NOISE SPACE LEARNING FOR VLA ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Pretrained vision-language-action (VLA) models encode broad manipulation priors, yet adapting them to specific tasks requires balancing limited robot interaction, computational resources, and safety constraints. Noise space learning (NSL) adapts flow-matching-based VLAs by learning a context-conditioned distribution over noise while keeping the pretrained VLA fixed. Building on prior noise-steering methods, we formulate NSL as an interface for VLA adaptation that accommodates different learning algorithms and instantiate it with PPO, SAC, and AWR. Theoretically, we establish conditions under which NSL attains the minimum information cost of a target behavioral change. Empirically, we study these NSL algorithms on LIBERO, RoboTwin, and four real-robot manipulation tasks under interaction, computation, and execution constraints. Across these settings, we develop a suite of effective NSL variants and derive practical recommendations for choosing among them under different resource and deployment regimes.

1 INTRODUCTION

Vision-language-action (VLA) models increasingly combine pretrained visual-language representations with continuous action generation. Early VLAs such as RT-2 (Brohan et al., 2023) predict discretized actions as tokens. Octo (Octo Model Team et al., 2024) subsequently combines a generalist robot transformer with a diffusion action head to model continuous, multimodal behavior. More recent VLAs, including π_0 (Black et al., 2024) and $\pi_{0.5}$ (Physical Intelligence et al., 2025), use flow matching to transform standard Gaussian noise into action chunks, with the flow generator conditioned on visual observations and language. This progression has made flow matching an increasingly prominent approach to continuous action generation. The broad manipulation priors of flow-matching-based VLAs provide a starting point for deployment, but precise alignment, contact handling, and dynamic manipulation still require adaptation to the target task.

The central challenge in VLA adaptation is to acquire task-specific behavior efficiently while preserving useful pretrained priors. Full-parameter updates to a large VLA are costly and can disturb these priors when training is limited to narrow task experience. For example, low-data supervised fine-tuning (SFT) can leave a policy fixated on training objects or spatial targets, even when a new instruction specifies a different target (Huang et al., 2026). A parameter-efficient alternative is low-rank adaptation (LoRA), which keeps the VLA backbone fixed and trains low-rank adapters. This approach can better preserve VLM-derived visual-semantic recognition than full-parameter fine-tuning (Zhang et al., 2025a), and retain earlier skills with substantially less forgetting in continual VLA adaptation (Hu et al., 2026). These observations motivate efficient adaptation that freezes the VLA backbone and learns a small task-specific module.

Keeping the VLA backbone fixed can help preserve pretrained priors and improve adaptation efficiency, but does not by itself ensure effectiveness. Different ways of adapting a fixed backbone face different limitations. Adapting only the action head can underfit task execution despite better preservation of visual-semantic knowledge (Zhang et al., 2025a). Inference-time candidate selection can improve task success, but seeking better actions through a broader search requires generating and scoring more candidates, increasing inference cost (Yang et al., 2025). Residual adaptation learns action corrections, but keeping these corrections close to the base policy for stable learning can limit the behavioral changes it can achieve (Ankile et al., 2025). These trade-offs motivate a central question: how can we adapt a VLA effectively and efficiently while preserving its priors?

To address this question, we study *noise space learning* (NSL) for flow-matching-based VLAs. NSL trains a lightweight *noise actor* to produce a context-conditioned distribution over the noise

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.

that initializes action generation while keeping the VLA backbone fixed. This formulation builds on learning a conditional noise actor in DSRL (Wagenmaker et al., 2025) and noise supervision obtained by inverting corrective actions in UniSteer (Lu et al., 2026) and FlowDAgger (Murray et al., 2026). NSL targets effective adaptation by favoring useful behaviors already represented by the generator, supports efficient learning through lightweight actor updates, and preserves the pretrained noise-to-action mapping. The practical challenge is how to optimize the noise actor to realize these benefits. We develop NSL variants based on three reinforcement learning paradigms: on-policy policy optimization with PPO (Schulman et al., 2017), off-policy actor–critic learning with SAC (Haarnoja et al., 2018), and advantage-weighted regression (AWR) (Peng et al., 2019) that uses supervised learning as subroutines. We study how these variants are impacted by different implementation techniques such as noise limit, task balancing, and success/failure balancing. Based on these results, we conclude which algorithms are effective and which implementation choices yield strong performance, aiming to establish best practices for NSL under different resource and deployment constraints.

