ONLINE VLA ADAPTATION WITH STREAMING REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Streaming Reinforcement Learning (RL) learns directly from sequential experience without relying on replay buffers or minibatch updates. It offers an appealing framework for online adaptation of vision-language-action (VLA) models, but this direction remains largely underexplored. In this work, we investigate this problem and develop Streaming with Intentional and Episodic Updates (SIE), a new streaming VLA adaptation framework that combines streaming critic learning with an Episodic Policy Updates (EPU) mechanism, which incorporates episodic success feedback through a lightweight actor update. Experiments with π_0 on LIBERO-90 demonstrate that our framework improves mean final success rate from 47.20% to 72.09% compared with the baseline. In a synchronous two-device benchmark, SIE reduces client wall time by 66.5%. Our results show that streaming RL is a promising direction for efficient VLA adaptation.

1 INTRODUCTION

Vision-Language-Action (VLA) models have attracted growing attention as a promising foundation for general-purpose robotic manipulation (Black et al., 2026; Physical Intelligence et al., 2025b; 2026). They are typically pre-trained on a large collection of manipulation tasks, but deploying them to a new task often requires further adaptation. Reinforcement Learning (RL) with online interaction provides a natural approach to adapting VLAs (Lu et al., 2025; Physical Intelligence et al., 2025a). However, real-world adaptation is often performed under strict latency and computational constraints, especially on resource-constrained robotic platforms, such as robots deployed for lunar exploration or disaster response. These constraints call for VLA adaptation methods that incur low memory and update overhead.

Streaming RL offers a promising direction. It learns from each incoming transition without experience replay or minibatch updates, effectively reducing storage and computational overhead compared with replay-based learning. Achieving satisfactory performance with streaming RL, however, is still challenging — a difficulty often described as the *stream barrier* (Elsayed et al., 2026). Although recent advances (Vasan et al., 2024; Sharifnassab et al., 2026) have made progress in overcoming this barrier across a range of RL benchmarks, whether they can enable efficient online adaptation of VLAs remains largely unclear.

In this work, we propose Streaming with Intentional and Episodic Updates (SIE), a framework for online VLA adaptation through streaming RL. We follow the paradigm of tuning a frozen pre-trained VLA with RL in the latent noise space (Wagenmaker et al., 2025; Lu et al., 2026) to enable lightweight online optimization. To learn from streaming transitions, we build on Intentional Updates (IU) (Sharifnassab et al., 2026), a recent streaming RL method, and introduce the Episodic Policy Updates (EPU) mechanism for better learning from successful episodes. Specifically, we notice that a replay-free streaming learner typically updates its actor during an episode, but robotic manipulation tasks often have sparse rewards only at the end of an episode. Thus, the training signal of the actor heavily relies on the quality of the critic. For example, if the critic fails to assign high values to decisions leading to a successful episode, then the actor may fail to learn from its success. To address this issue, EPU performs an auxiliary actor update for successful episodes, dynamically increasing the cumulative log-likelihood assigned to the decisions executed along the successful trajectory. Overall, SIE integrates continual critic-guided policy improvement with its EPU component for episodic success-driven updates.

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.

We implement SIE on top of the representative DSRL (Wagenmaker et al., 2025) method and evaluate it on three benchmarks, together with several analytical experiments to investigate the effectiveness and efficiency of SIE and the advantage of the EPU component. In particular, on a 15-task subset of LIBERO (Liu et al., 2023), SIE improves mean final success rate from 47.20% to 72.09%. Evaluations with $\pi_{0.5}$ on LIBERO and Meta-World, and with π_0 on ALOHA show that SIE outperforms baseline methods across different settings. In a synchronous two-device benchmark, it reduces client wall time by 66.5% and peak allocated learner GPU memory by 70.2% relative to replay-based DSRL. Analytical experiments further show that EPU increases the actor's relative preference for decisions executed in successful episodes.

2 Related Work

Adaptation of generative robot policies and VLAs. Diffusion Policy and flow-based VLAs such as π_0 model continuous action chunks through iterative generation (Chi et al., 2024; Black et al., 2026). Adapting these policies through RL requires accounting for their generative structure: DPPO optimizes the denoising process, while Flow Policy Optimization uses a flow-matching loss to avoid exact action-likelihood computation (Ren et al., 2024; McAllister et al., 2026). VLA-RL studies scalable RL post-training, and RECAP combines autonomous experience and human interventions with advantage-conditioned training (Lu et al., 2025; Physical Intelligence et al., 2025a). An alternative is to keep the pretrained policy frozen: V-GPS selects actions using a learned value function, while ResiP and Policy Decorator learn action-space residuals (Nakamoto et al., 2024; Ankile et al., 2024; Yuan et al., 2025). DSRL learns a compact noise policy that steers the frozen generator (Wagenmaker et al., 2025). Following DSRL's noise-space steering approach, subsequent work such as FRS (Tang et al., 2026) and UniSteer (Lu et al., 2026) use flow inversion to incorporate reference actions or human corrections into noise-space learning. We adopt DSRL's latent-space formulation and investigate its use under streaming learning constraints.

Streaming deep reinforcement learning. Learning incrementally from sequential experience has long been central to RL, as exemplified by temporal-difference (TD) learning and Q-learning (Sutton, 1988; Watkins & Dayan, 1992). TD-Gammon demonstrates early success with neural TD learning (Tesauro, 1995), and later work explores deep trace-based control without experience replay (Elfwing et al., 2018). For deep RL, Stream-X combines update-size control, traces, and normalization to overcome the *stream barrier* across prediction and control tasks (Elsayed et al., 2026); AVG establishes effective incremental learning with reparameterized policy gradients (Vasan et al., 2024). Intentional Updates (IU) specify desired changes in predictions or sampled-action log-probabilities and choose step sizes to realize them (Sharifnassab et al., 2026), while recent work extends streaming RL through representation learning (Nilaksh et al., 2026), continuous-control actor–critic methods (De Monte et al., 2026), and recurrent learning under partial observability (Farr et al., 2026). Building on these advances, we study streaming RL for the online adaptation of VLAs.

Learning from episodic success feedback. Reward-weighted regression (Wierstra et al., 2008) and AWR (Peng et al., 2019) connect policy learning to weighted supervised fitting. Self-imitation and related methods reuse high-return decisions or successful trajectories to supplement online RL (Oh et al., 2018; Li et al., 2022; Jia & Chen, 2026). For VLA adaptation, HABC derives transition weights from episode outcomes, while SRPO uses successful trajectory references to shape feedback for failed rollouts (Fang et al., 2026; Fei et al., 2025). Inspired by these works, our EPU supplements streaming policy updates with feedback from successful episodes.

3 Preliminaries

3.1 Latent-Space Adaptation

Flow-based VLAs such as π_0 and $\pi_{0.5}$ generate an action chunk by iteratively transforming Gaussian noise (Black et al., 2026; Physical Intelligence et al., 2025b). DSRL instead lets a small trainable actor choose this noise, and we follow this paradigm. At policy query t, the latent actor π_θ samples a low-dimensional latent noise z_t conditioned on the observation s_t, and the frozen VLA F_ϕ with

weights ϕ turns it into an action chunk:

$$\mathbf{a}_t = F_\phi(s_t, z_t), \quad z_t \sim \pi_\theta(\cdot \mid s_t). \tag{1}$$

Because F_ϕ is fixed, it can be treated as part of the environment, and hence adapting it reduces to learning the small actor π_θ over a low-dimensional action space.

