotic RL.

3.1 Detect: Failure Anticipation with Dynamics-Aware Discriminator

To maximize the utility of limited human supervision, interventions should occur only when they are likely to prevent rollout failure. If the policy is likely to succeed after an $(\mathbf{s}, \mathbf{a})$ pair, autonomous execution is sufficient; otherwise, a timely takeover can substantially increase the chance of task success (Hu et al., 2026). We thus train a discriminator to anticipate the eventual outcome following each $(\mathbf{s}, \mathbf{a})$ and trigger human intervention in advance when the predicted success likelihood is low, as well as return control to the policy once the likelihood recovers (illustrated in Figure 3)

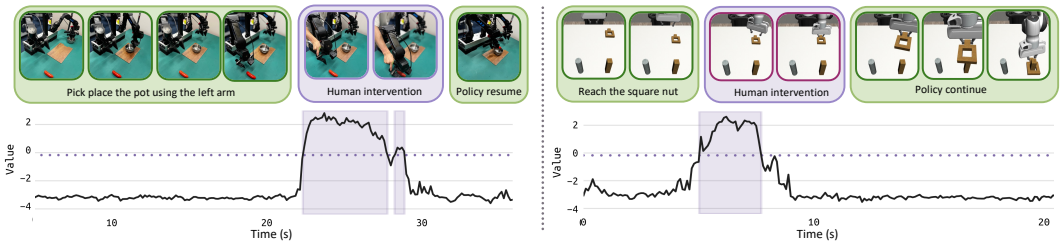

Figure 3: **Visualization of the failure discriminator.** We visualize the discriminator failure score on both simulation and real-world tasks. When the discriminator anticipates failure, the failure score rises above the threshold (dotted line), triggering human takeover. After a few corrective human actions, the score drops back below the threshold, allowing the policy to resume autonomous execution. The shaded regions indicate human intervention, and images on top show the corresponding interaction segments.

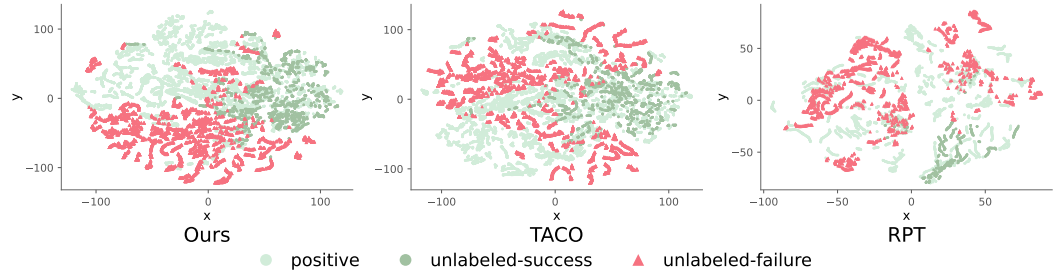

Figure 4: **t-SNE visualization of state-action representations.** Dynamics-aware pretraining yields clearer separation between success-like and failure-related $(\mathbf{s}, \mathbf{a})$ feature pairs than TACO (Zheng et al., 2023) and RPT (Dong et al., 2025). Light green denotes the positive set, while dark green and red denote the success- and failure-related subsets of the unlabeled set, respectively. Further details are provided in Appendix D.2.

PU learning-style discriminator training. In robot trajectories, task outcomes are typically labeled only at the trajectory level, while the contribution of each state–action pair to the eventual success or failure remains unknown. To address this ambiguity, we formulate discriminator training as non-negative positive–unlabeled (nnPU) learning (Kiryo et al., 2017), rather than relying on distances to expert data (Yu et al., 2026) or uncertainty estimates from conformal prediction (Xu et al., 2025; Gu et al., 2026). This formulation is particularly suitable because failed trajectories may still contain state–action pairs that would lead to successful execution.

Let $\mathcal{P}$ denote the *positive* set of state–action pairs collected from successful trajectories, and $\mathcal{U}$ denote the *unlabeled* set collected from failed trajectories. We train a discriminator $g(\mathbf{s}, \mathbf{a})$ to establish a robust decision boundary among positive samples, as well as potentially good and bad samples within failed trajectories. The learning objective is given as:

$$\mathcal{L}_{\text{nnPU}} = \eta \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{P}} \left[\ell(g(\mathbf{s},\mathbf{a}), +1)\right] + \left[\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{U}} \left[\ell(g(\mathbf{s},\mathbf{a}), -1)\right] - \eta \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{P}} \left[\ell(g(\mathbf{s},\mathbf{a}), -1)\right]\right]_+ \quad (3)$$

where $[x]_+ = \max(x, 0)$, $\ell(x, y) = \log(1 + \exp(-xy))$ is the logistic loss for $y \in \{+1, -1\}$, and η denotes the prior proportion of positive pairs in $\mathcal{U}$. The first term encourages positive pairs to receive positive scores, while the second estimates the negative risk from the unlabeled set. The last correction term subtracts the contribution of positive pairs embedded in $\mathcal{U}$, preventing potentially good $(\mathbf{s}, \mathbf{a})$ pairs in failed trajectories from being incorrectly treated as failures (Kiryo et al., 2017).

Dynamics-aware state-action representation pre-training. Due to the stringent sample efficiency requirement in real-robot RL, as well as the high dimensionality of robotic state and action data, we need some sort of representation pre-training to enable reliable discriminator learning under limited real-world interaction data, i.e., $g(\mathbf{s}, \mathbf{a}) := g(\phi(\mathbf{s}, \mathbf{a}))$. After experimenting with diverse representation pre-training methods (see Section 4.2 and Figure 4), we find that pre-training a state-action representation $\phi(\mathbf{s}_t, \mathbf{a}_t)$ by fitting a dynamics model f_ψ on successful trajectories provides the most effective solution, which has the following pre-training objective:

$$\mathcal{L}_{\text{dyn}}(\phi, \psi) = \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+H})\sim\mathcal{D}^+} \left[\left\| f_\psi(\phi(\mathbf{s}_t, \mathbf{a}_t)) - \mathbf{s}_{t+H} \right\|_2^2\right]. \quad (4)$$

Empirically, the effectiveness of dynamics-aware representations could be attributed to the preservation of predictive information for the resulting state, making high- and low-quality $(\mathbf{s}, \mathbf{a})$ pairs

substantially easier to separate, therefore yielding more discriminative representations (Huang et al., 2026). Figure 4 further supports this. Intuitively, such a property could stem from the fact that high- and low-quality transitions tend to produce systematically different state changes, which are explicitly captured by the dynamics modeling.

3.2 Evaluate: Robust Critic Learning for Long-Horizon, Sparse Reward Tasks

While the discriminator enables efficient human intervention, alternating between human and policy control produces more heterogeneous experience, requiring a critic that generalizes across mixed-quality samples. This is challenging in real-world RL for two reasons. First, limited trajectories can cause the critic to "memorize trajectories": a failed rollout may receive uniformly low values even when some intermediate states still admit successful continuations. Second, long-horizon tasks make long-range value propagation inefficient. Sparse outcome rewards exacerbate both issues.

Trajectory-held-out bootstrapping. To address the first issue, we partition trajectories into M disjoint splits and train M independent value networks, where the state-value function V_{ϕ_m} is trained only on data outside $\mathcal{D}_m$. Each sample from $\mathcal{D}_j$ is therefore bootstrapped by its corresponding held-out value network $V_{\bar{\phi}_j}$ and used to train the remaining networks. Since the bootstrap target for each sample is produced using a value network that has never been trained on its corresponding split, this construction reduces trajectory-specific bias and greatly stabilizes value learning. In practice, our proposed trajectory-held-out bootstrapping is instantiated upon the commonly used expectile regression objective in offline RL (Kostrikov et al., 2021) to maximally utilize all the limited real-robot data collected during each round of environment interaction:

$$\mathcal{L}_V = \sum_{m=1}^{M} \sum_{j \neq m} \mathbb{E}_{\mathcal{D}_j} \left[L_2^{\tau_e} \left(\mathrm{sg}\left[\mathcal{T}_t\left(V_{\bar{\phi}_j}(\mathbf{s}_t) \right) \right] - V_{\phi_m}(\mathbf{s}_t) \right) \right], \qquad L_2^{\tau_e}(u) = \left| \tau - \mathbf{1}_{\{u<0\}} \right| u^2. \tag{5}$$

Here, sg denotes stop-gradient, and $\mathcal{T}_t(V_{\bar{\phi}_j})$ denotes a general bootstrapping target.