We evaluate NSL on two simulation benchmarks, LIBERO (Liu et al., 2023) and RoboTwin (Chen et al., 2025b), and four real-robot manipulation tasks, cup stacking, toy fishing, sequential insertion, and bead placement, under the following constraints: **The interaction constraint** refers to the sample budget for adaptation that differ in simulation and real-robot experiments, which influences algorithm choice. **The computation constraint** refers to the computational effort allowed in adaptation and may limit the performance. **The execution constraint** concerns the safety of learned behavior on physical robots. We observe that noise distributions far from standard Gaussian can be tolerated in simulation but are associated with unsafe motions and safety interventions, such as emergency stops, on real robots.

Our contributions are threefold. First, we formulate NSL as a VLA adaptation interface that maximizes success rate through a context-conditioned noise actor that steers action generation while keeping the VLA backbone fixed. Second, our information-theoretic analysis establishes conditions under which NSL attains the minimum information cost of a target behavioral change. Third, we develop NSL variants based on PPO, SAC, and AWR through implementation refinements, and evaluate them on simulation and real robots. Our experiments show that NSL-SAC performs well in simulation with ample interaction budgets and requires more computation. However, its noise distribution can deviate from standard Gaussian, with associated unsafe motions and emergency-stop interventions on real robots. In contrast, NSL-AWR is better suited to interaction-limited real-robot adaptation, combining lower computational cost with safer execution.

2 Related Work

Adapting pretrained VLAs. Supervised and parameter-efficient methods update the effective action generator through either full-model fine-tuning or parameter-efficient updates to a limited set of weights (Kim et al., 2024; 2025; Hu et al., 2021; Jiang et al., 2026). Reinforcement-learning methods instead optimize generative policies using denoising or flow-transition likelihoods, critic-derived targets, or policy gradients (Ren et al., 2024; Zhang et al., 2025b; Chen et al., 2025a; Shi et al., 2026). These approaches can expand or reshape the behaviors produced by the VLA, with retention and computational cost depending on the update recipe. In continual VLA adaptation, full-parameter RL can cause substantial forgetting of earlier skills, while LoRA-based updates can improve retention (Hu et al., 2026). Residual methods such as ResFiT (Ankile et al., 2025) leave the base policy intact and correct its output, whereas hybrid methods such as EXPO-FT (Dong et al., 2026) combine generator fitting with action editing and selection. Inference-time methods instead alter integration or select among generated candidates (Ruan et al., 2026; Yang et al., 2025; Giridhar et al., 2026). While prior designs aim to balance effectiveness, prior preservation, adaptation and inference efficiency, we argue that NSL provides a better option for this balance.

Learning through noise. DSRL (Wagenmaker et al., 2025) uses a conditional noise actor for frozen diffusion policies and adopts an actor-critic variant. UniSteer (Lu et al., 2026), FlowDAgger (Murray et al., 2026), and Flow Reversal Steering (Tang et al., 2026) use approximate inversion to convert action-level corrections into noise supervision. These methods share a noise-space interface and differ in feedback requirements and critic construction. Golden Ticket (Patil et al., 2026) improves generative robot policies by searching for a single, context-independent noise vector. LP-

