ELASTIC QUERIES REINFORCEMENT LEARNING: SELF-AWARE POLICY EXECUTION FOR VLA MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

A generative policy acting in a closed loop must decide what action to produce, how much inference to spend producing it, and when to observe again. Flow-based vision-language-action (VLA) policies typically fix the latter two choices across an entire task, even as the value of computation and feedback changes across states. We formulate each policy query as a variable-duration decision over latent input, denoising steps, and execution length, and introduce Elastic Queries Reinforcement Learning (EQRL) to learn these choices while keeping the VLA frozen. A lightweight actor jointly learns latent steering and two schedule residuals around a difficulty-guided reference, using duration-aware query-level RL with a number-of-function-evaluations (NFE) objective. A local compute–feedback analysis distinguishes schedules with equal amortized inference cost. On the long-horizon LIBERO-10 task of putting both moka pots on the stove, EQRL raises five-seed success from 0.411 to 0.867 while reducing requested NFE from 0.500 to 0.441. On four LIBERO-90 tasks and ALOHA Cube Transfer, EQRL improves aggregate success and learning AUC while reducing requested NFE by 39.4% and 14.0% relative to DSRL. Scheduling ablations support the contributions of joint control and difficulty guidance, and physical-robot evaluations match or exceed DSRL's success while reducing inference cost and execution time across five tasks.

1 INTRODUCTION

Generative policies increasingly serve as components of closed-loop decision systems: they turn an observation into an action sequence, execute part of that sequence, and are queried again after new feedback. A policy query therefore commits two resources beyond the action itself: inference work before acting and the time spent acting before the next observation. Vision-language-action (VLA) models make this tension concrete. Flow-based VLAs such as π_0 generate temporally coherent action chunks from multimodal context (Brohan et al., 2022; Driess et al., 2023; Zitkovich et al., 2023; Kim et al., 2024; Black et al., 2024; Lipman et al., 2022), yet deployments commonly fix both the denoising budget and the number of actions executed before replanning (Zhao et al., 2023; Chi et al., 2025).

These choices are coupled. Spending more denoising steps can refine the next chunk, while executing a shorter prefix obtains fresh information sooner. Both increase inference work per action, but they need not produce the same behavior at the same average cost. Manipulation exposes this distinction as free-space approach alternates with contact, alignment, recovery, and precise placement. A fixed schedule imposes one inference–feedback operating point on all of these phases.

A natural way to adapt a pretrained generative action model is to keep the model frozen and learn a lightweight policy in its latent space. This preserves the behavioral prior of the VLA model and avoids expensive full-model fine-tuning. However, latent steering alone still leaves the execution schedule outside the learning problem. The adaptor may choose which latent action trajectory to generate, but the denoising budget and action-chunk length remain manually chosen hyperparameters. As a result, the learned policy cannot decide whether a given state deserves more computation, less computation, shorter commitment, or longer amortization.

EQRL makes each VLA query elastic. At every query, a lightweight RL adaptor selects a joint latent-schedule action (w, K, C): w steers the frozen sampler, K controls denoising effort, and C

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.

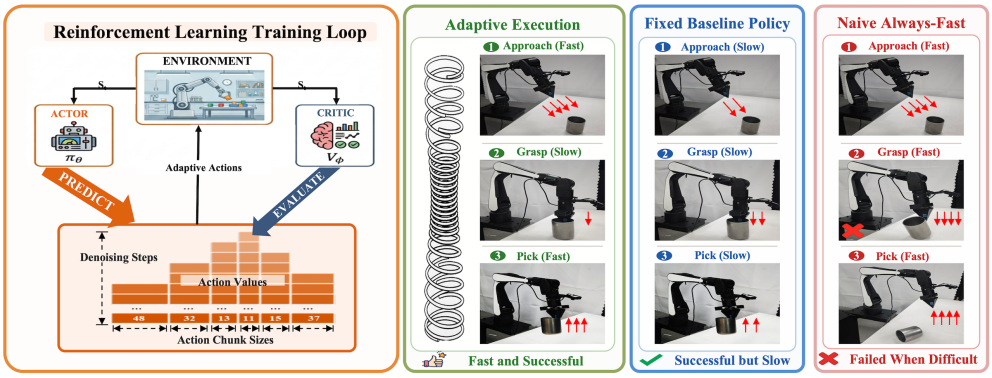


Figure 1: **Motivation for elastic querying.** EQRL adapts both denoising steps and execution chunk length across a task, moving quickly through easy phases while spending more feedback and compute near difficult contact phases.

controls how many generated actions are executed before replanning. These decisions couple behavior generation with its execution: K changes the computation within a query, while C determines when fresh feedback becomes available. Even schedules with the same amortized NFE, K/C, can therefore have different control behavior.

A local analysis separates the value of better sampling from the value of earlier feedback and shows why the two schedule coordinates should be adjusted separately. EQRL learns these adjustments jointly with latent steering around a critic-guided reference schedule. Task reward decides how the latent action and schedule work together.

EQRL trains at the query level with duration-aware backups. An amortized NFE objective regularizes overall compute, while the learned query policy allocates it across states.

This work contributes a query-level formulation of adaptive generative-policy execution, an analysis that separates sampling accuracy from feedback delay at a fixed amortized compute price, and a learning method that allocates those resources with task reward. The experiments examine whether elastic queries improve the success–compute outcome, whether both schedule coordinates contribute, and whether critic disagreement identifies states where a schedule change has value. Long-horizon manipulation exposes the success effect, while near-saturated tasks and physical execution expose the computational effect.

2 Related Work

VLA models and post-training. VLA models combine perception, language, and action (Brohan et al., 2022; Driess et al., 2023; Zitkovich et al., 2023), acquiring reusable behavioral priors from large-scale robot data and vision-language pretraining (O'Neill et al., 2024; Team et al., 2024; Kim et al., 2024; Black et al., 2024). Their reliance on offline imitation motivates post-training with task-level feedback under deployment shifts (Yang et al., 2023).

RL for generative robot policies. RL post-training includes online optimization, residual adaptation, and lightweight modules (Lu et al., 2025; Li et al., 2025; Guo et al., 2025; Liu et al., 2026b; Tan et al., 2025; Xiao et al., 2025). For diffusion and flow-based action generators (Black et al., 2024; Chi et al., 2025; Janner et al., 2022), latent steering adapts behavior without updating the generator (Xiao et al., 2025; Wagenmaker et al., 2025). EQRL extends latent steering to jointly learn the frozen VLA's denoising effort and execution chunk length.

Adaptive computation and temporal abstraction. Efficient generative samplers reduce neural function evaluations (Song et al., 2020; Lu et al., 2022; Song et al., 2023; Liu et al., 2022; Lipman et al., 2022; Prasad et al., 2024; Yu et al., 2025b; Clemente et al., 2026), while action chunking and temporal abstraction govern when feedback arrives (Sutton et al., 1999; Zhao et al., 2023; Sendai

et al., 2025; Tang et al., 2025). EQRL couples these computation and feedback decisions through amortized inference cost and optimizes them with task reward.

3 METHOD

3.1 ELASTIC QUERIES OF A FROZEN VLA

Given a task-conditioned observation o, a frozen flow-based VLA generates an action chunk

$$A = F_\theta^K(o; w) = (a^0, \ldots, a^{H-1}) \in \mathcal{A}^H,$$

where w is the latent/noise input and K is the number of sampling steps. At query j, beginning at environment time t_j, EQRL selects (w_j, K_j, C_j) and executes the first $C_j \le H$ actions, or $L_j \le C_j$ if the episode terminates earlier. Thus w selects behavior, K determines sampling effort, and C determines the commitment before fresh feedback. Only the lightweight adaptor is trained; the VLA parameters remain frozen.

Compute per action. A query spends K_j function evaluations on C_j requested actions. Over an episode, the requested amortized NFE is

$$\bar{\rho}_{\text{ep}} = \frac{\sum_j K_j}{\sum_j C_j} = \sum_j \underbrace{\frac{C_j}{\sum_i C_i}}_{\text{action fraction}} \frac{K_j}{C_j}. \tag{1}$$

This action-weighted rate makes schedules with different query frequencies comparable. A penalty on K_j alone omits how many actions share that cost, while an unweighted mean of K_j/C_j gives equal weight to chunks of different lengths. When a chunk terminates early, the executed-action rate uses L_j in the denominator instead (Appendix A.4).