The robot executes the first q actions of $\mathbf{a}_t$ before the next query, so t counts policy queries rather than environment steps. Using task success as the only feedback, a query receives reward $r_t = 0$ if the task succeeds within its q steps, which terminates the episode, and $r_t = -1$ otherwise; episodes without success are truncated at a time limit. With a per-step discount γ_{env}, the discount between queries is $\gamma = \gamma_{\text{env}}^q$, and the actor seeks to maximize $J(\theta) = \mathbb{E}[\sum_{t\geq 0} \gamma^t r_t]$ with ϕ fixed.

3.2 Soft Actor–Critic for Latent Control

DSRL trains the latent actor with Soft Actor–Critic (SAC) (Haarnoja et al., 2018; 2019), jointly learning a critic and an actor. The critic $Q_\psi(s, z)$ predicts the return after choosing latent z at observation s and following π_θ afterwards. It is trained to minimize the squared temporal-difference (TD) error on transitions (s, z, r, s'), with s' the next observation, sampled from a replay buffer $\mathcal{D}$:

$$\mathcal{L}_Q(\psi) = \mathbb{E}_{\mathcal{D}}\big[\big(y - Q_\psi(s, z)\big)^2\big], \qquad y = r + \gamma m\, Q_{\bar{\psi}}(s', z'), \quad z' \sim \pi_\theta(\cdot \mid s'), \tag{2}$$

where $Q_{\bar{\psi}}$ is a slowly updated copy of the critic, called the target network, and the mask m is 0 at termination and 1 otherwise. Compared to standard SAC, we omit the next-state entropy term from the critic target following Wagenmaker et al. (2025). The actor is then updated to favor latents with higher predicted value, while its entropy term is retained to keep its choices stochastic. With the critic held fixed, the actor minimizes

$$\mathcal{L}_\pi(\theta; s) = \mathbb{E}_{z\sim\pi_\theta(\cdot|s)}\big[\alpha \log \pi_\theta(z \mid s) - Q_\psi(s, z)\big], \tag{3}$$

where the temperature $\alpha > 0$ is learned to keep the policy entropy near a target value.

In the streaming RL setting we study (Elsayed et al., 2026), however, the learner updates on each transition (s_t, z_t, r_t, s_{t+1}) as it arrives, keeping memory and per-query computation bounded on resource-constrained robots. In this setting, we cannot sample minibatches from $\mathcal{D}$ or use multiple updates per collected transition, and there is no target network (Elsayed et al., 2026; Vasan et al., 2024) available. Without these components, learning from single transitions becomes considerably harder, a difficulty known as the stream barrier (Section 1).

4 Method

SIE updates the critic with Intentional Updates (IU) (Sharifnassab et al., 2026) and integrates Episodic Policy Updates (EPU) into the SAC framework.

4.1 Streaming Actor–Critic Learning with Intentional Updates

IU (Sharifnassab et al., 2026) specifies how much each update should change a value prediction and derives the step size from it. We describe its Q-learning version, Intentional Q-learning, in our notation, with the latent z as the action. For a transition at (s_t, z_t) with TD target y_t and TD error $\delta_t = y_t - Q_\psi(s_t, z_t)$ (replaced in practice by an adaptively clipped value $\widetilde{\delta}_t$), Intentional Q-learning requests that the update move $Q_\psi(s_t, z_t)$ a fraction $\eta_Q \in (0, 1]$ of the way toward the target:

$$Q_{\psi^+}(s_t, z_t) \approx Q_\psi(s_t, z_t) + \eta_Q \delta_t, \tag{4}$$

where ψ^+ denotes the parameters after the update. It realizes this intended change with a semi-gradient TD step whose step size is normalized by the squared gradient norm:

$$\psi^+ = \psi + \beta_t \delta_t\, g_t, \qquad \beta_t = \frac{\eta_Q}{\|g_t\|_2^2}, \qquad g_t = \nabla_\psi Q_\psi(s_t, z_t). \tag{5}$$

This step changes the prediction by $g_t^\top(\psi^+ - \psi) = \beta_t \delta_t \|g_t\|_2^2 = \eta_Q \delta_t$, as Eq. (4) requires. With a fixed learning rate, the change would instead scale with $\|g_t\|_2^2$, which can vary widely across transitions; IU makes each update the same fraction of progress toward its target.

Incorporating IU into SAC. In SIE, IU updates the Q-function of SAC (critic $Q_\psi(s, z)$ in Section 3.2) as follows. For each incoming transition (s_t, z_t, r_t, s_{t+1}), SIE computes y_t with the SAC target of Eq. (2), and updates ψ for one gradient step by Eq. (5), instead of minimizing the mini-batch loss in Eq. (2). The actor in SIE is updated in the same way as in the vanilla SAC framework, namely, after each critic update, SIE takes one gradient step on the actor loss $\mathcal{L}_\pi(\theta; s_t)$ in Eq. (3) with the updated critic.

4.2 Episodic Policy Updates

EPU enables the actor to learn directly from success, without heavily relying on the critic. Specifically, after a successful episode terminates, it raises the actor's log-likelihood of the latent actions chosen in that episode. To this end, SIE records the observation s_t and the sampled latent actions z_t at every query of an episode; For a recorded episode $\tau = \{(s_t, z_t)\}_{t=0}^{T-1}$ we define its total log-likelihood under actor parameters θ and the total recorded during the episode:

$$\ell_\tau(\theta) = \sum_{t=0}^{T-1} \log \pi_\theta(z_t \mid s_t), \quad B_\tau = \sum_{t=0}^{T-1} \log \pi_{\theta_t}(z_t \mid s_t). \tag{6}$$

Here θ_t denotes the actor parameters used to sample z_t at s_t. Thus, $\ell_\tau(\theta)$ evaluates all recorded decisions using the same actor parameters θ, whereas B_τ sums the log-likelihoods recorded when those decisions were selected. Let θ^- denote the actor parameters after the episode's last per-transition update and before the EPU update. The log-likelihood evaluated before EPU is therefore $\ell_\tau(\theta^-)$.

Adaptive update. The magnitude of the log-likelihood increase is a key design choice of EPU. However, using the same fixed magnitude for every successful episode is undesirable, because the per-transition updates performed during the episode have already raised the log-likelihood of its latent actions. In such cases, applying a fixed magnitude of increase may cause an overly aggressive update. Specifically, let d_τ denote the actual log-likelihood increase given by EPU, η_E denote a pre-specified budget (e.g. $\eta_E = 0.05$ in our experiment), and $D_\tau = \ell_\tau(\theta^-) - B_\tau$ measure the difference of the sum of log-likelihood. EPU determines d_τ by the following *adaptive increase rule*:

$$d_\tau = S_\tau \ \mathrm{clip}\left(\eta_E - D_\tau,\, 0,\, \eta_E\right), \tag{7}$$

where S_τ is a mask that restricts EPU to successful episodes: $S_\tau = 1$ if the episode succeeds and 0 otherwise. As shown in Eq. (7), for a successful episode, EPU applies the full budget when $D_\tau \le 0$, a partial increase $\eta_E - D_\tau$ when $0 < D_\tau < \eta_E$, and no update when $D_\tau \ge \eta_E$.