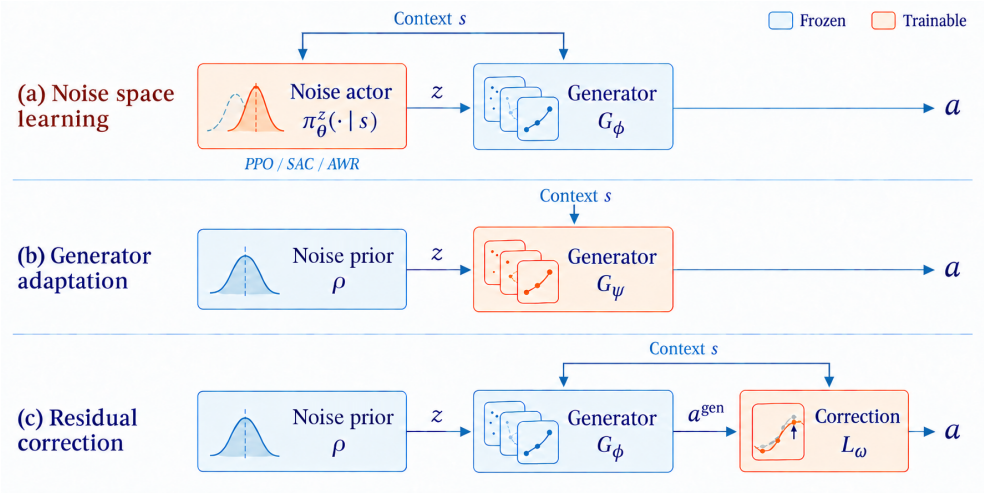


Figure 1: Adaptation interfaces for a deterministic generator. NSL learns a context-conditioned noise distribution, generator adaptation updates the noise-to-action map, and residual correction modifies generated actions before execution. Orange modules are trainable; blue modules are fixed.

DS (Simsir & Oguz, 2026) learns constrained noise perturbations for continuous control and robotic manipulation. LeaP (Dai et al., 2026) jointly learns a noise prior and generator for robotic imitation learning. Together, these studies establish noise-distribution design as a promising direction for improving generative robot policies. We build NSL upon these precedents and formulate it into a unified interface for adapting flow-matching-based VLAs.

3 Methods

3.1 Noise Space Learning

Let s denote the policy context, including the current observation and task instruction. Under deterministic inference, a pretrained VLA maps input noise $z \in \mathcal{Z}$ to an action chunk $a = G_\phi(s, z) \in \mathcal{A}$, including fixed action decoding and executed-prefix selection.

The standard noise prior ρ, typically $\mathcal{N}(0, I)$, induces a base action policy μ_ϕ. NSL replaces ρ with a learned context-conditioned noise distribution π_θ^z:

$$\mu_\phi(\cdot \mid s) = (G_{\phi,s})_\# \rho, \qquad \pi^a_{\theta,\phi}(\cdot \mid s) = (G_{\phi,s})_\# \pi^z_\theta(\cdot \mid s), \tag{1}$$

where $G_{\phi,s} = G_\phi(s, \cdot)$ and $\#$ denotes pushforward. NSL trains the noise actor while keeping the backbone and inference procedure fixed; Appendix A extends the formulation to stochastic generators.

At decision n, let R_n be the reward accumulated within the executed chunk and Γ_n its continuation discount. Sampling $Z_n \sim \pi^z_\theta(\cdot \mid S_n)$ and executing $A_n = G_\phi(S_n, Z_n)$, NSL maximizes

$$J(\theta; \phi) := \mathbb{E}\left[\sum_{n \geq 0} \left(\prod_{j<n} \Gamma_j\right) R_n\right] \tag{2}$$

over θ with ϕ fixed. Appendix B.1 gives the chunk-level reward and discount definitions.

3.2 Adaptation Interfaces and Information Cost

We compare NSL, generator fine-tuning, and residual correction (Figure 1) through their information costs. The central question is when a desired change in action distribution can be realized entirely by reweighting noise.

Interventions on the pretrained computation. Let A^{gen} denote the generator output and A the action chunk after any correction. Write $K_\phi(da^{\text{gen}} \mid s, z) = \delta_{G_\phi(s,z)}(da^{\text{gen}})$ for the generator and

$L_0(da \mid s, a^{\text{gen}}) = \delta_{a^{\text{gen}}}(da)$ for identity correction. The reference law and three single-factor interventions on (Z, A^{gen}, A), conditional on s, are

$$\begin{aligned} P_0 &= \rho K_\phi L_0, \quad P_{\text{NSL}} = \pi^z K_\phi L_0, \\ P_{\text{gen}} &= \rho K_\psi L_0, \quad P_{\text{corr}} = \rho K_\phi L_\omega. \end{aligned} \tag{3}$$

Different adaptions π^z, K_ψ, and L_ω change noise sampling, generation, and correction, respectively.