3.2 ANALYZING THE COMPUTE–FEEDBACK TRADE-OFF

Equal NFE does not imply equivalent execution: $(K, C) = (10, 20)$ and $(5, 10)$ both cost 0.5, but the latter obtains feedback twice as often and uses fewer steps per query. To isolate this distinction, hold the observation and latent input fixed and consider a local continuous surrogate,

$$\ell_o(K, C) = \underbrace{\frac{a_o}{K}}_{\text{sampling error}} + \underbrace{b_o C}_{\text{feedback delay}} + \underbrace{\eta \frac{K}{C}}_{\text{compute price}}, \qquad a_o, b_o, \eta > 0. \tag{2}$$

The first term captures the gain from refining a numerical sampler, the second captures error accumulated before the next observation, and the third prices computation per requested action. Appendix A.4 derives this local model; task reward, rather than the surrogate, trains EQRL.

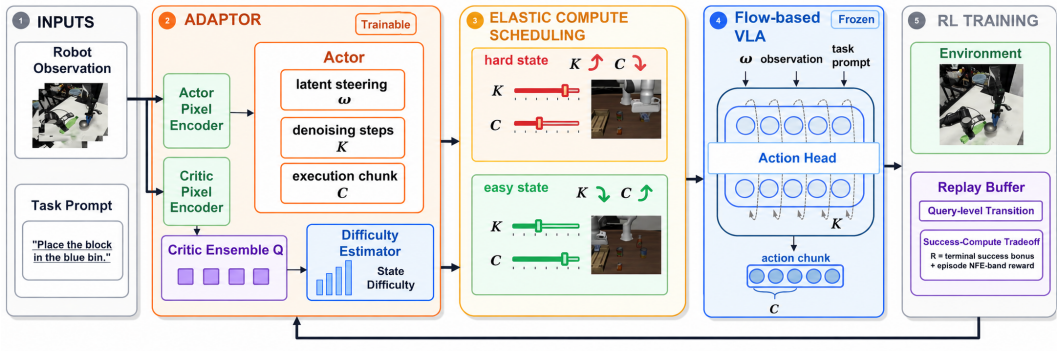


Figure 2: **EQRL architecture.** The lightweight actor learns latent steering and separate residuals for denoising effort and execution length while keeping the VLA frozen. The residuals adjust a critic-guided reference schedule.

Proposition 1 (Local schedule balance) *Over positive continuous K, C, the unique minimizer of Equation* (2) *is*

$$K_o^* = \left(\frac{a_o^2}{\eta b_o}\right)^{1/3}, \qquad C_o^* = \left(\frac{\eta a_o}{b_o^2}\right)^{1/3}. \tag{3}$$

If $a_o = a_0\xi$ and $b_o = b_0\xi$ for $\xi > 0$, then $K^* \propto \xi^{1/3}$ and $C^* \propto \xi^{-1/3}$.

The optimum separates compute intensity from the balance of error sources:

$$\frac{K_o^*}{C_o^*} = \left(\frac{a_o b_o}{\eta^2}\right)^{1/3}, \qquad K_o^* C_o^* = \frac{a_o}{b_o}. \tag{4}$$

Under the common scaling above, the optimum shifts toward $K \uparrow, C \downarrow$. If only sampling error grows, the optimum increases both K and C; if only feedback sensitivity grows, it decreases both. These distinct responses motivate separate learned adjustments around a shared reference schedule. Critic disagreement provides an empirical proxy for constructing this reference, evaluated through difficulty diagnostics and schedule interventions (Section 4.3). Appendix A.4 gives the proof and the general positive-exponent case.

3.3 Learning the Query Policy

A lightweight RL actor jointly learns latent steering and state-dependent schedule adjustments for the frozen VLA. Its three stochastic heads sample $u_j = (w_j, z_j^K, z_j^C) \sim \pi_\phi(\cdot \mid o_{t_j})$, with $z_j^K, z_j^C \in [-1, 1]$. The schedule outputs map to bounded residuals $\Delta K_j, \Delta C_j$ relative to the base schedule (K_0, C_0) (Appendix A.1). The continuous schedule combines these learned residuals with a reference schedule:

$$\hat{K}_j = K_j^p + \lambda_{r,j} g_j \Delta K_j, \qquad \hat{C}_j = C_j^p + \lambda_{r,j} g_j \Delta C_j,$$

where the difficulty prior provides the reference (K_j^p, C_j^p), $g_j \in [0, 1]$ gates the residuals as training progresses, and $\lambda_{r,j}$ sets their scale.

We clip the composed schedules to the current curriculum bounds and round them for interaction. The frozen VLA receives (o_{t_j}, w_j, K_j) and the robot executes up to C_j actions. An ensemble of M critics evaluates the joint action $\tilde{u}_j = [w_j, \bar{K}_j, \bar{C}_j]$, where bars denote the clipped continuous schedule coordinates normalized to $[-1, 1]$ using the fixed global bounds. These continuous inputs allow value gradients to train both residuals jointly with latent steering through the query-level objective in Section 3.4.

Reference schedule. Let $w_\pi(o) = \mathrm{mode}(\pi_w(\cdot \mid o))$. The difficulty prior uses critic disagreement measured at the base schedule:

$$q_{\mathrm{std}}(o) = \mathrm{Std}_{i=1}^{M}\left[Q_\psi^{(i)}\left(o, [w_\pi(o), \bar{K}_0, \bar{C}_0]\right)\right]. \tag{5}$$

Evaluating a common schedule provides a reference for comparing states. We normalize this empirical signal to a signed score $s(o) \in [-1, 1]$ and map it to (K_j^p, C_j^p), with greater sampling effort and shorter execution at higher difficulty. Appendix A.1 specifies the normalization, mapping, and gates.

3.4 Query-Level Learning and Compute Allocation

Executing a chunk induces a macro-transition with a variable duration, as in temporally abstract RL (Sutton et al., 1999). Let $\gamma \in (0, 1)$ be the discount per low-level action. The replay buffer stores $(o_{t_j}, \tilde{u}_j, R_j, o_{t_j+L_j}, L_j, d_j)$, where d_j is the terminal flag; bootstrapping uses γ^{L_j} to account for the actual duration. We describe the simulation objective here. Offline and online real-robot adaptation use the setting-specific objectives in Appendix A.1.

Task reward and episode compute. For sparse-success tasks, the final query of a successful episode receives reward one. An episode compute cost P_{ep} is distributed uniformly over its N_{ep} queries:

$$R_j = b_{\text{ep}} + \mathbf{1}[j = N_{\text{ep}}]\mathbf{1}[\text{success}], \qquad b_{\text{ep}} = -\lambda_{\text{cost}} \frac{P_{\text{ep}}}{N_{\text{ep}}}.$$

We construct P_{ep} from a target band for $\bar{\rho}_{\text{ep}}$ in Equation (1). Exceeding the upper band incurs an over-budget cost with a small debt tolerance. Successful episodes can incur an under-use cost below the lower band and receive a small saving bonus inside the band. Positive P_{ep} denotes a cost and negative P_{ep} denotes a bonus. We use the redistributed cost as a query-level training regularizer.

For a sampled next joint action $\tilde{u}'$, the critic target and actor objective are

$$y = R + \gamma^L (1 - d) \bar{Q}_{\text{targ}}(o', \tilde{u}'), \qquad \bar{Q}_{\text{targ}} = \frac{1}{M} \sum_{i=1}^{M} Q_{\bar{\psi}}^{(i)},$$

$$\mathcal{L}_{\text{sac}} = \mathbb{E}_{o, u \sim \pi_\phi} \left[\alpha \log \pi_\phi(u \mid o) - \bar{Q}(o, \tilde{u}) \right], \qquad \bar{Q} = \frac{1}{M} \sum_{i=1}^{M} Q_{\psi}^{(i)}.$$

The critics minimize squared Bellman error, and entropy regularizes the actor (Haarnoja et al., 2018); the critic target is entropy-free.