The adaptive magnitude d_τ is used to determine an actual update to actor's parameter θ. Specifically, we first compute the gradient of the log-likelihood at the current actor θ^-, denoted as $g_\tau = \nabla_\theta \ell_\tau(\theta)|_{\theta=\theta^-}$. Inspired by IU, we then define the update step of θ as:

$$\Delta\theta^\star = \frac{d_\tau}{\|g_\tau\|_2^2}\, g_\tau, \tag{8}$$

Under a first-order approximation, this update increases the episode log-likelihood by $g_\tau^\top \Delta\theta^\star = d_\tau$, with d_τ controlling the targeted increase, and g_τ determining the update direction.

The direction g_τ is also closely related to the gradient of the task success probability. To see this, let $J_{\mathrm{succ}}(\theta) = \Pr_\theta(S_\tau = 1)$ denote the probability of the episode to succeed, and p_θ denote the episode distribution under π_θ. For bounded-length episodes collected under a fixed policy, with dynamics and initial states independent of θ, the likelihood-ratio identity (Williams, 1992) gives

$$\nabla_\theta J_{\mathrm{succ}}(\theta) = \mathbb{E}_{\tau\sim p_\theta}\left[S_\tau \nabla_\theta \ell_\tau(\theta)\right]. \tag{9}$$

Eq. (9) shows that the gradient of task success probability (LHS) is equal to the expected gradients of log-likelihood of successful episodes (RHS). Thus, the log-likelihood gradient of a successful episode provides a sample direction whose success-weighted expectation aligns with the gradient of the task success probability. See Appendix C.2 for further theoretical discussion.

Algorithm 1 (Appendix C.1) summarizes the procedure of SIE.

5 EXPERIMENTS

In this section, we evaluate the adaptation performance of SIE across multiple VLA models and benchmarks. We also conduct analytical experiments to examine its system overhead, continual adaptation capability, and the benefits of episodic policy updates. Throughout our experiments, the VLA backbone remains frozen, and only the latent-space learner is trained. For DSRL, we implement DSRL-SAC in PyTorch, using its official JAX implementation as a reference.

5.1 SIE IMPROVES STREAMING ADAPTATION ON LIBERO

We first examine whether SIE improves streaming adaptation of a frozen π_0 policy on a 15-task subset of LIBERO-90 (Black et al., 2026; Liu et al., 2023). The π_0 model we used here is a PyTorch checkpoint fine-tuned on short-horizon LIBERO tasks; see Appendix A.1 for details.

Baselines. We compare SIE with three variants of DSRL: *DSRL-Replay* follows the original replay-based DSRL-SAC training configuration (Wagenmaker et al., 2025). *DSRL-Stream* sets the Update-to-Data (UTD) ratio, batch size, and online buffer capacity to one and disables the target network. *DSRL-IU* adapts the IU update rule (Sharifnassab et al., 2026) to the DSRL critic in this streaming setup while retaining the vanilla SAC actor update (Haarnoja et al., 2018).

Training and Evaluation. We train each method for 500k environment steps across 3 random seeds. Learning curves show task-wise success rate against environment steps. We evaluate the final policy over 50 episodes per run and refer to its success rate as the *final-50* success rate.

Results. As shown in Figure 1, SIE achieves a mean final-50 success rate of **72.09%** across all tasks, substantially outperforming DSRL-Stream (47.20%) and DSRL-IU (46.31%). This represents an absolute improvement of **24.89 percentage points** over DSRL-Stream.

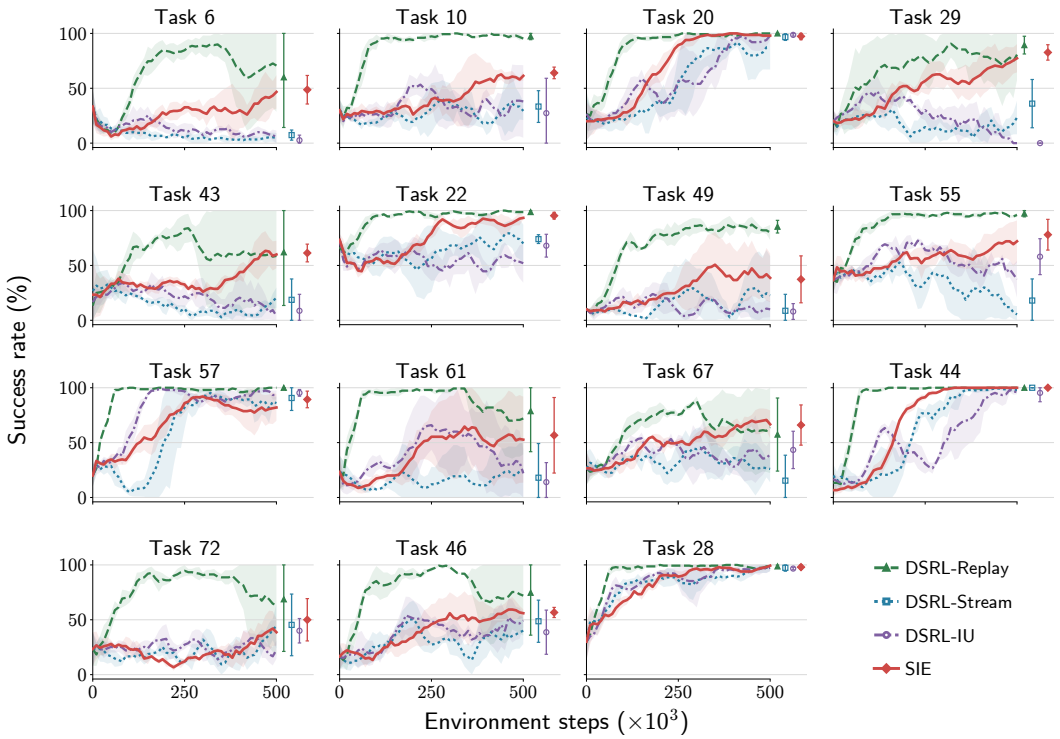


Figure 1: π_0 on 15 LIBERO-90 tasks.

5.2 SIE Improves Adaptation Across Multiple Models and Benchmarks

We next examine whether the gains of SIE over streaming baselines generalize to additional representative model backbones and benchmarks. Specifically, we evaluate $\pi_{0.5}$ on 10 LIBERO-90 tasks and 5 Meta-World tasks, and π_0 on ALOHA Transfer Cube (Physical Intelligence et al., 2025b; Zhao et al., 2023; Yu et al., 2020). For LIBERO, we use a PyTorch $\pi_{0.5}$ checkpoint fine-tuned on short-horizon LIBERO tasks. For Meta-World and ALOHA, we use the publicly released OpenPI/JAX $\pi_{0.5}$ checkpoint from FlowDAgger (Murray et al., 2026) and the official OpenPI π_0 checkpoint, respectively; see Appendix A.1 for details. Training settings for these models are the same as in Section 5.1, except that π_0 on ALOHA is trained for 2M environment steps for better convergence (following Wagenmaker et al. (2025)).