A common behavioral information bound. Fix a context distribution d independently of the adapted policy, and let q_M^a be the executed-action marginal of interface $M \in \{\text{NSL}, \text{gen}, \text{corr}\}$. Data processing (van Erven & Harremoës, 2012) gives a behavioral lower bound on its intervention information cost:

$$\begin{aligned} \mathcal{C}_M &:= \mathbb{E}_{s\sim d} D_{\text{KL}}(P_M(\cdot \mid s) \| P_0(\cdot \mid s)) \\ &\geq \mathcal{B}_M := \mathbb{E}_{s\sim d} D_{\text{KL}}(q_M^a(\cdot \mid s) \| \mu_\phi(\cdot \mid s)). \end{aligned} \tag{4}$$

For NSL, $\mathcal{C}_{\text{NSL}} = \mathbb{E}_{s\sim d} D_{\text{KL}}(\pi^z(\cdot \mid s) \| \rho)$; the bound is attained when noise reweighting preserves the pretrained noise posterior given the action. Changes to deterministic kernels can give infinite joint KL. Appendix A.4 gives the conditional KL decomposition and shared-smoothing formulation.

Proposition 1 (Minimum-information noise reweighting)**.** *Fix s and a measurable deterministic $G_{\phi,s}$ on standard Borel spaces. Let $\mu_\phi = (G_{\phi,s})_\# \rho$ and $q \ll \mu_\phi$ with finite $D_{\text{KL}}(q\|\mu_\phi)$. Over unrestricted noise distributions,*

$$\inf_{\pi^z : (G_{\phi,s})_\# \pi^z = q} D_{\text{KL}}(\pi^z \| \rho) = D_{\text{KL}}(q \| \mu_\phi). \tag{5}$$

The bound is attained by $d\pi^{z,*}/d\rho = (dq/d\mu_\phi) \circ G_{\phi,s}$.

The construction reweights noise values by the target-to-base density ratio of their generated actions, preserving their relative probabilities within each action's preimage. The proof appears in Appendix A.3.

Implications for downstream adaptation. For a bounded measurable utility $u(s, a)$, let $\mathfrak{R}_{\text{beh}}(\tau)$ and $\mathfrak{R}_M(\tau)$ denote the infimal behavioral and intervention information costs under the same fixed-context utility requirement $\mathbb{E}_{d,q} u \geq \tau$ and reference law P_0.

Proposition 2 (Information-optimal downstream adaptation)**.** *Let $G_\phi : \mathcal{S} \times \mathcal{Z} \to \mathcal{A}$ be a measurable deterministic generator on standard Borel spaces. With unrestricted measurable conditional noise distributions and $\mathfrak{R}_{\text{beh}}(\tau) < \infty$,*

$$\mathfrak{R}_{\text{NSL}}(\tau) = \mathfrak{R}_{\text{beh}}(\tau) \leq \mathfrak{R}_M(\tau), \qquad M \in \{\text{gen}, \text{corr}\}. \tag{6}$$

NSL therefore matches the behavioral information lower bound while retaining the pretrained generator. Appendix A.5 provides the full definitions and proof.

3.3 Optimization Methods for Noise Space Learning

We use conditional diagonal Gaussian noise actors with positive variance and prior-matched initialization. PPO, SAC, and AWR update the noise actor and auxiliary value models without backpropagating through the frozen VLA. Write $\mathcal{D}$ for replay; Appendix B.1 gives the transition and target definitions.

NSL-PPO. Using recent rollouts and estimated advantages $\widehat{A}_n$, PPO (Schulman et al., 2017) maximizes

$$\mathbb{E}\left[\min\left(r_n(\theta)\widehat{A}_n, \text{clip}(r_n(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}_n\right)\right], \quad r_n(\theta) = \frac{\pi_\theta^z(z_n \mid s_n)}{\pi_{\text{old}}^z(z_n \mid s_n)}. \tag{7}$$

Here $\epsilon \in (0, 1)$ sets the clipping range; the likelihood ratio is evaluated directly under the noise actor.

NSL-SAC. SAC (Haarnoja et al., 2018) learns twin noise-space critics $Q_{\chi_i}^z$ from replay and minimizes

$$\mathbb{E}_{s\sim\mathcal{D},\, z\sim\pi_\theta^z}\left[\alpha \log \pi_\theta^z(z \mid s) - \min_{i=1,2} Q_{\chi_i}^z(s, z)\right]. \tag{8}$$

Here $\alpha \geq 0$ is the entropy temperature. Actor gradients pass through the noise-space critic.