Where to spend the budget. The episode band regularizes total compute. A local alignment loss encourages the normalized allocation $a_{\text{alloc}} = \bar{K}^{01} - \bar{C}^{01}$ to follow signed difficulty $s(o)$, favoring more sampling and earlier feedback at high-difficulty states. Task-value learning trains latent steering and both schedule residuals jointly; alignment regularizes allocation, and entropy encourages exploration. Appendix A.1 gives the full actor loss and gate definitions.

4 Experiments

4.1 Experimental Setup

Settings and baselines. We evaluate on LIBERO and ALOHA simulation tasks and on real-robot manipulation (Liu et al., 2023; Zhao et al., 2023). All methods use π_0 (Black et al., 2024) as their base action model. EQRL and DSRL (Wagenmaker et al., 2025) keep the VLA frozen and train a latent-space adaptor; EQRL additionally selects denoising steps K and execution chunk length C. We also report the frozen base policy and task-specific behavior cloning (BC/SFT). In simulation, RLinf-Noise and RLinf-SDE are the RLinf implementations (Yu et al., 2025a) of Flow-GRPO (Liu et al., 2026a) and ReinFlow (Zhang et al., 2026), respectively. Real-robot comparisons use the deployable frozen, SFT, DSRL, and EQRL policies; the evaluated RLinf-Noise/SDE implementations do not support our π_0/AirBot setup.

Evaluation and metrics. Unless otherwise stated, simulation experiments use five random seeds. Tables report aggregates across seeds; figure captions specify curve selection and temporal smoothing. Task-wise DSRL/EQRL evaluations use 10 episodes per seed and checkpoint on Cream Cheese to Tray and ALOHA Cube Transfer, and 36 on Moka Pots on the Stove. We report final success and success AUC, the arithmetic mean of checkpoint success over training, computed within each seed before averaging across seeds. Simulation uncertainty is reported as sample SD across seeds unless a confidence interval is explicitly identified. Training budgets are stated in SAC optimizer updates. Amortized NFE denotes denoising work per requested action, $\sum_j K_j / \sum_j C_j$; it is distinct from per-query denoising steps K and from the executed-action rate $\sum_j K_j / \sum_j L_j$ when a chunk terminates early. Details of the adaptor, setting-specific objectives, and real-robot protocols are in Appendix A.1.

4.2 Main Simulation Results

We first evaluate four LIBERO-90 tasks and ALOHA Cube Transfer, then examine long-horizon task completion on LIBERO-10 Moka Pots on the Stove. Figure 3 shows all six learning curves; Appendix A.2 gives task-wise uncertainty statistics.

Multi-task performance and efficiency. The four LIBERO-90 tasks are Black Bowl on the Cabinet, Alphabet Soup to Tray, Cream Cheese to Tray, and Tomato Sauce to Tray. EQRL achieves the highest aggregate success and AUC and the lowest requested NFE among the compared methods (Table 1). Relative to DSRL, success increases from 0.95 to 0.96, full-training AUC from 0.86 to 0.90, and requested NFE falls from 0.500 to 0.303, a 39.4% reduction. On bimanual ALOHA Cube Transfer, EQRL improves success from 0.940 to 0.960 and AUC from 0.827 to 0.911, while reducing NFE from 0.200 to 0.172 (14.0%). It also outperforms RLinf-Noise and RLinf-SDE on all three metrics. These results show that joint control of sampling and feedback improves the performance–compute tradeoff across tasks and simulation settings.

Table 1: Five-seed simulation results. The LIBERO-90 row group aggregates four tasks.

Environment	Method	Success ↑	Full AUC ↑	NFE/action ↓
LIBERO-90 (four tasks)	DSRL	0.95	0.86	0.500
	RLinf-Noise	0.93	0.68	0.400
	RLinf-SDE	0.83	0.69	0.400
	EQRL	**0.96**	**0.90**	**0.303**
ALOHA Cube Transfer	DSRL	0.940	0.827	0.200
	RLinf-Noise	0.91	0.71	0.200
	RLinf-SDE	0.85	0.73	0.200
	EQRL	**0.960**	**0.911**	**0.172**

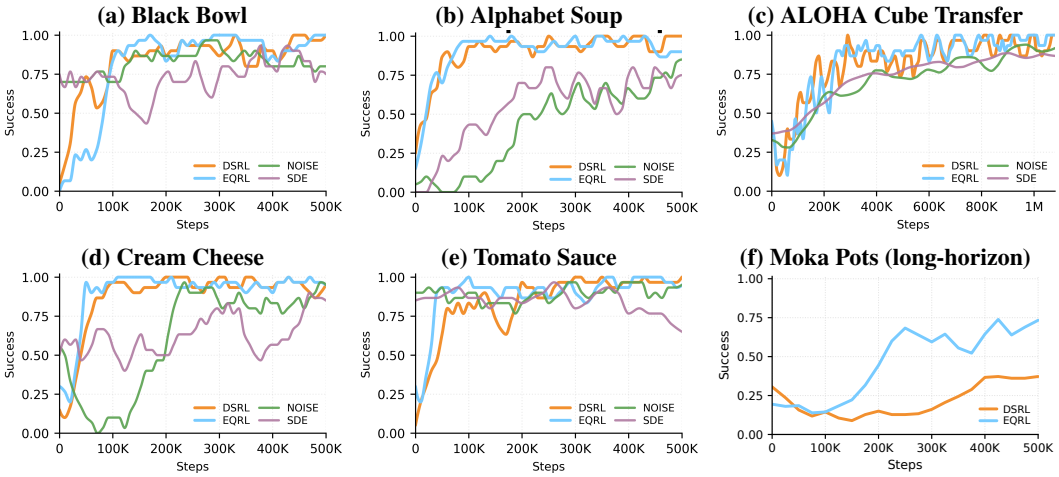


Figure 3: Learning curves across six simulation tasks. The four LIBERO-90 tasks occupy the left two columns; ALOHA and long-horizon Moka occupy the right column. Panels (a)–(e) show each method's highest full-training-AUC run among five seeds. Panel (f) shows seed 1 with a trailing mean over five evaluations. Aggregate comparisons are in Tables 1 and 2.

Long-horizon task completion. Moka Pots on the Stove requires placing both pots on the stove. Its expert demonstrations have the longest median trajectory in LIBERO-10: 401 actions. Across five seeds, EQRL raises success from 0.4111 to 0.8667, a gain of 45.56 percentage points, and full-training AUC from 0.2758 to 0.5333 (Table 2). Requested NFE decreases from 0.500 to 0.441 (11.8%). Elastic execution thus delivers a substantial gain in completing the extended two-placement sequence while reducing inference work.

Figure 4 shows how EQRL distributes computation within these longer episodes. At 500k updates (seed 1), the middle third has the highest critic disagreement and receives 0.518 requested NFE, versus 0.428 and 0.424 in the early and late thirds. This 21–22% increase comes from both more denoising steps and shorter action chunks. EQRL concentrates computation and feedback in the demanding portion of the sequence and saves computation elsewhere, connecting the long-horizon gains to state-dependent execution.

Table 2: Long-horizon Moka Pots results over five seeds. Success: mean ± sample SD; full AUC and requested NFE: seed means.

Method	Success ↑	Full AUC ↑	NFE ↓
DSRL	0.4111 ± 0.0865	0.2758	0.500
EQRL	$\mathbf{0.8667 \pm 0.1381}$	**0.5333**	**0.441**

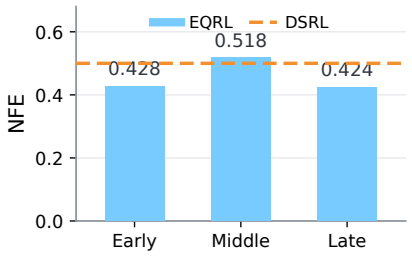


Figure 4: Within-episode NFE on Moka Pots at 500k updates (seed 1). Dashed line: fixed DSRL.

4.3 Ablations and Scheduling Mechanisms

Joint control of sampling and feedback. Figure 5 separates matched-NFE comparison from scheduling ablations on Cream Cheese to Tray. At the same requested NFE of 0.32, EQRL reaches high success earlier than DSRL (panel a). The K-only and C-only variants reach high success at 0.40 and 0.41 NFE; EQRL uses 20.0% and 22.0% less computation, respectively (panel b). Budget-only uses 0.35 NFE and exhibits larger drops in success. Jointly choosing sampling effort and feedback timing thus improves learning efficiency at lower inference cost.