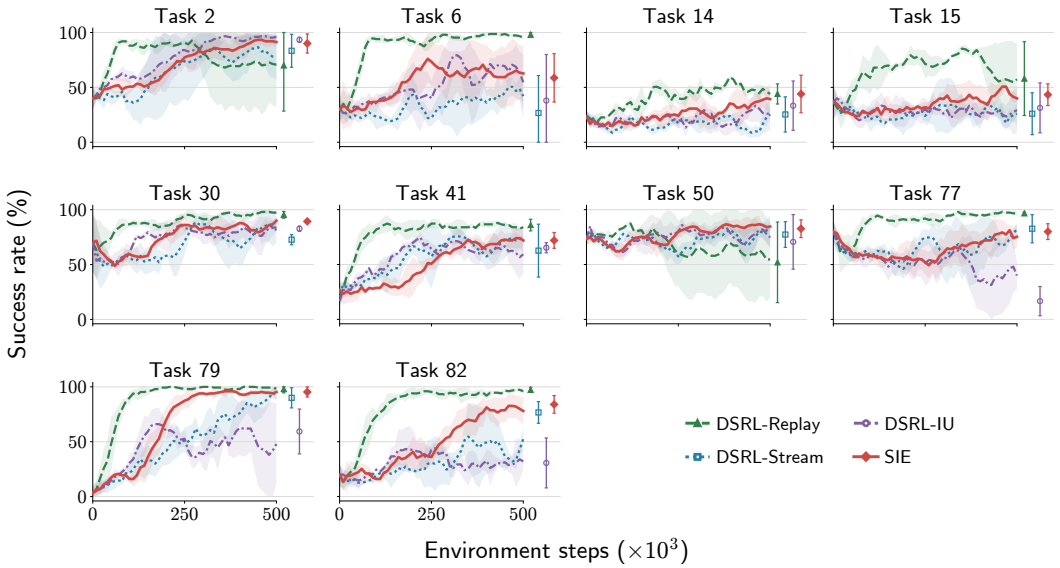


Figure 2: $\pi_{0.5}$ on 10 LIBERO-90 tasks. Curves and right-side final-50 points follow Figure 1.

Results on $\pi_{0.5}$–LIBERO. Figure 2 shows that SIE achieves a **73.93%** mean final-50 success rate, compared with 62.33% for DSRL-Stream and 52.13% for DSRL-IU. One might find that $\pi_{0.5}$ displays no marked advantage over π_0 in overall success rate. We would like to explain that this may stem from the high variance in zero-shot performance of the frozen $\pi_{0.5}$ checkpoint on the LIBERO-90 tasks. Specifically, a large portion of tasks either show near-zero success rates (making them unamenable to RL) or nearly saturated performance (leaving little room for further improvement).

Results on $\pi_{0.5}$–Meta-World and π_0–ALOHA. Figure 3(a)–(e) shows that SIE achieves a **63.07%** mean final-50 success rate on the five tasks, compared with 51.07% for DSRL-Stream and 58.53% for DSRL-IU. As shown in Figure 3(f), on ALOHA Transfer Cube, SIE reaches a **46.00%** final-50 success rate, while DSRL-Stream and DSRL-IU both yield 0% success.

5.3 SIE Improves Time and Memory Efficiency

To assess the advantage of streaming RL over replay-based RL in VLA adaptation, we conduct a lightweight simulating experiment to examine whether SIE reduces the cost of online adaptation. Specifically, to emulate a realistic VLA deployment scenario, we run the frozen π_0 backbone on an A100-equipped server (emulating a high-performance cloud server) , and the latent learner on an RTX 5060 Laptop GPU (emulating a resource-constrained edge device). We compare DSRL-Replay, SIE w/o EPU, and SIE over 1,000 queries per run. In synchronous execution, each query waits for the learner update; in asynchronous execution, learning continues in the background while the actor uses the latest weights. See Appendix A.2 for more details.

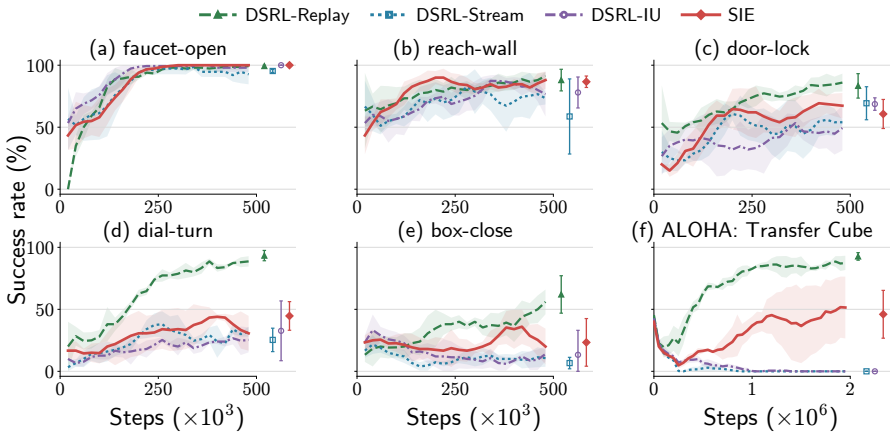


Figure 3: Adaptation on five $\pi_{0.5}$ Meta-World tasks (a–e; 500k steps training) and π_0 ALOHA Transfer Cube (f; 2M steps training).

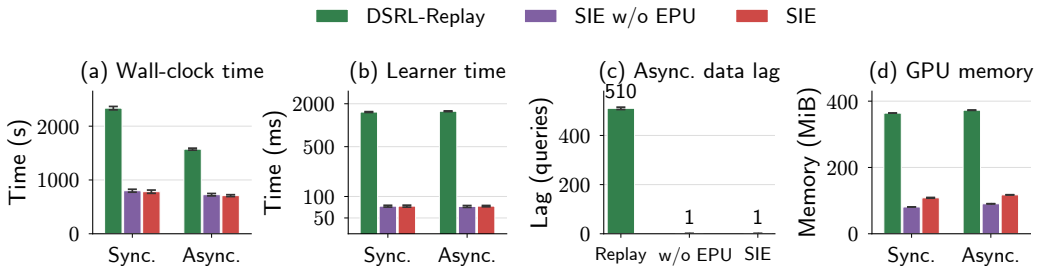


Figure 4: (a) Client wall time, including learner drain. (b) Mean learner time per query on a log scale. (c) Maximum asynchronous data lag per run: earlier query transitions not yet reflected in the actor used for sampling. (d) Peak Torch-allocated GPU memory on the learner, excluding the server and CPU storage. Panels (b,c) exclude the first 20 queries.

SIE reduces resource costs and policy lag relative to replay. Compared with DSRL-Replay, SIE reduces synchronous client wall time by **66.5%** and peak Torch-allocated learner GPU memory by **70.2%** (Figure 4). In asynchronous execution, its maximum data lag is one query in every run, versus a mean per-run maximum of 510 queries for DSRL-Replay. This highlights the clear advantage of our streaming RL formulation in runtime latency over traditional replay-based approaches.

EPU Preserves Time Efficiency. Compared with SIE w/o EPU, SIE's mean synchronous learner time is 73.3 versus 73.2 ms per query, while its peak learner GPU allocation is 34.0% higher; nevertheless, it still uses less peak learner GPU memory than DSRL-Replay by a factor of about 3.4 (Figure 4). This demonstrates that the EPU component does not noticeably compromise the efficiency advantages inherent to streaming RL.

5.4 SIE Does Not Impede Adaptation When the Task Switches

Robots deployed in real-world environments must continually adapt to dynamically switching tasks while retaining their proficiency in previously learned ones. Since EPU increases the likelihood of actions taken during successful episodes of the current task, a natural question is whether this mechanism may impede adaptation when the agent switches to a new task. We thus design an experiment to examine whether SIE can effectively adapt under this task-switching setup. Specifically, we run π_0 and $\pi_{0.5}$ on separate sequences of three LIBERO-90 tasks, with 200k environment steps per task (600k steps in total), and compare SIE with DSRL-Replay, DSRL-Stream, and DSRL-IU. At each evaluation point, we measure a success rate on all tasks encountered up to the current stage and report the mean success rate; see Appendix A.3 for more implementation details. As shown

in Figure 5, SIE achieves higher mean seen-task success than streaming baselines for both models, indicating that it maintains strong performance across dynamically switching tasks.