NSL-AWR. Our success-filtered AWR (Peng et al., 2019) variant fits the noise actor to replay noises with estimated advantage weights:

$$\max_{\theta} \mathbb{E}_{(s,z,\widehat{A},Y)\sim\mathcal{D}} \left[w \log \pi_\theta^z(z \mid s) \right], \qquad w = Y \min\{\exp(\widehat{A}/\beta), w_{\max}\}. \tag{9}$$

Here Y is the trajectory success indicator, $\beta > 0$ is the temperature, and $w_{\max} > 0$ caps the weights. Appendix A.6 derives the weighted-likelihood projection from KL-regularized improvement (Peng et al., 2019; Levine, 2018) and describes the effects of filtering and clipping.

4 Experiments

4.1 Experimental Setup

Benchmark and models. We evaluate SmolVLA and $\pi_{0.5}$ on LIBERO-10 (Liu et al., 2023), also known as LIBERO-Long. For each backbone, we use 10,000-step SFT checkpoints trained on either one demonstration per task (*One*, 10 total) or all 379 filtered demonstrations (*Full*), yielding four base policies. Each NSL run trains one shared, task-conditioned noise actor across all ten tasks. Compared methods start from the same checkpoint within each backbone–data setting. We present LIBERO results in the main text and RoboTwin results in Appendix C.

Methods and baselines. We compare NSL-PPO, NSL-SAC, and NSL-AWR. All three freeze the VLA and condition the noise actor on pooled VLA hidden features and robot proprioception. Baselines include the base VLA policy, Action-AWR for generator adaptation, and RLT for residual adaptation.

Experimental settings. On LIBERO-10, each training epoch covers 50 benchmark reset states per task, with eight stochastic rollouts per task–state pair, yielding 4,000 trajectories. Mixed-task batches contain up to 64 such pairs. Eight workers collect experience for a centralized learner, and task-aware replay balances tasks and samples successful and failed experience. AWR restricts actor regression to successful samples while training its value model on both outcomes. SAC and AWR retain replay across collection batches; PPO clears its buffer after each collection–update cycle. The policies predict 50 actions and execute the first ten before replanning, with a maximum episode length of 520 environment steps.

Evaluation metrics. After each epoch, we evaluate all 500 task–reset pairs in LIBERO-10 once and report the mean success rate across all ten tasks. The learning curves show the first 60 epochs.

Real-robot tasks. We study cup stacking, toy fishing, sequential insertion, and bead placement using task-specific $\pi_{0.5}$ policies initialized by supervised fine-tuning. Within each task, methods start from the same base policy. We compare NSL-AWR with Action-AWR and RLT under autonomous adaptation, and NSL-AWR with human guidance against DAgger and UniSteer. Each method is evaluated over 40 trials per task; we report success rates and cumulative training time averaged over tasks. Results appear in Section 4.7, with task definitions and collection details in Appendix B.2.

4.2 Results of Noise Space Learning

Figure 2 presents the learning curves of the three NSL methods across four base policies. All three improve over their base-policy references, with NSL-SAC achieving the highest peak scores, followed by NSL-AWR. In the Full setting, peak success rates reach 95.8% for SmolVLA and 99.8% for $\pi_{0.5}$; in the One setting, they rise from 24.4% and 40.4% to 77.0% and 71.2%, respectively. NSL-PPO also improves the base policies, but its sustained gains are more limited, with some configurations reaching a plateau or declining later in training. With the VLA frozen throughout, these gains show that learning only a conditional noise distribution can effectively improve the pretrained generator's task performance. This is consistent with the noise-reweighting mechanism characterized by Proposition 1: redistributing noise probabilities changes the generated behavior distribution while preserving the generator map. The high Full scores and substantial One gains demonstrate the practical effectiveness of this mechanism across demonstration budgets. Appendix B.4 examines NSL-AWR's noise evolution on fixed observations.