Robust bootstrapping target estimation. To address the long-horizon value propagation challenge, a classical solution is multi-step TD, such as TD(λ) (Sutton, 1988), which constructs bootstrap targets over different backup horizons. However, its returns are directly computed from sampled reward sequences and can exhibit high variance with limited real-world trajectories. Some recent studies improve this by bootstrapping discounted partial returns along the observed trajectory, i.e., $G_{i:j} \triangleq \sum_{u=i}^{j-1} \gamma^{u-i} r_u$ (Abraham et al., 2026; Zheng et al., 2026d), which has shown improved value estimation quality for long-horizon tasks. To extend it to the action chunking setting, we introduce a chunked version of discounted partial return, calculated as $G^H_{t:t+nH} \triangleq \sum_{i=0}^{n-1} (\gamma^H)^i \sum_{k=0}^{H-1} \gamma^k r_{t+iH+k}$, which computes the cumulative partial return spanning n action chunks. Moreover, to achieve a certain level of generalization beyond the limited real-robot data, we instead parameterize it with a neural network $G_\xi(\mathbf{s}_t, \mathbf{s}_{t+nH}, n)$, which estimates the expected discounted partial return between $\mathbf{s}_t$ and $\mathbf{s}_{t+nH}$ over n action chunks. We train G_ξ by regressing to the observed discounted partial return $G^H_{t:t+nH}$ while enforcing temporal compositional consistency:

$$\begin{aligned} \mathcal{L}_G(\xi) = \mathbb{E}_{n \sim \mathrm{Unif}\{1,\cdots,N\}, j \sim \mathrm{Unif}\{1,\cdots,N-1\}, \mathbf{s}_t \sim \mathcal{D}} \Big[& \left(G_\xi(\mathbf{s}_t, \mathbf{s}_{t+nH}, n) - G^H_{t:t+nH} \right)^2 \\ & + \lambda_{\mathrm{comp}} \mathbf{1}_{\{n \geq 2\}} \left[G_\xi(\mathbf{s}_t, \mathbf{s}_{t+nH}, n) - \left(G_\xi(\mathbf{s}_t, \mathbf{s}_{t+jH}, j) + (\gamma^H)^j G_\xi(\mathbf{s}_{t+jH}, \mathbf{s}_{t+nH}, n-j) \right) \right]^2 \Big] \end{aligned} \tag{6}$$

To further stabilize value estimation, in practice, we jointly supervise the value function using target returns from multiple backup horizons. The final bootstrapping target is computed as follows. Further details are provided in Appendix C.3.

$$\mathcal{T}_t(V_{\bar{\phi}_j}) = \frac{1}{N} \sum_{n=1}^{N} \left[G_\xi(\mathbf{s}_t, \mathbf{s}_{t+nH}, n) + \gamma^{nH} V_{\bar{\phi}_j}(\mathbf{s}_{t+nH}) \right], \tag{7}$$

Discriminator as intrinsic reward. The discriminator's ability to anticipate the eventual success of each $(\mathbf{s}, \mathbf{a})$ pair makes it an ideal choice to serve as an intrinsic reward, which can help to further alleviate the long-horizon, sparse reward issue. Similar to Tsao et al. (2026), we also augment the

sparse outcome reward with the dense discriminator score, rewarding state-action pairs that are more likely to lead to eventual success:

$$r_t = r_t^{\text{env}} + \lambda_{\text{disc}} \sigma(g(\phi(\mathbf{s}_t, \mathbf{a}_t))) \,, \tag{8}$$

where r_t^{env} is the sparse task reward, λ_{disc} is the scale parameter, and $\sigma(\cdot)$ is the sigmoid function.

3.3 COMPOSE: POSITIVE-NEGATIVE POLICY OPTIMIZATION AND COMPOSITION

Conventional robotic policy learning typically relies on a critic to selectively drive the policy to learn from high-quality samples, leaving low-quality samples with little contribution to policy learning and resulting in poor data utilization. We instead aim to exploit both high- and low-quality data while enabling flexible control over their influence at inference time.

Compositional policy optimization. To fully utilize the suboptimal rollout data, we draw inspiration from some recent works that decompose the optimal policy into learning a pair of positive and negative sub-policies, which enables more effective utilization of both preferred and disfavored samples (Liang et al., 2026; Chen et al., 2026; Zheng et al., 2026c). This allows learning complementary behavior distributions whose contrast can flexibly steer action generation via the classifier-free guidance (CFG) mechanism for diffusion/flow-based policies (Frans et al., 2025; Liang et al., 2026). Specifically, we follow the construction in DIPOLE (Liang et al., 2026), which optimizes the policy with a greedified KL-regularized RL objective:

$$\max_{\pi} \; \mathbb{E}_{\mathbf{s}\sim\mathcal{D}} \left[\mathbb{E}_{\mathbf{a}\sim\pi(\cdot|\mathbf{s})} \left[A_H^{\pi}(\mathbf{s}, \mathbf{a}) \right] - \frac{1}{w\beta} D_{\text{KL}} \left(\pi(\cdot \mid \mathbf{s}) \middle\| \mu(\cdot \mid \mathbf{s}) \left[\sigma(\beta A_H^{\pi}(\mathbf{s}, \mathbf{a})) / Z(\mathbf{s}) \right] \right) \right]. \tag{9}$$

where μ is the behavior policy in replay data, β controls sensitivity to the estimated advantage, and w determines how aggressively the policy is improved. The above objective admits an elegant decomposition scheme of the optimal policy $\pi^*(\mathbf{a} \mid \mathbf{s}) \propto \pi^+(\mathbf{a} \mid \mathbf{s})^{1+w} / \pi^-(\mathbf{a} \mid \mathbf{s})^w$, where

$$\begin{aligned} \pi^+(\mathbf{a} \mid \mathbf{s}) &\propto \mu(\mathbf{a} \mid \mathbf{s}) \sigma(\beta A_H^{\pi}(\mathbf{s}, \mathbf{a})) = \mu(\mathbf{a} \mid \mathbf{s}) \rho^+(\mathbf{a} \mid \mathbf{s}), \\ \pi^-(\mathbf{a} \mid \mathbf{s}) &\propto \mu(\mathbf{a} \mid \mathbf{s}) (1 - \sigma(\beta A_H^{\pi}(\mathbf{s}, \mathbf{a})) = \mu(\mathbf{a} \mid \mathbf{s}) \rho^-(\mathbf{a} \mid \mathbf{s}) \end{aligned} \tag{10}$$

This suggests that the optimal policy π^* can be decomposed into a pair of positive π^+ and negative π^- sub-policies by maximizing and minimizing the advantage with opposite weighting schemes (ρ^+ and ρ^-), thereby fully exploiting both high- and low-quality data. In our work, we instantiate the sample weights $\rho^{\pm}$ according to the source and quality of each interaction. Human interventions $\mathcal{D}^h$ can be naturally treated as high-quality supervision and therefore enjoy full positive weight, while autonomous policy rollout samples $\mathcal{D}^p$ are scored by the estimated advantage, given as

$$\rho^+(\mathbf{s}_t, \mathbf{a}_t) = \begin{cases} 1, & (\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{D}^h, \\ \sigma(\beta A_H^{\pi}(\mathbf{s}_t, \mathbf{a}_t)) \,, & (\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{D}^p, \end{cases} \qquad \rho^-(\mathbf{s}_t, \mathbf{a}_t) = 1 - \rho^+(\mathbf{s}_t, \mathbf{a}_t). \tag{11}$$

We use these sample weights to train the positive and negative sub-policies separately. For the diffusion policy parameterized by ϵ_θ as in Eq. (2), we can initialize the two sub-policies ϵ_{θ^+} and ϵ_{θ^-} from ϵ_θ and optimize them with the following learning objective (Liang et al., 2026):

$$\mathcal{L}_{\epsilon_{\theta^\pm}} = \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D},\, \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I}),\, \tau} \left[\rho^{\pm}(\mathbf{s}_t, \mathbf{a}_t) \cdot \| \boldsymbol{\epsilon} - \epsilon_{\theta^\pm}(\mathbf{a}_\tau, \mathbf{s}_t, \tau) \|_2^2 \right] \tag{12}$$

Controllable inference. Different tasks may favor different degrees of aggressiveness in policy improvement. The above policy decomposition scheme allows us to flexibly compose the positive and negative sub-policies to extract the optimized policy $\widetilde{\epsilon}$ through the CFG mechanism at inference without retraining. This can be achieved by computing the following linear combination of ϵ_{θ^+} and ϵ_{θ^-}, and steering generation towards positive behaviors and away from the negative ones:

$$\widetilde{\epsilon}(\mathbf{a}_\tau, \mathbf{s}_t, \tau) = (1 + w)\epsilon_{\theta^+}(\mathbf{a}_\tau, \mathbf{s}_t, \tau) - w\epsilon_{\theta^-}(\mathbf{a}_\tau, \mathbf{s}_t, \tau), \tag{13}$$

where $w \geq 0$ controls the guidance strength. In practice, we select w according to task and environment characteristics, with values between 0.2 and 1.0 working well across our tasks.