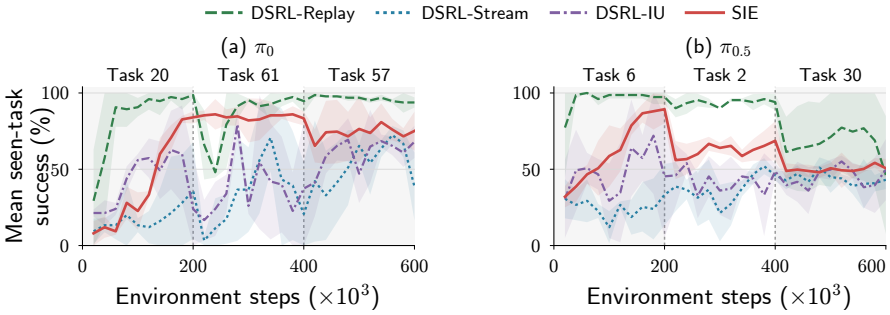


Figure 5: Continual adaptation across three 200k-step stages.

5.5 EPU Improves Performance and Raises Preference for Successes

In this subsection, we conduct an ablation study on EPU to examine its effectiveness, examine its effect on actor preference, and analyze the update fidelity ratio (see Section 7.7 in Sharifnassab et al. (2026)) and how EPU's adaptive increase rule (see Section 4.2) adapts.

Ablation of episodic policy updates. We compare SIE with *SIE w/o EPU* using the same training and evaluation settings as in Section 5.1. As shown in Table 1, EPU increases the mean final success rate from 66.3% to 72.1%, an improvement of **+5.8 percentage points**. To mitigate the influence of performance saturation and extreme values, we exclude three tasks on which both methods achieve a nearly 100% success rate (Tasks 20, 28, and 44), as well as the two tasks with the largest and smallest success-rate differences among the remaining 12 tasks. On this refined subset, EPU yields an average improvement of **8.60%**.

Table 1: EPU ablation on π_0–LIBERO: final-50 success (%), averaged over three training seeds. SIE w/o EPU is IU with $\eta_Q = 0.10$. **Avg.** equally weights all 15 tasks for both methods.

Method	**Avg.**	6	10	20	29	43	22	49
SIE w/o EPU	**66.3**	23.3	48.7	100.0	60.7	48.0	79.3	33.3
SIE	**72.1**	48.7	64.0	97.3	82.7	61.3	95.3	37.3
Method	55	57	61	67	44	72	46	28
SIE w/o EPU	68.7	96.7	78.0	58.0	100.0	46.0	55.3	98.0
SIE	78.0	89.3	56.7	66.0	100.0	50.0	56.7	98.0

EPU increases preference for decisions from successful episodes. To investigate how EPU enhances policy learning, we examine whether it directly increases the actor's relative preference for actions taken during successful episodes.

We compare EPU with two SAC actor-update strategies. (1) The *critic-mediated strategy* tests whether amplifying the critic update from a successful episode's final transition increases this preference: we scale that update's parameter change by $\kappa \in \{0, 1, 2, 4, 8\}$ and use the resulting critic for one SAC actor step ($\kappa = 0$ leaves the critic unchanged). (2) The *episode-SAC strategy* tests whether one extra SAC actor step increases this preference: after the successful episode's final per-transition actor update, we take one additional step using all its states as a single batch.

We evaluate both comparisons using the relative log-likelihood change, ΔP, defined as the change in the executed action's log-likelihood minus the mean change for 32 fixed reference actions at the same state. This metric isolates action-specific reinforcement from global shifts in policy density.

As shown in Figure 6(a), EPU consistently raises the preference for executed decisions across all decision lags prior to success ($\Delta P > 0$ for 95% of measured decisions), whereas critic-mediated and episode-SAC updates yield negligible preference changes. Even when the terminal critic update is amplified by a factor of $\kappa = 8$, Figure 6(b) demonstrates that the critic-mediated strategy's change remains an order of magnitude smaller than EPU. These results demonstrate that EPU provides a stronger and more direct reinforcement signal for successful trajectories (Appendix A.4).

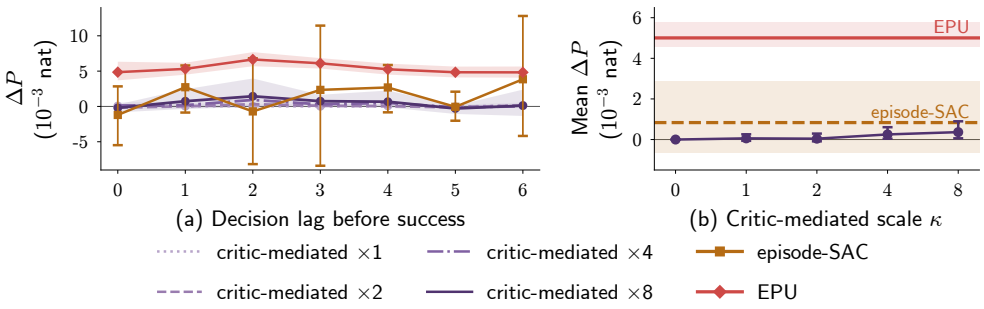


Figure 6: Relative actor-preference change ΔP. (a) Mean ΔP across decisions at each query distance from success: 0 is the final query, 1 the preceding query, and so on. (b) Mean ΔP across critic-mediated scales κ.

IU critic and EPU work as designed in SIE. IU and EPU derive their step sizes using a linear approximation, so the realized changes in critic value and actor log-likelihood may differ from their targets (Sections 4.1 and 4.2). We measure the *update fidelity ratio* (Sharifnassab et al., 2026), defined as the realized change divided by the requested change: $R_Q = \Delta Q/(\eta_Q \widetilde{\delta})$ for the critic and $F_E = \Delta \ell_\tau / d_\tau$ for EPU. Additionally, to investigate whether the adaptive increase rule for EPU (see Section 4.2) works as expected, we measure how frequently it is triggered and the extent to which it modifies the EPU update. Specifically, we report the percentages of successful episodes in which the EPU update is fully clipped, partially clipped, or not clipped, along with the mean fraction of the update removed by clipping, denoted as "Mean R_τ", where $R_\tau = 1 - d_\tau/\eta_E$ and d_τ and η_E are defined in Section 4.2. As shown in Table 2, the actual updates closely match their intended updates, and the adaptive increase rule effectively regulates the updates in EPU.

Table 2: Update fidelity ratio (a ratio of 1 means exact agreement) and effects of EPU's adaptive increase rule. Zero, partial, and full denote $R_\tau = 0$, $0 < R_\tau < 1$, and $R_\tau = 1$. Numbers here are calculated with data of runs on LIBERO tasks in Sections 5.1 and 5.2.