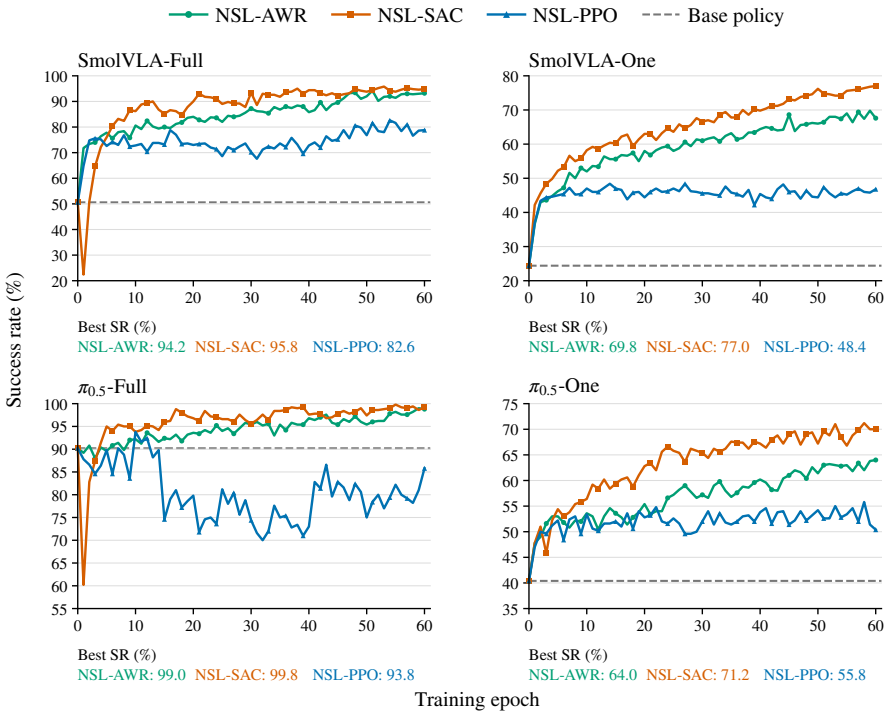


Figure 2: Noise space learning on LIBERO-10 with SmolVLA and $\pi_{0.5}$ under Full and One demonstration settings. Curves show success rates over 500 evaluation episodes at each epoch; dashed lines denote the corresponding base-policy references. Values below each panel report the best observed success rate within 60 epochs.

These results use AWR $1\times$, PPO $2\times$, and SAC $4\times$ update budgets. The methods share the collection schedule but differ in optimization compute. We examine this dependence through update-budget ablations, varying the number of updates while holding the collection protocol fixed.

4.3 Comparison with Generator and Residual Adaptation

Table 1 compares NSL with Action-AWR for generator adaptation and RLT for residual adaptation. Action-AWR freezes the VLM and updates the action expert through advantage-weighted flow matching on successful samples. Under the same collection schedule, NSL-SAC achieves the highest average SR of 83.5%, exceeding RLT's 81.0% and Action-AWR's 79.4%. NSL-AWR achieves a competitive average SR of 79.1%, close to Action-AWR. Across the individual model settings, NSL-AWR matches RLT on SmolVLA-Full and falls below both baselines on $\pi_{0.5}$-One.

NSL combines this performance with substantially lower update time. Averaged over the two models, NSL-SAC takes 9.93 seconds per collection batch, approximately $1/21.2$ of Action-AWR's update time and $1/11.2$ of RLT's. NSL-AWR delivers its competitive performance with the lowest update time of 1.09 seconds per batch. From an information-cost perspective, Proposition 2 establishes that NSL can attain the common behavioral information lower bound while retaining the pretrained generator. The experiments further demonstrate the practical value of this adaptation interface: by optimizing a lightweight noise actor without backpropagating through the VLA, NSL achieves competitive task performance at a low update cost.

4.4 Analysis of Update Budget

We compare update budgets on SmolVLA-Full and $\pi_{0.5}$-One while keeping the rollout protocol fixed. For each algorithm, we scale gradient updates per collection batch to $1\times$, $2\times$, and $4\times$ its baseline schedule. The top row of Figure 3 reports the mean success rate over the two models.

Table 1: Performance and update time on two model settings. Adapted-policy SR is the best observed within 60 epochs, with 500 evaluation episodes per epoch. Avg. is the equal-weight mean of the two model settings; Base policy gives the unadapted reference. Time is the gradient-update time per collection batch averaged over both models, excluding collection and evaluation.