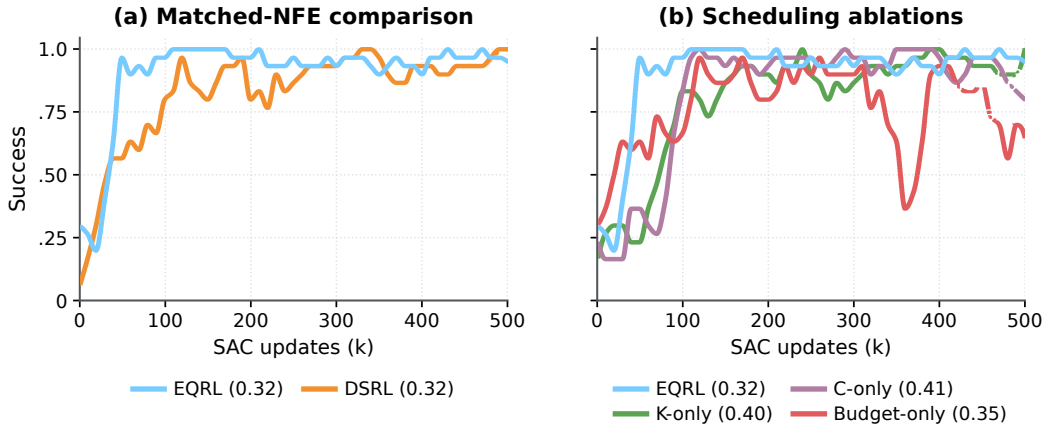


Figure 5: Matched-NFE comparison and scheduling ablations on Cream Cheese to Tray. DSRL in panel (a) is matched to EQRL at 0.32 NFE; the standard DSRL baseline for this task uses 0.50 NFE. Each method shows its highest full-training-AUC run among five seeds; both panels share the same EQRL run. Parentheses give requested NFE.

Learned residuals and difficulty guidance. Table 3 compares three variants: Prior-only disables learned schedule residuals, SAC-only disables the difficulty prior, and full EQRL uses both. All retain latent-policy learning and critic updates. Each uses 500k SAC updates, five seeds, and 10 final evaluation episodes per seed.

Table 3: Prior/residual ablations on Cream Cheese to Tray (five seeds; mean ± sample SD).

Variant	Success ↑	Full AUC ↑	Requested NFE ↓
Prior-only	0.960 ± 0.055	0.860 ± 0.016	0.344 ± 0.011
SAC-only	0.860 ± 0.055	0.834 ± 0.023	0.400 ± 0.016
Full EQRL	$\mathbf{1.000 \pm 0.000}$	$\mathbf{0.886 \pm 0.015}$	$\mathbf{0.320 \pm 0.007}$

Adding learned schedule residuals to Prior-only raises final success from 0.960 to 1.000, improves full-training AUC by 0.026, and reduces requested NFE by 7.0%. Relative to SAC-only, full EQRL improves success by 14 percentage points and AUC by 0.052, while reducing NFE by 20%. Full EQRL has the strongest mean outcome on all three metrics, supporting complementary roles for learned schedule adjustments and the difficulty prior.

Difficulty-guided allocation. Critic disagreement correlates positively with independent difficulty measures on Cream Cheese to Tray, ALOHA Cube Transfer, and Moka Pots on the Stove, with Spearman correlations of 0.333, 0.368, and 0.278, respectively. State interventions on three mature Cream Cheese runs link this signal to scheduling gains: shortening C improves highest-difficulty-quartile success by 7.3–30.0 percentage points, and increasing K gives a pooled gain of 4.4 points (95% CI $[0.045, 11.5]$). Earlier feedback delivers the more consistent state-dependent benefit, while additional denoising improves success in the highest-difficulty states. Appendix A.3 reports the difficulty proxies, quartile rates, and intervention statistics.

4.4 Real-Robot Results

For offline adaptation, each task uses 50–100 successful expert trajectories collected on DISCOVER Robotics PTK2 platforms and 500k adaptor updates. For online adaptation on a DISCOVER Robotics TOK4 platform, we initialize from 20 expert trajectories and collect approximately 15k robot actions while performing 500k SAC updates. The offline objective uses fixed-stride demonstration transitions with schedule supervision and a fixed-data compute surrogate; online robot interaction uses a causal query-local compute cost (Appendix A.1).

Baselines and success protocol. Each method is evaluated in 20 physical trials per task. Success requires completing every task stage consecutively; any failed stage fails the trial. All methods share the same reset procedure and object-position randomization within the same small predefined range, evaluating in-domain generalization across object placements. Per-task π_0-SFT uses 20k gradient updates, has no RL adaptor, and uses DSRL's fixed VLA execution schedule. The frozen zero-shot policy obtains $0/20$ on every task, while per-task SFT obtains 4–$9/20$ (Table 4). EQRL exceeds DSRL's success count on four tasks and matches it on Fruit-to-Bowl, while reducing inference cost and execution time on all five tasks.

Table 4: Real-robot end-to-end success. Every cell reports successful trials out of 20. The first four tasks use offline adaptation; the last task uses online adaptation.

Task	Frozen π_0	π_0-SFT	DSRL	EQRL
Bimanual Coke Pouring	0/20	4/20	11/20	**15/20**
Fruit-to-Bowl	0/20	9/20	**17/20**	**17/20**
Cucumber Handover-to-Bowl	0/20	8/20	18/20	**19/20**
Lemon-to-Bowl	0/20	7/20	17/20	**18/20**
Cup-to-Bowl and Bowl Lift (online)	0/20	8/20	18/20	**19/20**

Table 5: Real-robot efficiency. Time is mean trajectory execution time in seconds; the offline average uses the first four tasks. Online timing includes all 20 attempts per method, including failures; its sample SDs are in Appendix A.1.

	NFE/action ↓		Time (s) ↓	
Task	DSRL	EQRL	DSRL	EQRL
Bimanual Coke Pouring	0.500	**0.332**	8.94	**7.36**
Fruit-to-Bowl	0.500	**0.345**	6.91	**5.31**
Cucumber Handover-to-Bowl	0.500	**0.334**	5.98	**5.29**
Lemon-to-Bowl	0.500	**0.341**	6.76	**6.02**
Offline average	0.500	**0.338**	7.15	**6.00**
Cup-to-Bowl and Bowl Lift (online)	0.500	**0.341**	13.26	**11.31**

Compute and physical execution. Across the four offline tasks, EQRL raises average success from 78.75% $(63/80)$ to 86.25% $(69/80)$, a gain of 7.5 percentage points. At the same time, mean requested NFE falls from 0.500 to 0.338 and mean trajectory time from 7.15 to 6.00 seconds (Table 5). On the online Cup-to-Bowl and Bowl Lift task, EQRL completes $19/20$ trials versus $18/20$ for DSRL, with 0.341 versus 0.500 NFE. Mean trajectory time falls from 13.26 to 11.31 seconds, a 14.7% reduction computed from the unrounded means over all 20 attempts per method, including failures. The online comparison thus combines higher completion, 31.8% lower requested NFE, and shorter execution time. Figure 6 connects the aggregate saving to the actual sequence: requested inference work per action increases around alignment and grasping and decreases during steadier transfer.

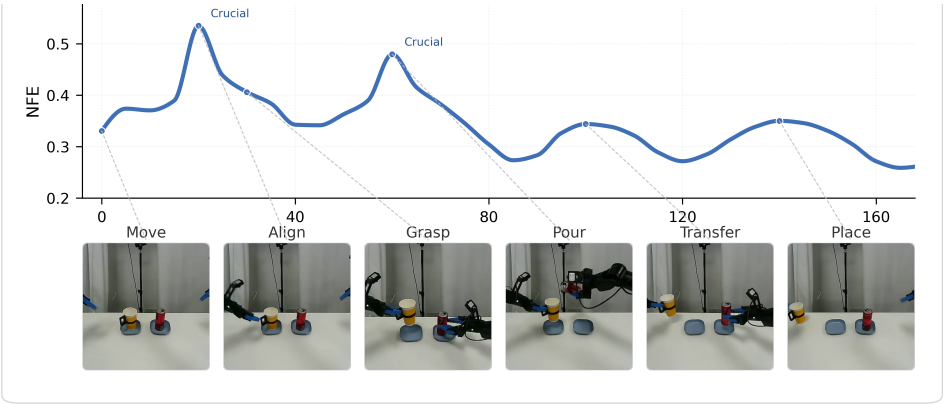


Figure 6: Bimanual Coke Pouring: requested NFE over an EQRL trajectory. Requested inference work per action peaks near alignment and grasping, then falls during steadier transfer and placement.