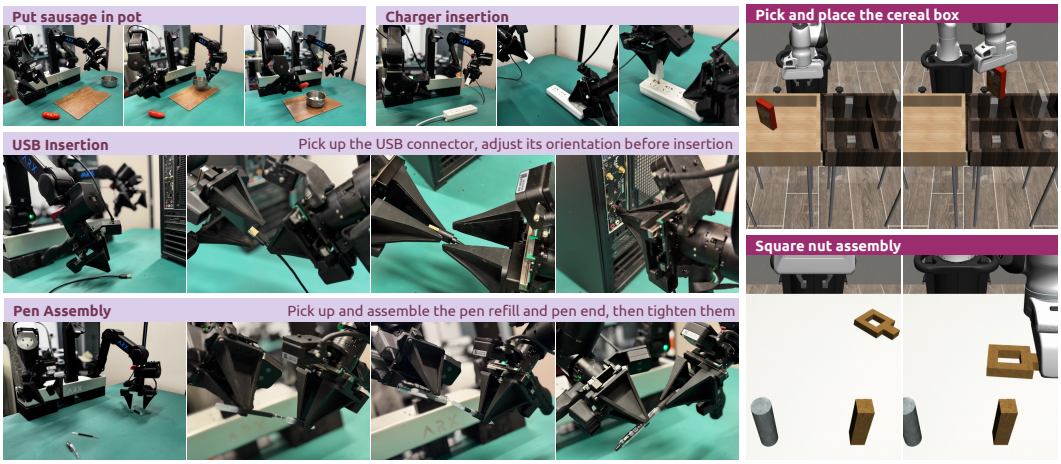

Figure 5: **Tasks in our evaluation suite.** Real-world tasks are shown in light purple boxes and simulation tasks in dark purple, with task names (and optional descriptions) above each demonstration.

4 Experiments

We conduct comprehensive experiments on both simulation and real-world bimanual manipulation tasks (the latter being particularly long-horizon and challenging) to demonstrate the effectiveness and efficiency of *DEC* against representative baseline methods. Moreover, we provide empirical evidence for the effectiveness of the key designs in *DEC*. Specifically, we demonstrate the reliability of the discriminator through failure-detection experiments. Ablation experiments also show that separately learning positive and negative sub-policies and composing them at inference time can achieve controllable inference while fully leveraging suboptimal data for policy improvement.

4.1 Experimental Setup

We finetune and evaluate policies over full episodes starting from the natural initial state. Our tasks span both simulated and real-world environments, as illustrated in Figure 5. The tasks and baseline methods are briefly introduced below. More implementation and task details are provided in Appendix D.1.

Tasks. For simulation evaluation, we use `PickPlaceCereal` and `NutAssemblySquare` in the ROBOSUITE environment (Zhu et al., 2020), covering object placement and precision insertion. For real-world evaluation, we consider four bimanual manipulation tasks: `PutSausageInPot`, `ChargerInsertion`, `USBInsertion`, and `PenAssembly`, spanning object generalization, precise insertion, in-hand adjustment, and long-horizon dexterous assembly.

Baseline methods. We compare *DEC* with representative policy finetuning and robotic RL baselines, including AWR+DAgger (Peng et al., 2019), HIL-SERL (Luo et al., 2025), DAgger (Kelly et al., 2019), DSRL (Wagenmaker et al., 2025), and RLT (Xu et al., 2026). Since these methods differ in both their applicability and empirical effectiveness across simulated and real-world settings, we select the most suitable and competitive baselines for each setting to compare. Specifically, we use the first four methods for simulation and the latter three for real-world evaluation. Implementation details for all baselines are provided in Appendix C.4.

4.2 Results

Efficient policy improvement with human intervention. Figures 6, 7 show that *DEC* consistently improves task success with limited environment interactions and human guidance, while requiring the least intervention time among human-in-the-loop methods. On `PickPlaceCereal`, only 20 rollout episodes yield 86.7% (+36.7%) success, exceeding all baselines even after 60 episodes, while on `PutSausageInPot`, success rises from 50.0% to 98.3% within 40 episodes. This improvement extends to substantially more challenging long-horizon bimanual tasks: on `PenAssembly` and `USBInsertion`, *DEC* increases success from 10.0% and 26.7% to 56.7% and 76.7%, respectively.

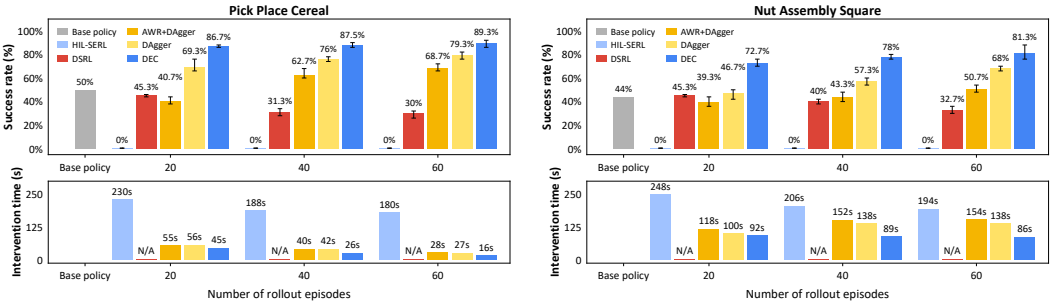

Figure 6: **Results for simulation tasks.** *DEC* demonstrates both highest performance and interaction efficiency across the two simulation tasks. Note that we augment AWR with imitation loss on human interventions (AWR+DAgger) to form a meaningful baseline. Intervention time denotes the human intervention used within each rollout stage. We use N/A for DSRL since human intervention is not applicable.

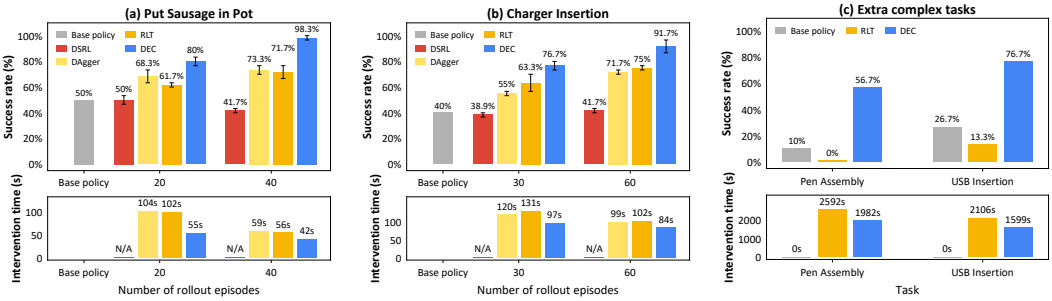

Figure 7: **Results for real-world tasks.** *DEC* shows the strongest performance on complex real-world manipulation tasks. As in Figure 6, we report two medium-difficulty tasks in **(a), (b)** and two substantially more challenging tasks in **(c)**. For `PenAssembly` and `USBInsertion`, we retain only the base policy and RLT as baselines, since the other methods achieve limited performance under the same interaction budget.

The contrast with RLT further suggests the limitation of relying on an additional lightweight policy for refinement in these challenging tasks. While DSRL relies on noise repetition to reduce the critic's input dimensionality and make learning tractable, we find that this same strategy degrades action generation with $\pi_{0.5}$ due to this noise distribution shift. Without policy pretraining, HIL-SERL fails to learn effective behaviors within the same interaction budget, highlighting the importance of pretrained behavioral priors. Overall, these results show that *DEC* efficiently leverages limited environment interactions and human guidance. Although directly finetuning a pretrained generative policy is challenging with limited experience, *DEC* effectively uses human guidance to improve the generative policy itself, which in turn enhances autonomous rollout quality and makes subsequent interaction effective.