Model	Update	Update fidelity ratio	Zero (%)	Partial (%)	Full (%)	Mean R_τ
π_0	IU critic	0.920 ± 0.171	–	–	–	–
	EPU	0.951 ± 0.074	61.61	7.78	30.62	0.345
$\pi_{0.5}$	IU critic	0.908 ± 0.201	–	–	–	–
	EPU	0.947 ± 0.079	64.30	7.31	28.39	0.319

6 Conclusion and Discussion

We presented SIE, a streaming RL framework for online adaptation of frozen VLAs. SIE combines IU-inspired critic learning with Episodic Policy Updates (EPU) that directly reinforce decisions from successful episodes. It improves adaptation performance over streaming baselines in the evaluated simulation settings and reduces client wall time relative to the replay-based method in the two-device benchmark.

Real-robot adaptation. Our evaluation is limited to simulation. A next step is to evaluate SIE on physical robots under noisy observations, control delays, and limited interaction budgets, while assessing the memory and latency costs of episodic updates during deployment.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods, assist with translation, and design or provide feedback on research methodology or experiments. Additionally, we used them to create or edit software code, polish the writing, brainstorm, summarize or analyse existing literature, identify relevant literature, and format references. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, or interpret results, and the remaining tasks requiring disclosure (such as generating synthetic datasets) are not applicable to this work.

We have reviewed all AI-assisted work: we checked LLM-generated research ideas for potential plagiarism through manual review; we verified the accuracy of AI-suggested literature summaries and recommendations and confirmed that the cited references exist; and LLM-generated code was verified and tested for correctness by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies resource-efficient online adaptation of pretrained VLA models. All experiments reported in this paper were conducted in simulation using existing benchmarks and pretrained checkpoints. Deployment on physical robots would require application-specific safety evaluation and safeguards, including human oversight and mechanisms for stopping unsafe actions.

REPRODUCIBILITY STATEMENT

Section 4 presents the method and update rules, with implementation details and supporting theoretical analysis in Appendix C. Appendix A documents the pretrained models and fine-tuning setup, evaluated tasks, training and evaluation protocols, resource measurements, and update diagnostics, including the procedures for aggregating results. Appendix B provides the learner architectures and hyperparameters. The source code is included in the supplementary materials accompanying this submission.

REFERENCES

Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement – residual RL for precise assembly, 2024. URL https://arxiv.org/abs/2407.16677.

Kevin Black, Noah Brown, Danny Driess, et al. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

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, 2024. URL https://arxiv.org/abs/2303.04137.

Riccardo De Monte, Matteo Cederle, and Gian Antonio Susto. Towards batch-to-streaming deep reinforcement learning for continuous control, 2026. URL https://arxiv.org/abs/2603.08588.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

Mohamed Elsayed, Elena Sorina Lupu, Gautham Vasan, and A. Rupam Mahmood. Streaming deep reinforcement learning finally works, 2026. URL https://arxiv.org/abs/2410.14606.

Tongyan Fang, Siyuan Huang, Naiyu Fang, Ganlong Zhao, Zhongjin Luo, Jianbo Liu, Xiaogang Wang, Ying Dong, and Hongsheng Li. Hierarchical advantage weighting for online RL fine-tuning of VLAs from sparse episode outcomes, 2026. URL https://arxiv.org/abs/2606.17043.

Noah Farr, Aryaman Reddi, Carlo D'Eramo, and Jan Peters. Streaming reinforcement learning under partial observability with real-time recurrent learning, 2026. URL https://arxiv.org/abs/2605.24709.

Senyu Fei, Siyin Wang, Li Ji, Ao Li, Shiduo Zhang, Liming Liu, Jinlong Hou, Jingjing Gong, Xianzhong Zhao, and Xipeng Qiu. SRPO: Self-referential policy optimization for vision-language-action models, 2025. URL https://arxiv.org/abs/2511.15605.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, et al. Soft actor-critic algorithms and applications, 2019. URL https://arxiv.org/abs/1812.05905.

Yinsen Jia and Boyuan Chen. Temporal self-imitation learning, 2026. URL https://arxiv.org/abs/2606.19752.

Yunfei Li, Tian Gao, Jiaqi Yang, Huazhe Xu, and Yi Wu. Phasic self-imitative reduction for sparse-reward goal-conditioned reinforcement learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 12765–12781. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/li22g.html.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791. Curran Associates, Inc., 2023. doi: 10.52202/075280-1939. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/8c3c666820ea055a77726d66fc7d447f-Paper-Datasets_and_Benchmarks.pdf.

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, 2025. URL https://arxiv.org/abs/2505.18719.

Junjie Lu, Xinyao Qin, Yuhua Jiang, Kaixin Wang, Chuheng Zhang, Bin Liang, Jun Yang, Min Xu, and Li Zhao. UniSteer: Unified noise steering for efficient human-guided VLA adaptation, 2026. URL https://arxiv.org/abs/2605.10821.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/3d43cc5692bf68944ee7cd31b97d0c11-Abstract-Conference.html.

Michael Murray, Daphne Chen, Simran Bagaria, Dean Fortier, Tess Hellebrekers, Galen Mullins, Harshavardhan Gajarla, Oier Mees, Maya Cakmak, and Andrey Kolobov. FlowDAgger: Human-in-the-loop adaptation of generative robot policies in latent space, 2026. URL https://arxiv.org/abs/2607.08877.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. *arXiv preprint arXiv:2410.13816*, 2024.

Nilaksh, Antoine Clavaud, Mathieu Reymond, François Rivest, and Sarath Chandar. Squeezing more from the stream : Learning representation online for streaming reinforcement learning, 2026. URL https://arxiv.org/abs/2602.09396.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3878–3887. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/oh18b.html.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning, 2019. URL https://arxiv.org/abs/1910.00177.

Physical Intelligence, Ali Amin, Raichelle Aniceto, et al. $\pi^*_{0.6}$: a VLA that learns from experience, 2025a. URL https://arxiv.org/abs/2511.14759.

Physical Intelligence, Kevin Black, Noah Brown, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025b. URL https://arxiv.org/abs/2504.16054.

Physical Intelligence, Bo Ai, Ali Amin, et al. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities, 2026. URL https://arxiv.org/abs/2604.15483.

Allen Z. Ren, Justin Lidard, Lars L. Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization, 2024. URL https://arxiv.org/abs/2409.00588.

Daniel Russo. Success conditioning as policy improvement: The optimization problem solved by imitating success, 2026. URL https://arxiv.org/abs/2601.18175.

Arsalan Sharifnassab, Mohamed Elsayed, Kris De Asis, A. Rupam Mahmood, and Richard S. Sutton. Intentional updates for streaming reinforcement learning, 2026. URL https://arxiv.org/abs/2604.19033.

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

Andy Tang, William Chen, Andrew Wagenmaker, Chelsea Finn, and Sergey Levine. Improving robotic generalist policies via flow reversal steering, 2026. URL https://arxiv.org/abs/2606.13675.

Gerald Tesauro. Temporal difference learning and TD-Gammon. *Commun. ACM*, 38(3):58–68, March 1995. ISSN 0001-0782. doi: 10.1145/203330.203343. URL https://doi.org/10.1145/203330.203343.

Gautham Vasan, Mohamed Elsayed, Alireza Azimi, Jiamin He, Fahim Shariar, Colin Bellinger, Martha White, and A. Rupam Mahmood. Deep policy gradient methods without batch updates, target networks, or replay buffers. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 845–891. Curran Associates, Inc., 2024. doi: 10.52202/079017-0026. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/019ef89617d539b15ed610ce8d1b76e1-Paper-Conference.pdf.