5 Conclusion

EQRL learns latent steering, denoising effort, and feedback timing for a frozen VLA. The four-task LIBERO-90 and ALOHA comparisons establish higher aggregate success and learning AUC at lower inference cost, and the long-horizon Moka Pots task shows a large improvement in completing a multi-stage sequence. On all five real-robot tasks, EQRL matches or exceeds DSRL's success while reducing computation and execution time. Scheduling ablations, prior/residual comparisons, and state interventions support joint schedule learning with difficulty guidance and separate learned residuals.

AI Use Statement

We used generative AI tools to assist with manuscript writing, language polishing, and grammar correction. The authors reviewed and edited all AI-assisted text and take full responsibility for the final content of this paper.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*, 2024.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

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.

Mateo Clemente, Leo Brunswic, Yang Yang, Xuan Zhao, Yasser Khalil, Haoyu Lei, Amir Rasouli, and Yinchuan Li. Two-steps diffusion policy for robotic manipulation via genetic denoising. *Advances in Neural Information Processing Systems*, 38:139962–139984, 2026.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Yanjiang Guo, Jianke Zhang, Xiaoyu Chen, Xiang Ji, Yen-Jen Wang, Yucheng Hu, and Jianyu Chen. Improving vision-language-action model with online reinforcement learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15665–15672. IEEE, 2025.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Ernst Hairer, Syvert P. Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer, 2 edition, 1993. doi: 10.1007/978-3-540-78862-1.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, et al. Simplevla-rl: Scaling vla training via reinforcement learning. *arXiv preprint arXiv:2509.09674*, 2025.

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

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *Advances in neural information processing systems*, 38:40783–40818, 2026a.

Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What can rl bring to vla generalization? an empirical study. *Advances in Neural Information Processing Systems*, 38:97121–97151, 2026b.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

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

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. *arXiv preprint arXiv:2405.07503*, 2024.

Kohei Sendai, Maxime Alvarez, Tatsuya Matsushima, Yutaka Matsuo, and Yusuke Iwasawa. Leave no observation behind: Real-time correction for vla action chunks. *arXiv preprint arXiv:2509.23224*, 2025.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 32211–32252, 2023.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999. doi: 10.1016/S0004-3702(99)00052-1.

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

Jiaming Tang, Yufei Sun, Yilong Zhao, Shang Yang, Yujun Lin, Zhuoyang Zhang, James Hou, Yao Lu, Zhijian Liu, and Song Han. Vlash: Real-time vlas via future-state-aware asynchronous inference. *arXiv preprint arXiv:2512.01031*, 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

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.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Yuqi Xie, Fengyuan Hu, Jimmy Wu, Zhengyi Luo, Linxi Fan, et al. Self-improving vision-language-action models with data generation via residual rl. *arXiv preprint arXiv:2511.00091*, 2025.

Sherry Yang, Ofir Nachum, Yilun Du, Jason Wei, Pieter Abbeel, and Dale Schuurmans. Foundation models for decision making: Problems, methods, and opportunities. *arXiv preprint arXiv:2303.04129*, 2023.

Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, et al. Rlinf: Flexible and efficient large-scale reinforcement learning via macro-to-micro flow transformation. *arXiv preprint arXiv:2509.15965*, 2025a.

Shu-Ang Yu, Feng Gao, Yi Wu, Chao Yu, and Yu Wang. D3p: Dynamic denoising diffusion policy via reinforcement learning. *arXiv preprint arXiv:2508.06804*, 2025b.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. *Advances in Neural Information Processing Systems*, 38: 106282–106319, 2026.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

A Appendix

A.1 Experimental Details

EQRL and DSRL share the same frozen π_0 base policy (Black et al., 2024) and a Pixel-SAC agent that acts in the noise latent space. Our method (EQRL) additionally learns an elastic schedule, i.e., the injected noise w together with the number of denoising steps and the action-chunk length, under a compute (NFE) budget and a q_{std}-based difficulty signal. The DSRL baseline (Wagenmaker et al., 2025) uses the same SAC backbone but keeps the denoising steps and action chunk fixed and removes all elastic modules (range, NFE budget, and difficulty). We report the hyperparameters for the real-robot experiments (Table 6), the LIBERO simulation tasks (Table 7), and the ALOHA simulation task (Table 8).

Hyperparameters whose meaning is shared across environments use the same row name in all tables. "SAC updates" counts optimizer updates in all three settings (capped by `max_steps`), not environment interactions. The online real-robot run uses 5×10^5 SAC updates and approximately 15k executed robot actions. In simulation, "Grad updates / query transition" counts gradient updates per newly collected query-level transition (`multi_grad_step`), subject to the total update budget. For online real-robot training, "Grad steps / collection" counts updates per collection round; offline training has no collection round. Hyperparameter tables report the parameters used in each setting; the accompanying text specifies the fixed-schedule DSRL configuration.

Adaptor architecture and query procedure. Separate four-layer, 32-channel CNNs with 50-dimensional bottlenecks feed three-layer, 128-unit actor and critic MLPs. Three tanh-Gaussian heads produce 32-dimensional latent steering w and normalized schedule outputs z_K, z_C; ten critic heads evaluate the joint latent-schedule action. Each query proposes latent steering and schedule residuals, computes causally normalized critic disagreement at the base schedule, combines the difficulty prior with gated residuals, clips and rounds (K, C), and executes $L \le C$ actions from the frozen VLA. The adaptor then updates using the transition and compute objective appropriate to the setting. After clipping to the active bounds, critic inputs are normalized using the fixed global bounds:

$$\bar{K}_j = 2\frac{\hat{K}_j - K_{\min}}{K_{\max} - K_{\min}} - 1, \qquad \bar{C}_j = 2\frac{\hat{C}_j - C_{\min}}{C_{\max} - C_{\min}} - 1.$$

The critic evaluates these continuous coordinates; environment interaction uses the bounded, rounded integer schedule.

Schedule parameterization. The range curriculum has progress $\alpha_j \in [0, 1]$ and sets $g_j = g_{\min} + (1 - g_{\min})\alpha_j$. Writing $[x]_+ = \max(x, 0)$, for either coordinate $X \in \{K, C\}$ the curriculum sets the active bounds, and the actor output maps around X_0:

$$X_{\min,j} = X_0 + \alpha_j(X_{\min} - X_0), \qquad X_{\max,j} = X_0 + \alpha_j(X_{\max} - X_0),$$
$$X_j^{\text{learned}} = X_0 + [z_j^X]_+(X_{\max,j} - X_0) - [-z_j^X]_+(X_0 - X_{\min,j}).$$

Thus $z_j^X = -1, 0, 1$ gives the active lower bound, base value, and active upper bound, respectively.

Reference schedule and gates. All reported runs use `rank_tanh`. During rollout, causal EMA estimates of the mean and second moment of q_{std} give its standardized score z; updates use batch statistics. We set $d = \text{sigmoid}(z)$, clipping it to $[10^{-5}, 1 - 10^{-5}]$ during rollout. During updates, $r_c = 2\,\text{rank}(d) - 1$ is the centered percentile rank; for a single rollout observation, we use $r_c = \tanh(\sqrt{2/\pi}\,\text{logit}(d))$. The signed difficulty is $s(o) = \tanh(\tau r_c(o)) \in [-1, 1]$, where τ controls the signal scale.