Accurate discriminator and separable representations. We construct a test set from held-out trajectories to evaluate the discriminator's ability to anticipate whether a given $(\mathbf{s}, \mathbf{a})$ pair will eventually lead to failure, with detailed settings provided in Appendix D.2. Our discriminator achieves strong detection performance across simulated and real-world experiments, with particularly favorable recall relative to existing methods (Table 1). Intuitively, our PU-learning-style objective equips the discriminator with the ability to estimate eventual outcomes from $(\mathbf{s}, \mathbf{a})$ pairs. The t-SNE visualization (Figure 4) and representation comparisons in Appendix (Table 7) further support the idea of dynamics-aware pretraining, which yields more separable representations and consistently higher AUROC than the alternative representations evaluated.

Robust value learning. Figure 11 and Appendix E.2 illustrate the effects of our critic designs. Learned discounted partial returns produce smoother value progression and more stable, lower-variance value estimation than multi-step TD, while trajectory-held-out bootstrapping avoids the nearly uniform low estimates observed without it. These complementary effects support using G for long-range reward propagation and held-out value networks to mitigate trajectory-specific fitting.

Suboptimal experience and inference flexibility matter. To assess the contribution of suboptimal experience to *DEC*, we compare against two variants: one without the negative policy branch and

Table 1: **Failure detection on simulated and real-world tasks.** We report AUROC, accuracy (Acc.), and recall (Rec.). *DEC* achieves the highest recall and best or tied-best accuracy across all four tasks. Full results in simulation and real-world appear in Appendix Tables 5 and 6.

Methods	NutAssemblyRound			PickPlaceCereal			candy_in_plate			sausage_in_pot		
	AUROC↑	Acc.↑	Rec.↑	AUROC↑	Acc.↑	Rec.↑	AUROC↑	Acc.↑	Rec.↑	AUROC↑	Acc.↑	Rec.↑
LPB	0.92	**0.80**	0.74	0.91	0.83	0.83	0.92	0.77	0.54	0.90	0.75	0.57
logpZO	0.89	0.33	0.03	0.88	0.48	0.28	0.93	0.59	0.13	0.88	0.57	0.26
FLOAT	0.95	0.51	0.29	**0.96**	0.66	0.53	**0.97**	0.58	0.06	**0.99**	0.53	0.15
DEC(Ours)	**0.96**	**0.80**	**0.82**	0.95	**0.87**	**0.93**	0.95	**0.80**	**0.80**	0.92	**0.76**	**0.99**

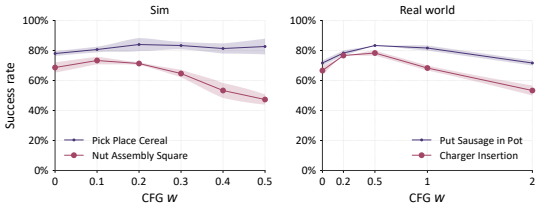

Figure 8: Success rate under different guidance weights w, on two simulation and two real-world tasks.

Table 2: Success rate throughout different training stages. % is omitted for brevity.

Task name	PickPlaceCereal			PutSausageInPot	
Training episodes	20 ep	40 ep	60 ep	20 ep	40 ep
w/o negative branch	69.3	74.7	75.3	73.3	93.3
w/o D^-	78.7	79.3	82.0	**79.3**	91.1
w/o human	49.3	52.7	56.0	48.9	51.1
DEC (Ours)	**84.7**	**86.7**	**88.7**	78.7	**98.3**

another that excludes failed trajectories from policy learning (Table 2). Both consistently underperform the full method, highlighting the value of explicitly modeling suboptimal experience. Removing human intervention also significantly reduces performance, underscoring the complementary role of corrective supervision. Moreover, composing the two branches supports controllable inference: an appropriate guidance strength w further improves performance over positive-only execution ($w = 0$), as shown in Figure 8.

5 Related Work

RL has long been studied for improving robotic policies directly through physical interaction, where sample efficiency is critical due to the cost of robot rollouts. Practical real-world RL systems therefore commonly build on off-policy learning, replay of prior experience, demonstrations, and human interventions to accelerate exploration and stabilize training (Ibarz et al., 2021; Ball et al., 2023; Luo et al., 2024; 2025). However, these systems are typically built around relatively lightweight policies. With increasingly capable pretrained robot policies, recent work has instead explored applying RL directly on top of these models to leverage their strong behavioral priors during online adaptation. One line directly updates the pretrained policy using value-based or advantage-conditioned objectives, often together with demonstrations or human corrections (Chen et al., 2025; Intelligence et al., 2025a; Xiao et al., 2026a; Dong et al., 2026). Another line freezes most or all of the base policy and learns lightweight modules that refine its outputs, including residual action policies, latent-space steering, and auxiliary actor-critic heads (Wagenmaker et al., 2025; Xu et al., 2026; Dong et al., 2026; Liu et al., 2026; Zheng et al., 2026a). Related approaches further use RL specialists to collect improved trajectories that are subsequently distilled back into the generalist policy (Xiao et al., 2026b). Despite this progress, existing methods often either rely on potentially brittle auxiliary policies or make limited use of the heterogeneous supervision available during online interaction.

6 Conclusion

We presented *DEC*, a framework for efficient real-world RL finetuning of pretrained robot policies. Its dynamics-aware discriminator guides human takeover and policy resumption, while trajectory-held-out bootstrapping and learned partial returns stabilize value estimation under sparse rewards. Complementary positive and negative policies learn from both successful and suboptimal experience and enable controllable action generation through composition. Experiments in simulation and on long-horizon, dexterous real-world tasks demonstrate policy improvement and sample efficiency with reduced human intervention, highlighting the effectiveness of *DEC* as a unified framework for real-world RL finetuning. Discussion on limitations and future works can be found in Appendix A.

AI USE STATEMENT

In this work, we used generative AI tools solely for language polishing and stylistic refinement. We did not use generative AI tools to generate technical content, experimental results, analyses, or conclusions, and other required disclosure categories are not applicable to this work. All AI-assisted text was reviewed and revised by the authors for accuracy and consistency with the intended technical content. We take full responsibility for the final content of this work, including any text produced with the aid of generative AI.

REFERENCES

Armaan A Abraham, Lucy Xiaoyang Shi, and Chelsea Finn. Long-horizon q-learning: Accurate value learning via n-step inequalities. *arXiv preprint arXiv:2605.05812*, 2026.

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

Huayu Chen, Kaiwen Zheng, Qinsheng Zhang, Ganqu Cui, Yin Cui, Haotian Ye, Tsung-Yi Lin, Ming-Yu Liu, Jun Zhu, and Haoxiang Wang. Nft: Bridging supervised learning and reinforcement learning in math reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 124025–124042, 2026.

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. In *Proceedings of Robotics: Science and Systems, RSS 2025, Los Angeles, CA, USA, Jun 21-25, 2025*, 2025. doi: 10.15607/RSS.2025.XXI.019.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Perry Dong, Kuo-Han Hung, Tian Gao, Dorsa Sadigh, and Chelsea Finn. Expo-ft: Sample-efficient reinforcement learning finetuning for vision-language-action models. *arXiv preprint arXiv:2605.25477*, 2026.

Qingxiu Dong, Li Dong, Yao Tang, Tianzhu Ye, Yutao Sun, Zhifang Sui, and Furu Wei. Reinforcement pre-training. *arXiv preprint arXiv:2506.08007*, 2025.

Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Diffusion guidance is a controllable policy improvement operator. *arXiv preprint arXiv:2505.23458*, 2025.

Qiao Gu, Yuanliang Ju, Shengxiang Sun, Igor Gilitschenski, Haruki Nishimura, Masha Itkina, and Florian Shkurti. Safe: Multitask failure detection for vision-language-action models. *Advances in Neural Information Processing Systems*, 38:40041–40076, 2026.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Ryan Hoque, Ashwin Balakrishna, Ellen Novoseller, Albert Wilcox, Daniel S Brown, and Ken Goldberg. Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning. *arXiv preprint arXiv:2109.08273*, 2021.

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. *IEEE Transactions on Robotics*, 2026.

Zhefeng Huang, Yilin Cai, Ankit Patel, Mohammad Hajiha, Brendan Browne, and Yue Chen. Failure detection for surgical robot imitation policies via flow-matching world modeling. *arXiv preprint arXiv:2607.27511*, 2026.

Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021.

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, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a vla that learns from experience, 2025a. URL https://arxiv.org/abs/2511.14759.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025b. URL https://arxiv.org/abs/2504.16054.

Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8077–8083. IEEE, 2019.

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

Ryuichi Kiryo, Gang Niu, Marthinus C Du Plessis, and Masashi Sugiyama. Positive-unlabeled learning with non-negative risk estimator. *Advances in neural information processing systems*, 30, 2017.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2021.

Qiyang Li, Zhiyuan Paul Zhou, and Sergey Levine. Reinforcement learning with action chunking. *Advances in Neural Information Processing Systems*, 38:55518–55553, 2026.

Ruiming Liang, Yinan Zheng, Kexin Zheng, Tianyi Tan, Jianxiong Li, Liyuan Mao, Zhihao Wang, Guang Chen, Hangjun Ye, Jingjing Liu, et al. Dichotomous diffusion policy optimization. In *International Conference on Learning Representations*, volume 2026, pp. 61669–61690, 2026.

Jingyi Liu, Zhaohong Mai, ShunSen He, Hang Ren, Chao Wang, Shunbo Zhou, XiaoDong Wu, and Heng Zhang. Hil-resrl: A model-agnostic finetuning adapter via human-in-the-loop residual reinforcement learning. *arXiv preprint arXiv:2606.22860*, 2026.

Jianlan Luo, Perry Dong, Yuexiang Zhai, Yi Ma, and Sergey Levine. Rlif: Interactive imitation learning as reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pp. 36329–36351, 2024.

Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 10(105):eads5033, 2025.

NVIDIA, :, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen

Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots, 2025. URL https://arxiv.org/abs/2503.14734.

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.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3 (1):9–44, 1988.

Tianyi Tan, Yinan Zheng, Ruiming Liang, Zexu Wang, Kexin Zheng, Jinliang Zheng, Jianxiong Li, Xianyuan Zhan, and Jingjing Liu. Flow matching-based autonomous driving planning with advanced interactive behavior modeling. *Advances in Neural Information Processing Systems*, 38:38310–38335, 2026.

Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, Steven Bohez, Konstantinos Bousmalis, Anthony Brohan, Thomas Buschmann, Arunkumar Byravan, Serkan Cabi, Ken Caluwaerts, Federico Casarini, Oscar Chang, Jose Enrique Chen, Xi Chen, Hao-Tien Lewis Chiang, Krzysztof Choromanski, David D'Ambrosio, Sudeep Dasari, Todor Davchev, Coline Devin, Norman Di Palo, Tianli Ding, Adil Dostmohamed, Danny Driess, Yilun Du, Debidatta Dwibedi, Michael Elabd, Claudio Fantacci, Cody Fong, Erik Frey, Chuyuan Fu, Marissa Giustina, Keerthana Gopalakrishnan, Laura Graesser, Leonard Hasenclever, Nicolas Heess, Brandon Hernaez, Alexander Herzog, R. Alex Hofer, Jan Humplik, Atil Iscen, Mithun George Jacob, Deepali Jain, Ryan Julian, Dmitry Kalashnikov, M. Emre Karagozler, Stefani Karp, Chase Kew, Jerad Kirkland, Sean Kirmani, Yuheng Kuang, Thomas Lampe, Antoine Laurens, Isabel Leal, Alex X. Lee, Tsang-Wei Edward Lee, Jacky Liang, Yixin Lin, Sharath Maddineni, Anirudha Majumdar, Assaf Hurwitz Michaely, Robert Moreno, Michael Neunert, Francesco Nori, Carolina Parada, Emilio Parisotto, Peter Pastor, Acorn Pooley, Kanishka Rao, Krista Reymann, Dorsa Sadigh, Stefano Saliceti, Pannag Sanketi, Pierre Sermanet, Dhruv Shah, Mohit Sharma, Kathryn Shea, Charles Shu, Vikas Sindhwani, Sumeet Singh, Radu Soricut, Jost Tobias Springenberg, Rachel Sterneck, Razvan Surdulescu, Jie Tan, Jonathan Tompson, Vincent Vanhoucke, Jake Varley, Grace Vesom, Giulia Vezzani, Oriol Vinyals, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Fei Xia, Ted Xiao, Annie Xie, Jinyu Xie, Peng Xu, Sichun Xu, Ying Xu, Zhuo Xu, Yuxiang Yang, Rui Yao, Sergey Yaroshenko, Wenhao Yu, Wentao Yuan, Jingwei Zhang, Tingnan Zhang, Allan Zhou, and Yuxiang Zhou. Gemini robotics: Bringing ai into the physical world, 2025. URL https://arxiv.org/abs/2503.20020.

Raymond Tsao, Andrew Wagenmaker, and Sergey Levine. Learning process rewards via success visitation matching for efficient rl. *arXiv preprint arXiv:2606.23640*, 2026.

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

Wei Xiao, Weiliang Tang, Yuying Ge, Hui Zhou, Yao Mu, Li Zhang, and Yixiao Ge. Rove: Unlocking human interventions for humanoid manipulation via reinforcement learning. *arXiv preprint arXiv:2606.17011*, 2026a.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Zhengyi Luo, Yuqi Xie, Fengyuan Hu, Jim Fan, Guanya Shi, et al. Self-improving vision-language-action models with data generation via residual rl. In *International Conference on Learning Representations*, volume 2026, pp. 13209–13236, 2026b.

Charles Xu, Jost Tobias Springenberg, Michael Equi, Ali Amin, Adnan Esmail, Sergey Levine, and Liyiming Ke. Rl token: Bootstrapping online rl with vision-language-action models. *arXiv preprint arXiv:2604.23073*, 2026.

Chen Xu, Tony Khuong Nguyen, Emma Dixon, Christopher Rodriguez, Patrick Miller, Robert Lee, Paarth Shah, Rares Ambrus, Haruki Nishimura, and Masha Itkina. Can we detect failures without failure data? uncertainty-aware runtime failure detection for imitation learning policies. *arXiv preprint arXiv:2503.08558*, 2025.

Wenye Yu, Jun Lv, Zixi Ying, Yang Jin, Chuan Wen, and Cewu Lu. Armada: Autonomous online failure detection and human shared control empower scalable real-world deployment and adaptation. *IEEE Robotics and Automation Letters*, 2026.

Haoqi Yuan, Zhixuan Liang, Anzhe Chen, Ye Wang, Haoyang Li, Pei Lin, Yiyang Huang, Zixing Lei, Tong Zhang, Jiazhao Zhang, Jie Zhang, Jingyang Fan, Gengze Zhou, Qihang Peng, Chenxu Lv, Xiaoyue Chen, An Yang, Fei Huang, Junyang Lin, Dayiheng Liu, Jingren Zhou, Chenfei Wu, and Xiong-Hui Chen. Qwen-robotmanip technical report: Alignment unlocks scale for robotic manipulation foundation models, 2026. URL https://arxiv.org/abs/2606.17846.

Huaihang Zheng, Yi Yang, Kai Ma, Shenglin Xu, Tian Xie, Guozheng Li, Xiangyu Wang, Yiren Ma, Si Liu, Yinian Mao, et al. Torl-vla: Tactile guided online reinforcement learning for contact-rich manipulation. *arXiv preprint arXiv:2606.09337*, 2026a.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, volume 2026, pp. 60580–60606, 2026b.

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. Diffusionnft: Online diffusion reinforcement with forward process. In *International Conference on Learning Representations*, volume 2026, pp. 134129–134150, 2026c.

Kexin Zheng, Xianyuan Zhan, and Xintao Yan. Horizon adaptive offline policy learning via value stitching, 2026d. URL https://arxiv.org/abs/2606.21136.

Ruijie Zheng, Xiyao Wang, Yanchao Sun, Shuang Ma, Jieyu Zhao, Huazhe Xu, Hal Daumé III, and Furong Huang. TACO: Temporal latent action-driven contrastive loss for visual reinforcement learning. *Advances in Neural Information Processing Systems*, 36:48203–48225, 2023.

Yinan Zheng, Ruiming Liang, Kexin Zheng, Jinliang Zheng, Liyuan Mao, Jianxiong Li, Weihao Gu, Rui Ai, Shengbo Li, Xianyuan Zhan, et al. Diffusion-based planning for autonomous driving with flexible guidance. In *International conference on learning representations*, volume 2025, pp. 37207–37227, 2025.