Andrew Wagenmaker, Yunchu Zhang, Mitsuhiko Nakamoto, 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*, volume 305 of *Proceedings of Machine Learning Research*, pp. 258–282. PMLR, 2025. URL https://proceedings.mlr.press/v305/wagenmaker25a.html.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992. URL https://www.gatsby.ucl.ac.uk/~dayan/papers/cjch.pdf.

Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. Episodic reinforcement learning by logistic reward-weighted regression. In *International Conference on Artificial Neural Networks*, pp. 407–416. Springer, 2008.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992. URL https://doi.org/10.1007/BF00992696.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (eds.), *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1094–1100. PMLR, 30 Oct–01 Nov 2020. URL https://proceedings.mlr.press/v100/yu20a.html.

Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Zhang Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. In *International Conference on Learning Representations*, volume 2025, pp. 28129–28164, 2025.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.016.

A Experimental Details

A.1 Adaptation across models and benchmarks

Models and tasks. All pretrained VLAs remain frozen during adaptation; only the latent actor, critic, and temperature are trained. For LIBERO, we fine-tune LeRobot's public `pi0_base`[1] and `pi05_base`[2] checkpoints on the public LIBERO dataset[3] for 30,000 optimization steps each. The π_0 model uses a batch size of 32 on one NVIDIA A100 80 GB GPU; $\pi_{0.5}$ uses a global batch size of 64 across four NVIDIA A100 80 GB GPUs (16 samples per GPU).

The LIBERO task IDs, in figure order, are 6, 10, 20, 29, 43, 22, 49, 55, 57, 61, 67, 44, 72, 46, and 28 for π_0; and 2, 6, 14, 15, 30, 41, 50, 77, 79, and 82 for $\pi_{0.5}$. The task subsets were selected after observing experimental results. All four methods have complete three-seed results on all 15 π_0 tasks. The EPU ablation uses the same 15 tasks and seeds.

Meta-World uses the public OpenPI/JAX checkpoint `pi05-metaworld`[4] on faucet-open-v3, reach-wall-v3, door-lock-v3, dial-turn-v3, and box-close-v3. ALOHA uses the official OpenPI checkpoint `pi0_aloha_sim`[5] on `AlohaTransferCube-v0`. Main adaptation experiments use NVIDIA A100 80 GB GPUs. Tables 3 and 4 summarize the learner architecture, training settings, and benchmark differences.

Training and evaluation. A transition corresponds to one latent-policy query. Non-success queries receive reward -1; terminal success receives 0. Each method uses training seeds 0, 1, and 2, seeded training resets, and stochastic evaluation with fixed action RNG seeds.

LIBERO curves include a 10-episode evaluation at step zero and every 10k steps. Meta-World uses ten episodes at each common 20k-step point from 20k to 480k, without a step-zero point; ALOHA uses 25 episodes at step zero and every 50k steps through 1.95M. We smooth each seed with a trailing five-evaluation mean, then report the three-seed mean and sample standard deviation (SD). Separate final evaluations use 50 episodes per task and seed at the training budget endpoint and are not smoothed. Shaded bands and error bars show sample SD, clipped to 0–100%.

A.2 Resource measurements

The A100 server runs the frozen π_0 and LIBERO task 57; an 8 GB RTX 5060 Laptop GPU runs the image-only learner. DSRL-Replay, SIE w/o EPU, and SIE each use seeds 0–2 in synchronous and asynchronous execution, for 1,000 queries per run, up to 20 environment actions per query, and a 400-step limit per episode. All methods begin updating after the first query; the first 20 queries are excluded only from per-query timing and lag statistics.

[1] https://huggingface.co/lerobot/pi0_base
[2] https://huggingface.co/lerobot/pi05_base
[3] https://huggingface.co/datasets/HuggingFaceVLA/libero
[4] https://huggingface.co/mmurray-ms/pi05-metaworld
[5] gs://openpi-assets/checkpoints/pi0_aloha_sim

Client wall time includes learner drain but excludes initialization and checkpoint saving. Learner time includes ordinary updates, EPU processing, and weight publication. Data lag counts earlier transitions not yet incorporated into the published actor at sampling time; Figure 4(c) uses each run's maximum after warmup. GPU memory is peak Torch allocation on the learner, excluding CPU episode/queue storage and server memory. Bars show three-seed means and sample SD of per-run values.

A.3 Task-switch protocol

Task orders are $20 \to 61 \to 57$ for π_0 and $6 \to 2 \to 30$ for $\pi_{0.5}$, with 200k environment steps per task and seeds 0–2. Model parameters, optimizer state, and RNG state carry across switches. When task switches occur, IU critic's traces reset, while RMS and scale statistics remain; DSRL-Replay retains its replay buffer. Every 20k steps, we evaluate 25 episodes per seen task and average across tasks and three seeds, without smoothing. Final results use separate 50-episode evaluations per task and seed. Mean seen-task success combines learning new tasks and retaining earlier ones. In our experiment, it is observed that two $\pi_{0.5}$ DSRL-Replay runs exhibit positive-Q divergence with exploding critic losses and gradients during the final task.

A.4 Update diagnostics

Actor-preference diagnostics. We use π_0 on LIBERO tasks 29 and 57, seeds 0–2, over 400k–420k-step continuation windows. For each recorded decision, we measure changes in

$$P_\theta(s_t, z_t) = \log \pi_\theta(z_t \mid s_t) - \frac{1}{32}\sum_{j=1}^{32} \log \pi_\theta(z_{t,j}^{\mathrm{ref}} \mid s_t),$$

using fixed reference latents drawn from the behavior policy. The critic intervention restores the same pre-update actor, Adam state, temperature, and actor RNG, then takes one actor step with $\psi_\kappa = \psi_{\mathrm{before}} + \kappa(\psi_{\mathrm{after}} - \psi_{\mathrm{before}})$, $\kappa \in \{0, 1, 2, 4, 8\}$. Changes are relative to the $\kappa = 0$ branch.

Update fidelity and EPU budget clipping. We analyze the SIE runs on the LIBERO task sets used in the main performance figures: 15 tasks for π_0 (6, 10, 20, 29, 43, 22, 49, 55, 57, 61, 67, 44, 72, 46, 28) and ten for $\pi_{0.5}$ (2, 6, 14, 15, 30, 41, 50, 77, 79, 82), each with three seeds. Critic responses use the latest individual update sampled every 500 environment steps; EPU records each episode end. R_Q divides the ensemble-mean prediction change by the signed, clipped intended change $\eta_Q \widetilde{\delta}$. F_E compares the realized log-likelihood increase with the adaptive increase d_τ determined by Eq. (7), using the original d_τ before backtracking. Response statistics omit absolute denominators at most 10^{-4} and exclude skipped EPU updates. We compute mean and sample SD within each run, then average them equally across tasks and seeds. Varying the denominator threshold from 10^{-6} to 10^{-3} changes these mean responses by less than 0.001. Accepted EPU updates satisfy the finite-step bound in Eq. (11).

B Hyperparameters

Table 3: Shared latent-learner settings for the main adaptation experiments.

Parameter	Value	Parameter	Value
Latent dimension	32	Actor hidden widths	$[128, 128, 128]$
Number of critics	10	Critic hidden widths	$[128, 128, 128]$
Critic aggregation	Mean	Learner image size	64×64
CNN channels	$[32, 32, 32, 32]$	CNN strides	$[2, 1, 1, 1]$
Image bottleneck	50	Per-step discount	0.999
Actor learning rate	10^{-4}	Critic learning rate	3×10^{-4}
Temperature learning rate	3×10^{-4}	Maximum log std	2
IU trace decay	0.8	IU RMS decay	0.999
IU clipping multiplier	20	EPU budget η_E	0.05

Table 4: Method and benchmark settings. Warm-up counts query transitions.