A prior warmup and success gate give prior authority $h_j = g_j e_j\{f + (1-f)\,\text{clip}[(n_j - n_0)/T, 0, 1]\}$. Here n_j counts optimizer updates, n_0 and T are the warmup start and duration, $e_j \in [0, 1]$ gates the prior using recent rollout success, and f is a prior floor. With prior strength β and deadband δ, the signed prior input is

$$p_j = \beta h_j\ \text{sgn}(s_j)\frac{(|s_j| - \delta)_+}{1 - \delta}.$$

The reference schedule uses the active curriculum bounds:

$$K_j^p = K_0 + [p_j]_+(K_{\max,j} - K_0) - [-p_j]_+(K_0 - K_{\min,j}),$$
$$C_j^p = C_0 - [p_j]_+(C_0 - C_{\min,j}) + [-p_j]_+(C_{\max,j} - C_0).$$

The residual scale moves from its pre-prior value λ_{pre} toward λ_{post} as prior authority rises:

$$\lambda_{r,j} = \lambda_{\text{pre}} - (3h_j^2 - 2h_j^3)(\lambda_{\text{pre}} - \lambda_{\text{post}}), \qquad \Delta K_j = K_j^{\text{learned}} - K_0, \quad \Delta C_j = C_j^{\text{learned}} - C_0.$$

The resulting continuous targets are clipped to the active schedule range before rounding.

Alignment loss. For the local training signal, let $a_j = \bar{K}_j^{01} - \bar{C}_j^{01}$, where both coordinates are normalized to $[0, 1]$ within the current active bounds, and let a_j^{mode} use the deterministic actor mode. With signed difficulty s_j, confidence $|s_j|$, margin m, and mode weight λ_{mode}, the alignment loss is

$$\mathcal{L}_{\text{align}} = \mathbb{E}_j\left[|s_j|\big(m - s_j a_j\big)_+^2\right] + \lambda_{\text{mode}}\mathbb{E}_j\left[|s_j|\big(m - s_j a_j^{\text{mode}}\big)_+^2\right].$$

The optimized actor objective is $\mathcal{L}_\pi = \mathcal{L}_{\text{sac}} + \lambda_d h_j^{\text{align}}\mathcal{L}_{\text{align}} - \lambda_H \mathcal{H}_{K,C}$, where λ_d and λ_H weight alignment and schedule-head entropy, h_j^{align} is the training-progress/difficulty gate, and $\mathcal{H}_{K,C}$ is the entropy of the normalized outputs z_K, z_C. The prior and alignment use critic-derived difficulty with separately configured signal scales; the value objective jointly trains latent steering and the two residuals.

Evaluation and statistical aggregation. The simulation logger's `success_auc` is the arithmetic mean of evaluation success values over training; `success_ema` is a trailing mean over the last five evaluations. The latter is a smoothed curve statistic, not an additional set of independent trials. We distinguish full-training AUC from early- and late-training summaries throughout the experimental tables. For seed s with M_s evaluation checkpoints, we first compute $\text{AUC}_s = M_s^{-1}\sum_{m=1}^{M_s}\hat{p}_{s,m}$, where $\hat{p}_{s,m}$ is checkpoint success, then average AUC_s across seeds. Sample SD uses denominator $S - 1$ for S seeds; 95% t-intervals use the seed summaries as statistical units and are not clipped to $[0, 1]$. For the prior/residual ablation in Table 3, each metric is summarized directly from five seeds, with final success measured over 10 episodes per seed.

The four-task LIBERO, ALOHA, and scheduling-ablation figures select the run with the largest full-training AUC separately for each method and task. The Moka Pots learning curve shows seed 1 with a trailing mean over five evaluations; tables use cross-seed aggregates. Figure 4 reports requested NFE within the early, middle, and late temporal thirds of episodes for the 500k-update seed-1 policy, separately from training-time AUC. Appendix A.2 collects task-wise statistics and their sources.

Offline real-robot objective. Each task uses 50–100 successful expert trajectories and 500k adaptor updates. We construct fixed-stride transitions $(o_t, \tilde{u}_t, r_t, o_{t+s})$ (default $s = 1$), with discount γ^s, zero intermediate reward, and terminal reward one. The recorded trajectory and stride determine the next observation and duration, independently of the candidate chunk length C. At sampled states, we hold the observation and injected noise fixed, evaluate low/base/high (K, C) schedules with the frozen VLA, and compare generated prefixes with future expert actions to form schedule pseudo-labels. The reported configuration places these labels in the replay action's (K, C) coordinates and applies an auxiliary regression loss to the two schedule heads. The NFE target range and weights regularize predicted K/C through a differentiable fixed-data surrogate; there is no episode reward redistribution from rollouts under candidate schedules. Expert comparison supplies schedule supervision, not counterfactual environment reward. The reported results use `demo_pad` for w: recorded low-level expert actions occupy the first latent coordinates and the remainder is zero-padded. This is a replay-construction proxy, not an inverse latent label.

Online real-robot objective. Online robot interaction uses query transitions with discount γ^{L_j}, but its compute penalty is causal and query-local. When the efficiency gate is enabled, the implementation uses $R_j = r_j^{\text{task}} - \lambda_{\text{step}}L_j - \lambda_{\text{NFE}}K_j$. It does not redistribute the simulation episode-ratio cost across queries.

Real-robot task names. The offline tasks in Tables 4 and 5 are *Bimanual Coke Pouring*, *Fruit-to-Bowl*, *Cucumber Handover-to-Bowl*, and *Lemon-to-Bowl*. The pouring task requires grasping the cup and bottle and pouring into the cup; the cucumber task includes a handover before placement in the bowl. In Bimanual Coke Pouring, the left arm holds the paper cup and the right arm grasps and tilts the Coke bottle above it. Fruit-to-Bowl requires grasp, transfer, and release of the specified fruit into the target bowl; Lemon-to-Bowl requires the lemon to end inside the bowl after grasp and placement. In Cucumber Handover-to-Bowl, one arm grasps the cucumber, transfers it to the other arm, and the receiving arm places it in the bowl. The online task is *Cup-to-Bowl and Bowl Lift*: grasp the cup, place it in the bowl, then lift the bowl. Each method–task pair uses 20 physical trials, and success requires completing every stage. Object positions vary within a predefined range in the same task setting to evaluate in-domain generalization; all methods share the reset protocol and placement distribution.

Online real-robot timing. For Cup-to-Bowl and Bowl Lift, trajectory execution time is measured over all 20 physical attempts per method, including failed attempts. The mean $\pm$ sample SD is 13.255 ± 1.007 seconds for DSRL and 11.310 ± 0.773 seconds for EQRL. Here the statistical unit is a physical attempt, and sample SD uses denominator $20 - 1$. The same attempts give $18/20$ and $19/20$ task successes, respectively. Table 5 rounds the means to two decimals; the 14.7% time reduction is calculated as $1 - 11.310/13.255$. NFE reduction uses the requested-action rate, $1 - 0.341/0.500 = 31.8\%$.

Table 6: Hyperparameters for the real-robot EQRL experiments. The offline surrogate uses NFE band $[0.28, 0.50 \to 0.45]$ and under-use/saving weights $0.35/0.08$. Online training uses the query-local objective and 1000 gradient updates per collection round.

Hyperparameter	Offline	Online
Task	Fruit-to-Bowl	Cup-to-Bowl and Bowl Lift
Base policy	π_0, frozen	π_0, frozen
Train mode	Offline	Online
SAC updates	5×10^5	5×10^5
Batch / discount γ	256 / 0.999	256 / 0.999
LR (actor / critic / temp)	10^{-4} / $3{\times}10^{-4}$ / $3{\times}10^{-4}$	10^{-4} / $3{\times}10^{-4}$ / $3{\times}10^{-4}$
Critic ensemble `num_qs`	10	10
Action magnitude	2.0	2.0
Target entropy	0.0	0.0
Base (denoising steps, chunk)	(10, 20)	(10, 25)
Denoising-step range	[5, 15]	[5, 15]
Action-chunk range	[10, 30]	[5, 50]
Difficulty source / weight	`q_std` / 0.045	`q_std` / 0.08
Difficulty prior scale (signal)	0.65 (`rank_tanh`)	0.85 (`rank_tanh`)

Table 7: EQRL hyperparameters for LIBERO-90. Shared settings appear above the task-specific settings. †Black Bowl values are reconstructed from the run name (its config dict was not saved). DSRL shares the base, optimizer, batch size, discount, critic ensemble, action magnitude, update budget and ratio; it uses fixed $(K, C) = (10, 20)$ with automatic entropy tuning and latent steering.