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

A Discussion & Limitations & Future Works

Here, we discuss the limitations and related potential solutions, as well as promising future directions of our work. Despite its sample efficiency, *DEC* incurs computational overhead from training multiple auxiliary models and maintaining two policy branches, both of which are evaluated during action generation. Moreover, the discriminator and value models rely on task-specific experience, and their transfer to unseen tasks and environments remains unverified. Future work could explore a shared policy that represents positive and negative behaviors through explicit conditioning, improving parameter efficiency while retaining controllable generation. Another promising direction is to develop multi-task discriminator and value models that generalize across tasks and environments, reducing the need for task-specific data collection, training, and calibration.

B Algorithm and Training Pipeline

DEC alternates between batch data collection and sequential model updates after an initial warmup. Each round uses fixed models for interaction and reuses accumulated experience to update the discriminator, return and value models, and compositional policy branches in order. Algorithm 1 summarizes the complete pipeline.

Algorithm 1 *DEC*: training pipeline

Require: Pretrained policy π_0; optional demonstrations $\mathcal{D}^{\text{demo}}$
1: Collect rollouts $\mathcal{D}_0^p$ using π_0
2: Initialize PU pools $(\mathcal{P}, \mathcal{U})$, reserving held-out successful trajectories for calibration
3: Training dynamic-aware encoder ϕ, Warm up discriminator g_0 and return/value models (G_0, V_0)
4: Calibrate the intervention threshold τ_0 on the held-out successful trajectories
5: Set $\mathcal{D} \leftarrow \mathcal{D}_0^p$; initialize $\pi_0^+ \leftarrow \pi_0$, $\pi_0^- \leftarrow \pi_0$
6: **for** round $k = 1, \ldots, K$ **do**
7: Collect N online rollouts $\mathcal{D}_{k,N}^{\text{online}}$ with human intervention using frozen $(\pi_{k-1}^{\pm}, g_{k-1}, \tau_{k-1})$
8: Aggregate new experience $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_{k,N}^{\text{online}}$
9: Add successful autonomous segments to $\mathcal{P}$ and those takeover or failure segments to $\mathcal{U}$
10: Update g_k using $\mathcal{P}$ and $\mathcal{U}$, recalibrate threshold τ_k
11: Update (G_k, V_k) using autonomous segments in $\mathcal{D}$
12: Update $\pi_k^{\pm}$ with source-aware, advantage-weighted supervision on $\mathcal{D}$ and $\mathcal{D}^{\text{demo}}$
13: **end for**
14: **return** $\pi_K^{\pm}$ for guided inference

B.1 Warmup and Batch Data Collection

Warmup. We collect initial rollouts with the pretrained policy to train the dynamics-aware representation and failure discriminator and initialize the return and value models (G, V). Both policy branches start from the pretrained policy, and the dynamics-aware representation remains frozen during online updates.

Batch rollout. Each round collects N trajectories using fixed policy branches, discriminator, and intervention threshold. The discriminator guides human takeover and policy resumption, and the resulting experience is added to replay. We split trajectories at takeover and resumption into contiguous autonomous and human-controlled segments. This segmentation is shared by discriminator and value learning.

B.2 Discriminator Update and Calibration

Segment-level PU supervision. We retain the initial PU data and incorporate newly collected autonomous segments according to their endpoints. Segments that directly reach task success enter the positive pool $\mathcal{P}$, while those ending in human takeover or task failure enter the unlabeled pool $\mathcal{U}$. Thus, an autonomous segment preceding human recovery does not inherit the eventual episode success. Human-controlled segments are excluded from discriminator training.

Threshold-centered regularization. We evaluate the nnPU objective on logits centered at the previous decision boundary and normalized by a fixed calibration scale. We additionally apply $\mathcal{L}_{\text{cap}} = \mathbb{E}[(|\bar{g}| - c)_+^2]$, where $\bar{g}$ denotes the normalized logit and $c > 0$ sets the cap. This penalizes excessive logit magnitudes without directly optimizing the intervention threshold.

Threshold calibration. After warmup and each discriminator update, we set the intervention threshold to the empirical $(1 - \delta)$-quantile of failure scores $f = -g$ on a fixed held-out set of successful trajectories, where δ specifies the upper-tail fraction. These trajectories are excluded from discriminator training, and the threshold remains fixed throughout the next rollout batch. Empirically, we find that setting the threshold to the 98th percentile ($\delta = 0.02$) works well.

B.3 Return and Value Learning

We use the updated discriminator to relabel autonomous replay rewards, which remain fixed throughout the return and value updates. Training samples are constructed separately within autonomous segments and never cross human-controlled intervals. A segment ending in takeover uses the unsuccessful-boundary convention rather than receiving credit for subsequent human recovery. We train G using observed partial returns and temporal compositional consistency, and update V through trajectory-held-out bootstrapping. The fitted models then provide chunk-level advantages, which remain fixed during policy optimization.

B.4 Policy Update and Deployment

Human corrections and optional demonstrations supervise only the positive branch, while autonomous actions train both branches using complementary advantage-based weights. We optimize the normalized weighted generative objectives defined in the main text. After all updates are complete, the composed policy and updated discriminator are deployed for the next rollout batch.

C Implementation Details

C.1 Base Policies and Finetuning

For real-world experiments, we use the pretrained $\pi_{0.5}$ vision-language-action model (Intelligence et al., 2025b) as our base policy, which generates continuous action chunks conditioned on visual observations, proprioceptive states, and language instructions. We first perform supervised finetuning on task-specific human demonstrations using the flow-matching objective, applying low-rank adaptation (LoRA) to the vision-language backbone and action expert while keeping the underlying pretrained weights frozen. The resulting checkpoint initializes both the positive and negative policy branches in *DEC*, whose LoRA parameters are subsequently optimized separately using accumulated real-world experience. At deployment, the two branches are composed using the guidance rule introduced in the main text.

For simulation experiments, we use a language-conditioned flow-matching policy which is pretrained on ROBOSUITE tasks as the base policy, which predicts continuous action chunks from multi-view RGB observations, robot proprioception, and task instructions. Following prior work that adapts pretrained policies with task-relevant demonstrations before RL finetuning (Xu et al., 2026; Xiao et al., 2026b), we train the base policy on a small set of expert demonstrations from ROBOSUITE using the supervised flow-matching objective. The resulting checkpoint initializes both the positive and negative policy branches in *DEC*, whose trainable parameters are subsequently optimized separately using accumulated human interventions and autonomous rollouts under the complementary advantage-weighted objectives described in the main text.

C.2 Discriminator and Dynamic Pretraining

We extract multi-view visual features using a frozen DINOv3 ViT-B/16 backbone and pool them into a 4×4 grid per view. A Transformer combines these features with proprioception and action tokens, and is pretrained on successful trajectories to predict normalized visual features after executing a H-step action chunk using mean squared error.

We then freeze the dynamics model and flatten token representations from an intermediate layer as input to an MLP discriminator. Empirically, we find that representations from layers 1–3 work well. The discriminator is trained with the logistic nnPU objective in Eq. 3, using balanced positive and unlabeled minibatches and an additional logit penalty $\lambda_{\text{cap}}\mathbb{E}[\max(|g| - c, 0)^2]$. Default real-world hyperparameters are summarized in Table 3.

Table 3: Default real-world hyperparameters for dynamics pretraining and discriminator initialization.

Hyperparameter	Dynamics model	Discriminator head
Architecture	Transformer	MLP
Depth / hidden width	9 / 384	3 / 512
Dropout	0.1	None
Optimizer	AdamW	AdamW
Learning rate	10^{-4}	3×10^{-4}
Learning-rate schedule	Constant	Cosine decay
Weight decay	10^{-4}	10^{-4}
Batch size	256	512
Action-chunk length H	30	30
Dynamics pretraining steps	5,000	—
Positive prior η	—	0.3
Positive : unlabeled sampling	—	1 : 1
Logit cap c / penalty weight λ_{cap}	—	2.0 / 0.01
Calibration trajectories	—	10 successful
Failure-score threshold quantile	—	0.98

C.3 Return Modeling and Value Learning

We use frozen state features with trainable projections. The return model G_ξ takes the projected start and endpoint features concatenated with the scalar horizon n; both G_ξ and each V_{ϕ_m} use the MLP architecture in Table 4. For each sampled start state, we draw n uniformly from the available horizons up to N, and sample $j \mid n \sim \text{Unif}\{1, \dots, n-1\}$ when $n \geq 2$. The compositional term is omitted for $n = 1$. Each sampled bootstrap target is detached and used directly in expectile regression, with terminal continuation terms masked out. We update G_ξ and the value networks once per minibatch using separate AdamW optimizers, followed by Polyak updates of the target value networks.