Parameter	DSRL-Replay	DSRL-Stream	DSRL-IU	SIE
Batch size	256	1	1	1
Updates per transition	20	1	1	1
Replay capacity	100k	—	—	—
Critic optimizer	Adam	Adam	IU	IU
Critic scale η_Q	—	—	0.50	0.10
Target network	Yes	No	No	No
Target update rate	0.005	—	—	—
Image augmentation	Yes	No	No	No
Warm-up queries	$500^\dagger$	1	1	1
Parameter	**LIBERO π_0**	**LIBERO $\pi_{0.5}$**	**Meta-World**	**ALOHA**
Training steps	500k	500k	500k	2M
Query interval	20	20	10	50
Episode limit (steps)	400	400	400	300
RL robot-state dimension	0	0	4	14
Latent magnitude bound	1	1	1	2
Target entropy	-16	-16	-16	0

†DSRL-Replay uses 1,000 warm-up queries on ALOHA; in the resource benchmark, its update-start threshold is 1 query; hidden-layer normalization is disabled and the minimum log standard deviation is -5, except for Meta-World DSRL-Replay, which uses hidden LayerNorm and a minimum of -20.

C Additional Method Details

C.1 Update implementation

Algorithm 1 Streaming with Intentional and Episodic Updates (SIE)

Inputs: Frozen F_ϕ; actor θ, critic ψ, temperature α; scales η_Q, η_E.

Streaming interaction and learning

Initialize learner; reset to s
$\tau \leftarrow \varnothing$; $B \leftarrow 0$
while budget remains **do**
  Sample $z \sim \pi_\theta(\cdot \mid s)$
  Append (s, z) to τ
  $B \leftarrow B + \log \pi_\theta(z \mid s)$
  Execute $F_\phi(s, z)$ prefix
  Observe r, s', m and outcome
  Intentional critic step (5)
  Actor step using (3), then update α
  if episode ended **then**
    $\theta \leftarrow \mathrm{EPU}(\theta, \tau, B, S_\tau)$
    Clear τ, B and critic traces
    Reset to s
  else
    $s \leftarrow s'$
  end if
end while

EPU(θ, τ, B, S)

if $S = 0$ **then**
  return θ
end if
$D \leftarrow \ell_\tau(\theta) - B$
$d \leftarrow \mathrm{clip}(\eta_E - D, 0, \eta_E)$
if $d = 0$ **then**
  return θ
end if
$g \leftarrow \nabla_\theta \ell_\tau(\theta)$
if sensitivity is invalid **then**
  return θ
end if
$\Delta\theta \leftarrow d\, g / \|g\|_2^2$
return GuardedStep$(\theta, \Delta\theta, d, \tau)$

GuardedStep verifies the increment as in Eq. (11). Episodes containing unscored initialization actions skip EPU.

Intentional critic. Section 4.1 presents the basic IU rule; its prediction-change relation holds to first order. The critic $Q_\psi(s, z)$ is the mean of an ensemble of critic heads (Table 3), and IU treats this mean as a single scalar prediction. The optimizer receives $\nabla_\psi Q_\psi(s_t, z_t)$ and the TD error δ_t separately, rather than the gradient of a squared TD loss, and combines eligibility traces, diagonal RMS scaling, a running normalizer of the update magnitude, and adaptive TD-error clipping (Sharifnassab et al., 2026). With traces active, this normalization controls a decaying aggregate of predic-

tion changes, so $\eta_Q \widetilde{\delta}_t$ is not an exact constraint on the current prediction. We retain this update rule but sample the next latent from the current actor, rather than using the maximizing-action target of Intentional Q-learning. Critic eligibility traces are cleared at episode boundaries.

Minimum-norm policy update. For $g_\tau \neq 0$, Eq. (8) gives the unique solution to

$$\min_{\Delta\theta} \ \tfrac{1}{2}\|\Delta\theta\|_2^2 \quad \text{subject to} \quad g_\tau^\top \Delta\theta = d_\tau. \tag{10}$$

Finite-step verification. The adaptive increase d_τ determines the parameter update in Eq. (8). Since the proposal uses a local approximation, we evaluate its realized log-likelihood increment before accepting it. Starting at unit scale, the guard seeks a step satisfying

$$\theta^+ = \theta^- + c_\tau \Delta\theta^\star, \qquad 0 < \ell_\tau(\theta^+) - \ell_\tau(\theta^-) \leq d_\tau + \varepsilon_\tau, \quad 0 < c_\tau \leq 1, \tag{11}$$

where the production tolerance is $\varepsilon_\tau = \max(10^{-12}, 10^{-4} d_\tau)$. Positive underfill is accepted; overshoot is reduced by bisection. Nonpositive or non-finite initial increments trigger successive step halving. If no admissible step is found, the actor is left unchanged. This check controls the recorded latent log-likelihood, not a global KL distance or task-return improvement.

C.2 Relation to the success gradient and success conditioning

Where EPU departs from Eq. (9). Eq. (9) assumes that all decisions in an episode are sampled from one fixed policy π_θ, and it describes the gradient averaged over many such episodes. EPU differs from this setting in two ways. First, SIE updates the actor after every transition, so the decisions within one episode are sampled by a sequence of slightly different actors θ_t rather than by a single policy. Second, EPU scales each step to reach the adaptive increase d_τ, so its step size depends on the episode itself, through D_τ and $\|g_\tau\|_2$, rather than being fixed in advance. Eq. (9) therefore motivates the direction of EPU: raising the likelihood of decisions from successful episodes favors, in expectation, decisions with a higher success probability. However, the EPU update is not an unbiased estimate of $\nabla_\theta J_{\mathrm{succ}}$, and a single EPU update is not guaranteed to increase the success probability.

Exact success conditioning. Here $Q^\pi_{\mathrm{succ}}(s, z)$ denotes the probability of eventual success after taking latent action z in state s and subsequently following π. Under the fixed-policy assumptions of Eq. (9), success-weighted fitting also admits a policy-level interpretation. With the sampling policy π_θ fixed, maximizing $\mathbb{E}_{\tau\sim p_\theta}[S_\tau \sum_{t<T} \log \widehat{\pi}(z_t \mid s_t)]$ over unrestricted state-conditional policies yields (Russo, 2026)

$$\pi_+(z \mid s) = \pi_\theta(z \mid s) \frac{Q^{\pi_\theta}_{\mathrm{succ}}(s, z)}{V^{\pi_\theta}_{\mathrm{succ}}(s)}, \qquad V^\pi_{\mathrm{succ}}(s) = \mathbb{E}_{z\sim\pi(\cdot|s)}[Q^\pi_{\mathrm{succ}}(s, z)], \tag{12}$$

on visited states with $V^{\pi_\theta}_{\mathrm{succ}}(s) > 0$. Here $V^\pi_{\mathrm{succ}}(s)$ is the policy's average success probability at s: conditioning increases the density of actions with above-average success probability and decreases that of actions with below-average success probability. Time or history is included in s when needed.

For the same two reasons, EPU does not implement this exact operator and does not inherit its improvement guarantee. Moreover, EPU targets the success probability, whereas the critic estimates the discounted return defined in Section 3.1; the two objectives are related but not identical.