Model / metric	Base policy	Action-AWR	RLT	NSL-AWR	NSL-PPO	NSL-SAC
SmolVLA-Full SR (%)	50.6	93.0	94.2	94.2	82.6	**95.8**
$\pi_{0.5}$-One SR (%)	40.4	65.8	67.8	64.0	55.8	**71.2**
Avg. SR (%)	45.5	79.4	81.0	79.1	69.2	**83.5**
Update time (s/batch)	—	210.20	111.60	**1.09**	2.34	9.93

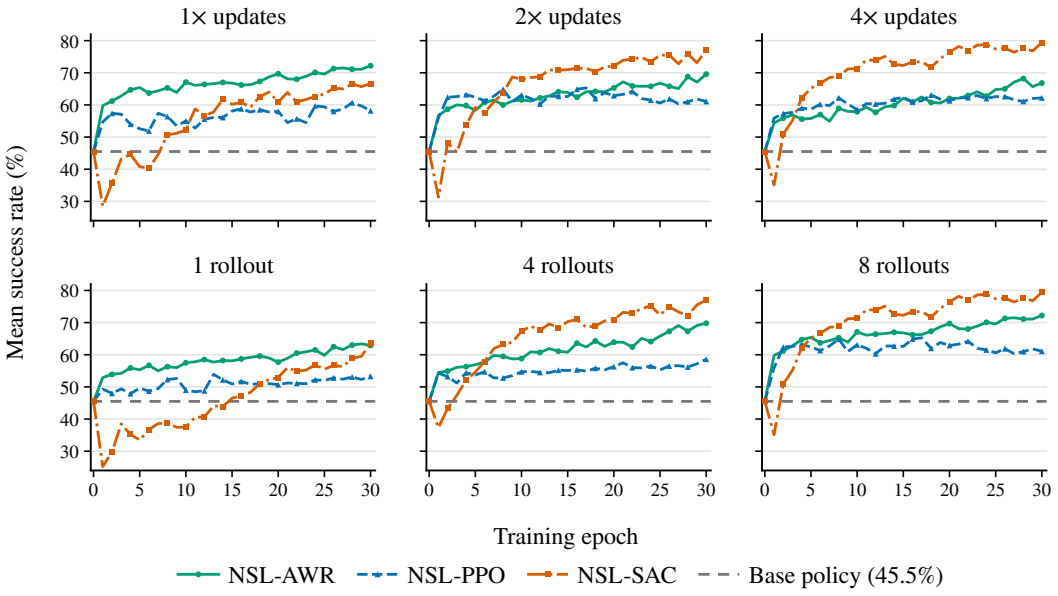


Figure 3: Effects of update budget and rollout count, averaged equally over SmolVLA-Full and $\pi_{0.5}$-One. Top: $1\times$, $2\times$, and $4\times$ update budgets with the rollout protocol fixed. Bottom: one, four, and eight rollouts per task–reset pair with AWR $1\times$, PPO $2\times$, and SAC $4\times$ updates. Dashed lines show the mean base-policy success rate (45.5%). Per-algorithm views appear in Appendix B.3.

Increasing updates has algorithm-dependent effects. NSL-AWR performs best at $1\times$, with further updates reducing success rates. NSL-SAC benefits substantially from more updates, although the additional gain from $2\times$ to $4\times$ is smaller. NSL-PPO gains less: additional updates improve early performance, but the learning curves still plateau later in training.

These patterns reflect different responses to repeated use of collected data. NSL-AWR's weighted regression on successful noises achieves strong gains with fewer updates, whereas NSL-SAC benefits from additional value and policy updates. Thus, NSL-AWR performs better under smaller update budgets, while NSL-SAC's advantage emerges as the update budget increases.

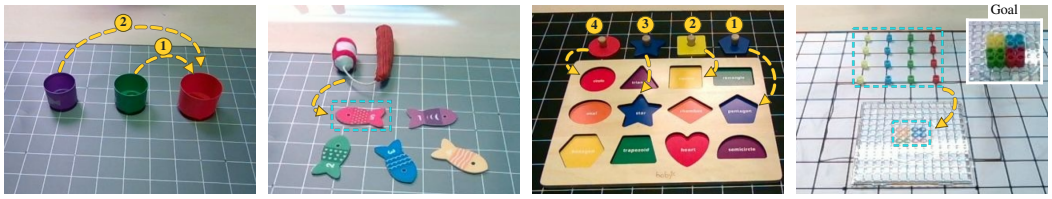
4.5 Analysis of Rollout Count

We compare one, four, and eight rollouts per task–reset pair on SmolVLA-Full and $\pi_{0.5}$-One, keeping the update ratios per collected transition unchanged at AWR $1\times$, PPO $2\times$, and SAC $4\times$. The bottom row of Figure 3 reports the mean success rate over the two models. More rollouts improve all three methods, with SAC benefiting most, although the additional gains from four to eight rollouts are smaller.