Hyperparameter	Shared by all four tasks
Base policy	π_0 (LIBERO), frozen
Train mode	Online
SAC updates	5×10^5
Batch / discount γ	256 / 0.999
LR (actor / critic / temp)	10^{-4} / $3{\times}10^{-4}$ / $3{\times}10^{-4}$
Critic ensemble `num_qs`	10
Action magnitude	1.0
Target entropy	auto (-17)
Base (denoising steps, chunk)	(10, 20)
Difficulty source (signal)	`q_std` (`rank_tanh`)
Grad updates / query transition	20

	Alphabet Soup to Tray	Black Bowl on the Cabinet	Cream Cheese to Tray	Tomato Sauce to Tray
Denoising-step range	[4, 14]	[3, 15]	[3, 15]	[3, 15]
Action-chunk range	[16, 42]	[16, 44]	[10, 40]	[16, 44]
NFE band [lower, upper→anneal]	$[0.34, 0.50 \to 0.40]$	$[0.28, 0.50 \to 0.39]^\dagger$	$[0.28, 0.50 \to 0.425]$	$[0.28, 0.50]$
NFE under / saving weight	0.55 / 0.018	$0.35/0.09^\dagger$	0.35 / 0.06	0.35 / 0.12
Difficulty weight	0.024	0.06	0.06	0.06
Difficulty prior scale	0.82	$0.80^\dagger$	1.0	0.85

Table 8: EQRL hyperparameters for ALOHA Cube Transfer. DSRL shares the frozen base, SAC optimizer and update settings, batch size, discount, critic ensemble, action magnitude, and target entropy, and uses fixed $(K, C) = (10, 50)$ with latent steering.

Hyperparameter	`transfer_cube` (EQRL)
Base policy	π_0 (ALOHA), frozen
Train mode	Online
SAC updates	1×10^6
Batch / discount γ	256 / 0.999
LR (actor / critic / temp)	10^{-4} / $3{\times}10^{-4}$ / $3{\times}10^{-4}$
Critic ensemble `num_qs`	10
Action magnitude	2.0
Target entropy	0.0
Base (denoising steps, chunk)	(10, 50)
Denoising-step range	[3, 13]
Action-chunk range	[42, 50]
NFE band [lower, upper→anneal]	[0.12, 0.20→0.18]
NFE under / saving weight	0.40 / 0.06
Difficulty source / weight	`q_std` / 0.035
Difficulty prior scale (signal)	0.55 (`rank_tanh`)
Grad updates / query transition	20

A.2 Additional Simulation Statistics

Table 9 reports five-seed task-wise comparisons using the aggregation rules in Appendix A.1. The following paragraphs give additional uncertainty statistics and task-specific details.

Table 9: Five-seed task-wise simulation results. Success is mean ± sample SD; full-training AUC and requested NFE are means across seeds.

Task	Method	Success ↑	Full AUC ↑	NFE/action ↓
LIBERO-90				
Cream Cheese to Tray	DSRL	0.980 ± 0.045	0.934	0.500
	EQRL	1.000 ± 0.000	0.886	0.320
ALOHA Cube Transfer	DSRL	0.940 ± 0.089	0.827	0.200
	EQRL	0.960 ± 0.055	0.911	0.172
LIBERO-10				
Moka Pots on the Stove	DSRL	0.4111 ± 0.0865	0.2758	0.500
	EQRL	0.8667 ± 0.1381	0.5333	0.441

Cream Cheese to Tray. This LIBERO-90 task is also used for the scheduling and prior/residual ablations. Endpoint-success 95% t-intervals are $[0.068, 0.532]$ for the frozen base, $[0.924, 1.036]$ for DSRL, and $[1, 1]$ for EQRL; all five EQRL endpoint values are one, giving zero between-seed sample variance. DSRL's endpoint-success SD is 0.045. The frozen base has seed-level success rates $0.0, 0.4, 0.5, 0.3, 0.3$ and fixed $(K, C) = (10, 20)$, giving requested NFE 0.500 with zero SD. Its SFT data cover 40 tasks from LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-10, excluding the target Cream Cheese to Tray task. Late-training AUC is 0.984 ± 0.011 ($[0.970, 0.998]$) for DSRL and 0.980 ± 0.021 ($[0.954, 1.006]$) for EQRL. The full-EQRL means in Table 3 agree with Table 9; its full-training AUC and requested-NFE SDs are 0.015 and 0.007.

ALOHA Cube Transfer. Figure 3(c) shows learning curves; Table 10 adds five-seed statistics from 10 evaluation trials per seed and checkpoint. DSRL's endpoint-success CI is $[0.829, 1.051]$; its SD in Table 9 is reconstructed from the rounded interval as $\mathrm{CI\ halfwidth}\sqrt{5}/2.7764$. DSRL's full-training AUC is 0.827 ($[0.793, 0.861]$); EQRL's success, AUC, and NFE intervals are in Table 10. The rollout diagnostics report episode return, logged stop count, episode length, and absolute EQRL inference latency per query.

Table 10: Additional five-seed ALOHA Cube Transfer statistics. Where available, the middle column gives mean ± sample SD; brackets are unclipped 95% t-intervals. Relative NFE reduction uses DSRL as the reference.

Method	Metric	Mean [±SD]	95% CI
DSRL	Early-training AUC	0.343	$[0.220, 0.466]$
	Late-training AUC	0.957	$[0.929, 0.985]$
	Requested NFE	0.200	$[0.200, 0.200]$
EQRL	Success	0.9600 ± 0.0548	$[0.8920, 1.0280]$
	Full-training AUC	0.9105 ± 0.0185	$[0.8875, 0.9334]$
	Requested NFE	0.1720 ± 0.0030	$[0.1683, 0.1757]$
	Relative NFE reduction	0.1400 ± 0.0148	$[0.1216, 0.1585]$
EQRL	Episode return	159.42 ± 0.51	$[158.79, 160.05]$
	Logged stop count	6.40 ± 1.14	$[4.98, 7.82]$
	Episode length (actions)	183.60 ± 4.39	$[178.15, 189.05]$
	Inference per query (ms)	220.73 ± 4.11	$[215.63, 225.83]$

Moka Pots on the Stove. The Moka Pots on the Stove task requires placing both moka pots on the stove; success is determined after completing both placements. Its expert demonstrations have the longest median length in LIBERO-10: 401 actions, compared with 285.5 for the next-longest task. At 500k updates, EQRL's seed-level success summaries are 0.8611, 0.8889, 0.6389, 1.0000, and

0.9444, giving mean 0.8667, sample SD 0.1381, and 95% CI $[0.6953, 1.0381]$. Its early-, full-, and late-training AUCs are 0.4077, 0.5333, and 0.6152, respectively. For DSRL, each seed evaluates 36 episodes at 11 checkpoints from 0k through 500k SAC updates in 50k increments, including the initial checkpoint. Full-training AUC is 0.2758 ± 0.0338; final success is 0.4111 ± 0.0865 with 95% CI $[0.3037, 0.5185]$. Its SD is computed directly from the five seed-level final evaluations.

A.3 Difficulty Validation and Schedule Interventions

The component ablations establish the value of flexible schedules and difficulty guidance. We now examine the state-level mechanism: whether the difficulty signal identifies states where rescheduling helps, and how those states respond to changes in C and K.

Difficulty measured independently of the prior. Table 11 compares critic disagreement with failure on Cream Cheese to Tray, difficult-stage membership on ALOHA Cube Transfer, and outcome difficulty on Moka Pots on the Stove. All three proxies are independent of the schedule prior. For Cream Cheese to Tray, the mean correlation is 0.333 across three seeds (range 0.234–0.436), with every seed-wise 95% CI excluding zero; the ALOHA Cube Transfer and Moka Pots on the Stove intervals are $[0.086, 0.619]$ and $[0.047, 0.496]$, respectively. The gap between Q1 and Q4 gives an interpretable state-level signal: Cream Cheese to Tray failure rises from 0.0% to 31.2%, ALOHA difficult-stage membership from 28.6% to 75.0%, and Moka Pots on the Stove outcome difficulty from 16.0% to 41.2%. The signal therefore identifies phases in which different execution choices can matter.

Table 11: Critic-independent difficulty validation. Q1/Q4 are the lowest/highest Q-std quartiles; their entries are difficulty rates (%). Cream Cheese to Tray reports the three-seed mean correlation; the surrounding text gives the seed range and task-specific confidence intervals.