C.4 Baselines

We evaluate AWR and HIL-SERL only in simulation, RLT only in the real world, and DAgger and DSRL in both settings. All methods are trained and evaluated over complete task episodes starting from natural initial states. Below, we describe each baseline and its adaptation to our experimental setting.

AWR (Peng et al., 2019). We adapt AWR to our generative policy by reweighting the flow-matching objective according to learned advantage estimates, placing greater emphasis on higher-value actions (Zheng et al., 2025). Advantages are computed using clipped double-Q estimates and a value function trained with expectile regression using $\tau = 0.7$. Human interventions are replayed alongside autonomous experience, and we denote this variant as AWR+DAgger. We use learning rates of 10^{-4} and 3×10^{-4} for policy and value learning, respectively.

HIL-SERL (Luo et al., 2025). We implement HIL-SERL as an RLPD-style off-policy actor–critic without initialization from the pretrained generative policy. Demonstrations and online interventions are replayed together with autonomous experience using an equal sampling ratio. We use frozen pretrained ResNet-10 visual encoders and two-layer MLP actors and critics with 256 hidden units per layer, including an ensemble of two Q-functions. Training is initialized with 20 demonstration episodes, uses a batch size of 256, and performs two critic updates per actor update.

Table 4: Default real-world hyperparameters for return and value learning.

Hyperparameter	Value
Executed action-chunk length H	30
Maximum backup horizon N (chunks)	8
Primitive-step discount γ	0.99
Composition coefficient λ_{comp}	0.5
Expectile level τ_e	0.7
Number of value networks M	5
MLP hidden dimensions (G_ξ and V_{ϕ_m})	$(512, 512, 512)$
Visual / proprioceptive projection dimensions	128 / 32
Activation / normalization	Mish / LayerNorm
Optimizer	AdamW
Learning rate (G_ξ and V_{ϕ_m})	10^{-4}
Weight decay	2×10^{-6}
Batch size	512
Gradient clipping norm	2.0
Polyak coefficient (online-parameter weight)	0.005
Warmup updates	40,000
Updates per online round	10,000

DAgger (Kelly et al., 2019). We use human-gated DAgger, collecting human corrections during policy rollouts and finetuning the pretrained policy on the aggregated corrections using the supervised generative objective. For simulation, training uses a learning rate of 3×10^{-6}, a batch size of 64, and 5,000 optimization steps per round. For real-world tasks, we finetune $\pi_{0.5}$ using LoRA with a constant learning rate of 2.5×10^{-5} and a batch size of 48, and each round consists of 5 rollout episodes followed by 200 optimization steps

DSRL (Wagenmaker et al., 2025). DSRL keeps the pretrained generative policy frozen and learns a state-conditioned policy over its initial noise, allowing RL to steer the generated action chunks by updating only the latent actor and associated critics. We use the noise-aliased variant (DSRL-NA), with a 50-dimensional latent action that is directly reshaped into the generator's 5×10 chunk-shaped noise input. Training is initialized with the same rollout episodes as ours and uses 30 actor–critic updates per update cycle. DSRL does not use online human corrections. For real-world settings, we apply DSRL to the task-specific SFT checkpoint of $\pi_{0.5}$ while keeping the base policy frozen. We find that repeating a single noise vector across the entire action chunk, as in the original π_0 implementation, severely degrades performance in our setting. We therefore use a partially repeated noise parameterization: the latent actor predicts distinct values for a subset of noise entries, while repetition is restricted to the remaining entries.

RLT (Xu et al., 2026). We instantiate RLT on our task-adapted $\pi_{0.5}$ policy. A 2048-dimensional RL token is trained by reconstructing frozen VLA features and then frozen together with the VLA during online RL. The actor conditions on this token, proprioception, and a VLA reference chunk to predict 30-step action chunks. The actor and twin critics use two 256-unit hidden layers. Human interventions enter replay and provide imitation targets for actor learning.

D Experimental Setup

D.1 Task Descriptions

Our evaluation suite comprises two simulated manipulation tasks and four real-world bimanual tasks, as illustrated in Figure 5. All policies are finetuned and evaluated over complete episodes starting from natural initial states.

D.1.1 Simulation Tasks

We use the built-in Franka Panda robot in ROBOSUITE (Zhu et al., 2020) to evaluate two tasks covering object placement and precision insertion. Human intervention is provided through teleoperation with a SpaceMouse.

`PickPlaceCereal`. The robot must pick up a cereal box and place it in the designated target compartment. The policy must infer the box's position and orientation from visual observations, establish a stable grasp, and transport and release the object at the correct location. This task tests visually guided grasping and placement across variations in object pose.

`NutAssemblySquare`. The robot must grasp a square nut, reorient it, and fit it onto a square peg. Successful insertion requires both positional and angular alignment between the nut's opening and the peg. The task therefore combines object reorientation with precise, contact-rich manipulation.

D.1.2 Real-World Tasks

We use a dual-arm ARX A5 robot to evaluate four manipulation tasks spanning spatial generalization, insertion under limited visual observability, in-hand adjustment, and long-horizon assembly. Human intervention is performed by directly manipulating the robot arms. All real-world tasks run at a control frequency of 30 Hz and typically take 30-120 s to complete.

`PutSausageInPot`. The robot must place a sausage into a pot through coordinated bimanual manipulation. The sausage and pot positions vary over a wide workspace, requiring the policy to adapt its grasping and placement motions to different object–target configurations. The main challenge is maintaining reliable execution across these spatial variations.

`ChargerInsertion`. The robot must grasp a charger, align it with the socket, and complete insertion. The task demands centimeter-level alignment despite limited visibility of the insertion interface. The policy must coordinate the approach and insertion motions while correcting positional and angular misalignment from incomplete visual feedback.

`USBInsertion`. The robot must pickup the connector from the table, infer the orientation of a USB cable, adjust the connector pose through bimanual in-hand manipulation, and insert it into the corresponding port. The adjustment stage must establish a grasp and orientation suitable for subsequent alignment and insertion. This task couples orientation estimation, dexterous grasp adjustment, and precise insertion within a single sequence.

`PenAssembly`. The robot must assemble a pen from three separate components through a long-horizon sequence of bimanual manipulation. This is the most challenging task in our suite, requiring successive grasping, reorientation, and precise assembly operations. Each stage must leave the components in a configuration suitable for the next, making successful completion sensitive to errors accumulated throughout the sequence.

Here, we provide visualizations of the randomized initial space for each real world task in Figure 9. The randomized regions are highlighted in purple.



Figure 9: **Visualization for initial regions for real world tasks.** Objects are randomly initialized in purple areas during training and evaluation.

D.2 Discriminator Experiments

We evaluate failure detection on five simulated manipulation tasks in ROBOSUITE and four real-world bimanual tasks, spanning pick-and-place, stacking, and contact-rich assembly (illustated in

Figure 10). The data are collected by executing task-adapted policies and recording complete observation–action trajectories together with their task outcomes. In total, we collect 500 simulated and 400 real-world trajectories for training, calibration and evaluation. During training, state–action pairs from successful trajectories form the positive set, whereas complete failed trajectories form the unlabeled set; no failure-onset annotation is used. For evaluation, failed trajectories are additionally annotated at the frame level by human expert, while successful trajectories are treated as failure-free. Intervention thresholds are calibrated using held-out successful trajectories without accessing failure annotations. We report frame-level AUROC, accuracy, and failure recall.

Discriminator baselines. We compare our nnPU discriminator with three representative failure-detection methods. **LPB** (Yu et al., 2026) measures the distance between the current policy representation and its nearest neighbor in an expert feature bank, treating large distances as evidence of failure. **logpZO** (Xu et al., 2025) trains a flow-based model on expert representations, constructs an out-of-distribution score from the learned flow field, and calibrates a time-dependent decision boundary using functional conformal prediction. **FLOAT** (Gu et al., 2026) compares each partially observed rollout with expert trajectories through partial optimal-transport matching and uses the resulting alignment cost as its failure score. For the simulated experiments, these baselines use frozen representations extracted from the same pretrained flow policy.