Task	Difficulty proxy	ρ	Q1 (%)	Q4 (%)
Cream Cheese to Tray	Failure	0.333	0.0	31.2
ALOHA Cube Transfer	Difficult stage	0.368	28.6	75.0
Moka Pots on the Stove	Outcome difficulty	0.278	16.0	41.2

Do difficult states benefit from changing the schedule? We intervene on C or K at evaluation states and measure success gain over the reference schedule. Across three mature Cream Cheese to Tray runs at 500k updates, shortening C has a positive relation between Q-std and gain ($\rho = 0.214$–0.548, with each interval excluding zero), and Q4 gains of $+7.3$–$+30.0$ percentage points. Increasing K gives a pooled Q4 gain of $+4.4$ points (95% CI $[0.045, 11.5]$). Its across-state rank correlation is 0.035 with a 95% interval of $[-0.311, 0.338]$ (Table 12). Earlier feedback therefore has the clearer state-dependent effect, while additional denoising improves success in the highest-difficulty quartile. These distinct responses support independently learned K and C corrections.

Table 12: Cream Cheese to Tray schedule interventions. Success gains are percentage points. Shorter-C entries are ranges across three runs; each correlation CI excludes zero. Larger-K entries give the three-seed pooled estimates with 95% CIs in brackets.

Intervention	$\rho(Q_{\mathrm{std}}, \Delta S)$	Q4 ΔS (pp)
Shorter C	0.214–0.548	$+7.3$–$+30.0$
Larger K	0.035 $[-0.311, 0.338]$	$+4.4$ $[0.045, 11.5]$

A.4 Local Analysis of Compute and Feedback

This analysis explains the scheduling trade-off at a fixed observation and latent input. Its conclusions concern an explicit local surrogate; the learned policy is evaluated with task outcomes, bounded integer schedules, and changing observations. We first explain the surrogate terms, then prove Proposition 1 and give their sensitivity to separate error sources.

Sampling error under local regularity. The frozen π_0 sampler numerically integrates a learned flow using forward Euler steps (Black et al., 2024). Fix (o, w) and an integration interval of unit length. Suppose the vector field is Lipschitz in its action argument with constant Λ_o on a region containing the exact and numerical trajectories, and each step from the exact trajectory has local truncation error at most $B_o h^2$. For K uniform steps, $h = 1/K$, the standard error recursion is

$$e_{n+1} \le (1 + h\Lambda_o)e_n + B_o h^2, \qquad e_0 = 0.$$

Summing the recursion gives

$$\|A^K - A^\infty\| \le \frac{B_o}{\Lambda_o K}\big((1 + \Lambda_o/K)^K - 1\big) \le \frac{B_o(e^{\Lambda_o} - 1)}{\Lambda_o K},$$

with the continuous limit B_o/K when $\Lambda_o = 0$. This is the usual first-order integration bound (Hairer et al., 1993). If a local action-loss proxy is Lipschitz in A, its numerical-error contribution is correspondingly bounded by a_o/K, after absorbing the constants into a_o. Here A^∞ is the exactly integrated frozen flow. Accuracy relative to that flow does not ensure monotone task-success improvement: the learned flow may itself be imperfect, and terminal success depends on subsequent interaction.

Feedback delay under a local drift envelope. Suppose the additional local error at the mth action after a query is bounded by $\epsilon_o + m\delta_o$ over the considered chunk range, with $\delta_o > 0$. This assumption describes a regime where uncorrected disturbances or prediction mismatch accumulate approximately linearly before a new observation. The average contribution over C actions satisfies

$$\frac{1}{C}\sum_{m=0}^{C-1}(\epsilon_o + m\delta_o) = \epsilon_o + \frac{\delta_o(C-1)}{2} \le \epsilon_o + b_o C, \qquad b_o = \delta_o/2.$$

Dropping the schedule-independent term and combining these two local contributions with a compute price $\eta K/C$ gives Equation (2). Additive separation and the drift envelope are modeling assumptions. They need not describe discontinuous contact events or error growth over arbitrarily long open-loop horizons. Neither the constants nor an error-decay law are fitted to the reported success rates.

Proof of Proposition 1. For $a, b, \eta, K, C > 0$, the product of the three terms is constant:

$$\frac{a}{K}\,(bC)\,\frac{\eta K}{C} = ab\eta.$$

The arithmetic–geometric mean inequality yields $\ell(K, C) \ge 3(ab\eta)^{1/3}$. Equality holds exactly when $a/K = bC = \eta K/C$. Solving these equalities gives $K^* = (a^2/(\eta b))^{1/3}$ and $C^* = (\eta a/b^2)^{1/3}$, which are positive and unique. Substituting $a = a_0\xi$ and $b = b_0\xi$ proves the stated common-difficulty scaling. The analytic scale ξ here is distinct from the normalized critic signal used by the adaptor.

General error exponents. The same reasoning extends to a positive continuous surrogate

$$\ell_{p,q}(K, C) = aK^{-p} + bC^q + \eta K/C, \qquad a, b, \eta, p, q > 0.$$

Let $D = pq + p + q$. Its unique minimizer is

$$K^* = \left(\frac{(pa)^{q+1}}{qb\,\eta^q}\right)^{1/D}, \qquad C^* = \left(\frac{pa\,\eta^p}{(qb)^{p+1}}\right)^{1/D}. \tag{6}$$

To see uniqueness, set $x = \log K$ and $y = \log C$. The transformed objective $ae^{-px} + be^{qy} + \eta e^{x-y}$ has Hessian

$$\begin{pmatrix} p^2 a e^{-px} & 0 \\ 0 & q^2 b e^{qy} \end{pmatrix} + \eta e^{x-y} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix},$$

which is positive definite. The stationary conditions $paK^{-p} = qbC^q = \eta K/C$ give Equation (6); strict convexity establishes the unique global minimum. This extension shows that the qualitative coupling does not require the exact exponents $p = q = 1$.

Separate error sources and learned corrections. The log sensitivities of Equation (6) are

$$\begin{pmatrix} \partial_{\log a} \log K^* & \partial_{\log b} \log K^* & \partial_{\log \eta} \log K^* \\ \partial_{\log a} \log C^* & \partial_{\log b} \log C^* & \partial_{\log \eta} \log C^* \end{pmatrix} = \frac{1}{D} \begin{pmatrix} q+1 & -1 & -q \\ 1 & -(p+1) & p \end{pmatrix}.$$

A common multiplicative increase ξ in a and b gives $K^* \propto \xi^{q/D}$ and $C^* \propto \xi^{-p/D}$, preserving the prior's direction. Increasing only the sampling-error coefficient a instead increases both coordinates: more work per query is accompanied by longer amortization. Increasing only the feedback-error coefficient b decreases both: more frequent feedback is accompanied by less work per query. A higher compute price decreases K^* and increases C^*.

These distinctions motivate learning separate schedule adjustments around a shared reference. Critic disagreement provides one empirical signal about the current task; it does not identify a, b, or their marginal gains. EQRL uses this signal to construct the reference schedule and learns both residuals from task feedback; the intervention experiments examine the two scheduling coordinates separately. The closed forms apply before integer and range constraints; boundary optima can saturate, so the power laws are not predictions of the exact executed schedule.

Amortized accounting and early termination. For any nonempty episode with positive requested chunk lengths,

$$\rho_{\text{req}} = \frac{\sum_j K_j}{\sum_j C_j} = \sum_j \frac{C_j}{\sum_i C_i} \frac{K_j}{C_j}.$$

Thus a chunk's weight is its fraction of requested actions. For example, two queries with $(K, C) = (5, 5)$ and $(5, 45)$ have total requested NFE $10/50 = 0.2$; the unweighted query-average rate is $(1 + 1/9)/2 = 5/9$. These are different statistics because the two queries cover different numbers of actions. If the actual executed lengths satisfy $0 < \sum_j L_j \le \sum_j C_j$, then

$$\rho_{\text{exec}} = \frac{\sum_j K_j}{\sum_j L_j} = \rho_{\text{req}} \frac{\sum_j C_j}{\sum_j L_j} \ge \rho_{\text{req}}.$$

They coincide when every requested action is executed. NFE entries use the requested rate unless explicitly labeled executed NFE; the query-level discount uses actual duration L_j. These identities justify the accounting units, while reward redistribution and the setting-specific training objectives are specified separately in Section 3.4 and Appendix A.1.