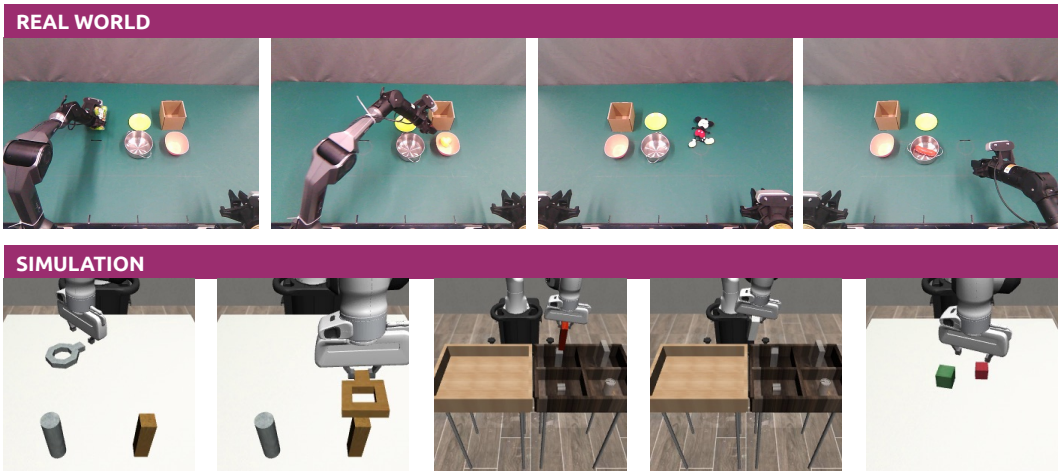

Figure 10: **Tasks used in the discriminator experiments.** The top row shows the four real-world tasks: Candy in Plate, Duck in Bowl, Micky in Box, and Sausage in Pot. The bottom row shows the five simulated tasks: NutAssemblyRound, NutAssemblySquare, PickPlaceCereal, PickPlaceMilk, and Stack. Each panel presents a representative observation from a successful trajectory.

Representation comparison. We also study how the state–action representation affects failure discrimination while keeping the nnPU objective, discriminator head, data splits, and threshold-calibration procedure fixed. We compare four representations on three representative simulation tasks. **RPT** (Dong et al., 2025) learns sensorimotor representations through masked reconstruction over short observation–action histories. **TACO** (Zheng et al., 2023) uses a temporal action-driven contrastive objective to associate the current state and action sequence with the resulting future state. **Policy** directly reuses the fused task-and-scene representation of the frozen pretrained control policy without additional representation pretraining. This controlled comparison isolates the contribution of dynamics-aware pretraining from that of the subsequent nnPU discriminator.

E Additional Results and Analyses

E.1 Discriminator Results

Failure detection across tasks. Tables 5 and 6 provide the full failure-detection results. *DEC* achieves the highest failure recall on eight of the nine tasks, with best or tied-best accuracy on all five simulated tasks. Although FLOAT often achieves higher AUROC, its recall remains substantially lower at the calibrated thresholds. Together, these results show that *DEC* combines strong

discriminative performance with high failure recall at the calibrated thresholds, providing a reliable signal for guiding human intervention across diverse tasks.

Table 5: Discriminator performance on simulated benchmark tasks. Higher is better for all metrics.

Methods	NutAssemblyRound			NutAssemblySquare			PickPlaceCereal			PickPlaceMilk			Stack		
	AUROC↑	Acc.↑	Rec.↑	AUROC↑	Acc.↑	Rec.↑	AUROC↑	Acc.↑	Rec.↑	AUROC↑	Acc.↑	Rec.↑	AUROC↑	Acc.↑	Rec.↑
LPB	0.92	**0.80**	0.74	0.88	0.71	0.60	0.91	0.83	0.83	0.87	0.78	0.73	0.70	0.42	0.32
logpZO	0.89	0.33	0.03	0.86	0.32	0.01	0.88	0.48	0.28	0.81	0.49	0.11	0.57	0.18	0.02
FLOAT	0.95	0.51	0.29	0.90	0.48	0.26	**0.96**	0.66	0.53	**0.94**	0.70	0.49	**1.00**	0.31	0.18
DEC(Ours)	**0.96**	**0.80**	**0.82**	**0.92**	**0.79**	**0.74**	0.95	**0.87**	**0.93**	**0.94**	**0.89**	**0.83**	0.96	**0.87**	**0.86**

Table 6: Discriminator performance on real-world benchmark tasks. Higher is better for all metrics.

Methods	Micky_in_box			candy_in_plate			duck_in_bowl			sausage_in_pot		
	AUROC↑	Acc.↑	Rec.↑	AUROC↑	Acc.↑	Rec.↑	AUROC↑	Acc.↑	Rec.↑	AUROC↑	Acc.↑	Rec.↑
lpb	0.91	**0.82**	**0.88**	0.92	0.77	0.54	0.89	**0.81**	0.72	0.90	0.75	0.57
logpZO	0.90	0.57	0.17	0.93	0.59	0.13	0.90	0.49	0.30	0.88	0.57	0.26
float	**0.98**	0.66	0.23	**0.97**	0.58	0.06	**0.94**	0.66	0.19	**0.99**	0.53	0.15
DEC(Ours)	0.97	0.61	0.82	0.95	**0.80**	**0.80**	0.90	0.67	**0.83**	0.92	**0.76**	**0.99**

Effect of representation pretraining. Table 7 compares representations under the same nnPU training and calibration protocol. Dynamics-aware pretraining achieves the highest AUROC across all three tasks, supporting the effectiveness of dynamic transition prediction for learning state–action features that distinguish successful and failing behaviors.

Table 7: Comparison of latent representations for failure detection. Higher is better for all metrics. The best result in each column is highlighted in bold.

Methods	NutAssemblySquare			PickPlaceCereal			Stack		
	AUROC↑	Acc.↑	Rec.↑	AUROC↑	Acc.↑	Rec.↑	AUROC↑	Acc.↑	Rec.↑
RPT	0.68	0.64	0.61	0.67	0.64	0.56	0.78	0.71	0.67
TACO	0.88	**0.83**	**0.88**	0.93	0.81	**0.96**	0.92	**0.87**	**0.93**
Policy	0.81	0.74	0.70	0.85	0.83	0.90	0.79	0.77	0.79
DEC(Ours)	**0.92**	**0.83**	0.74	**0.95**	**0.87**	0.93	**0.96**	**0.87**	0.86

E.2 Value Learning Visualization

We visualize state-value predictions along representative trajectories from the real-world `PutSausageInPot` task, following the return and value learning setup in Appendix C.3. We separately examine trajectory-held-out bootstrapping and learned discounted partial returns. Figure 11 plots model predicted values against normalized trajectory progress, with corresponding observations shown above each panel.

Trajectory-held-out bootstrapping. The left panel compares value estimates with and without trajectory-held-out bootstrapping on a rollout that ultimately fails after the sausage is dropped. With trajectory-level exclusion, bootstrap targets are supplied by value networks trained on other trajectory splits. The resulting values increase as the robot makes progress and decrease sharply at the dropping event, whereas the variant without holdout assigns nearly uniform low values throughout the displayed segment. This suggests that held-out bootstrapping helps distinguish promising intermediate states from the eventual failed outcome, mitigating trajectory-specific overfitting.

Discounted partial returns. The right panel compares value learning using learned discounted partial returns against 3-step TD on a successful rollout. Learned partial returns yield a smoother increase in value toward task completion, while 3-step TD exhibits noticeable fluctuations, particularly during the final stage. This smoother progression supports the use of learned partial returns to

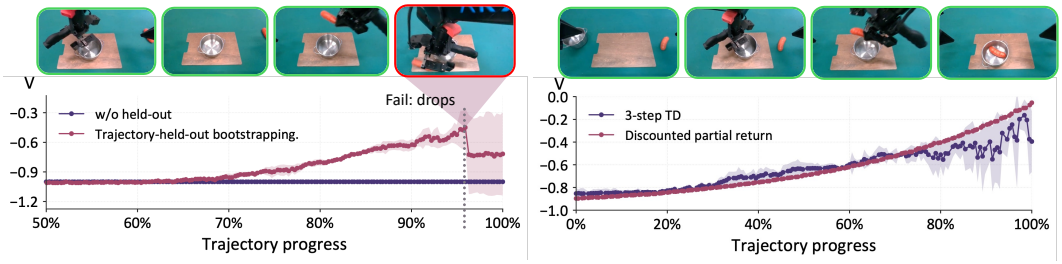

Figure 11: Value-learning visualization on the real-world `PutSausageInPot` task. **Left:** Effect of trajectory-held-out bootstrapping on a failed rollout; the vertical dotted line marks the dropping event. **Right:** Comparison of learned discounted partial returns and 3-step TD on a successful rollout. Observations above each panel illustrate the corresponding task progression.

stabilize long-range value propagation and provide more consistent evaluations of intermediate task progress.