AWR learns faster early in training when collection is limited, while SAC gradually catches up. As collection increases, SAC overtakes AWR earlier and achieves higher final success. This difference is consistent with how the methods use experience: AWR directly performs weighted regression on noises from successful trajectories, allowing it to use observed successes quickly. SAC instead relies on its critic to guide policy updates; more rollouts provide richer state–noise training samples that

Table 2: Component ablations. Values are the best success rates (%) within the first 30 epochs, with changes from the corresponding baseline in parentheses (pp). PPO and AWR already use unbounded noise, so their no-limit entries reuse the baseline.

Model	Method	Baseline (%)	w/o Noise Limit	w/o Task Balancing	w/o Success/Failure Balancing	All Removed
SmolVLA Full	SAC	93.4	32.0 (−61.4)	80.6 (−12.8)	85.2 (−8.2)	0.0 (−93.4)
	PPO	78.8	78.8 (−0.0)	69.8 (−9.0)	65.0 (−13.8)	60.6 (−18.2)
	AWR	87.2	87.2 (−0.0)	87.2 (−0.0)	87.0 (−0.2)	90.2 (+3.0)
$\pi_{0.5}$-One	SAC	66.6	56.6 (−10.0)	60.4 (−6.2)	61.8 (−4.8)	39.8 (−26.8)
	PPO	54.8	54.8 (−0.0)	48.0 (−6.8)	50.6 (−4.2)	43.4 (−11.4)
	AWR	59.0	59.0 (−0.0)	61.6 (+2.6)	58.4 (−0.6)	60.2 (+1.2)



(a) Cup stacking (b) Toy fishing (c) Sequential insertion (d) Bead placement

Figure 4: Real-robot tasks. Arrows indicate operations, and numbers denote their order. The inset shows a completed bead-placement goal from a successful demonstration.

support value-driven optimization. PPO's overall performance also improves with more rollouts, but its learning curves still plateau relatively early, with limited gains later in training.

4.6 Analysis of Training Components

We ablate *noise limits* on generator inputs, *task balancing* across rollout batches and replay storage and sampling, and *success/failure balancing* in update batches on SmolVLA-Full and $\pi_{0.5}$-One. Removing all components uses unbounded noise and uniform sampling from a global replay buffer. Update objectives remain unchanged, including NSL-AWR's actor regression on successful samples. Table 2 summarizes the results.

NSL-SAC and NSL-PPO benefit from task and success/failure balancing on both models. NSL-SAC is particularly sensitive to removing noise limits, whereas NSL-AWR maintains comparable performance even with all components removed. The benefits of these components therefore depend on the optimizer.

Figure 10 in Appendix B.5 examines NSL-SAC diagnostic runs on SmolVLA-Full. Without a noise limit, the actor's Gaussian means drift farther from initialization on fixed contexts, while the upper tail of critic targets remains above the return bound. Both quantities are lower with bounded noise. This pattern is consistent with value-driven outward drift in noise space. NSL-AWR instead fits observed successful noises through weighted regression, consistent with its lower sensitivity to these components.

4.7 Real-Robot Experiments

We select NSL-AWR for the real-robot comparison based on its behavior in simulation. Sections 4.4–4.6 show that it learns quickly with fewer rollouts and low update budgets and is less sensitive to training components, making it suitable for costly robot interaction with small collection batches. NSL-SAC benefits more from increased collection and optimization, whereas NSL-PPO shows more limited later improvement. Unintended table contact also occurred in an earlier real-robot bead-placement SAC experiment. Appendix B.6 reports noise approaching the amplitude limit and actor proposals with higher predicted Q values and greater critic disagreement than replay noise. These observations further motivate our choice of NSL-AWR, which learns observed successful noises through weighted regression, and our study of its use with human corrections.

Table 3: Real-robot success rates over 40 trials per task and training time. Avg. SR and cumulative model-update time are averaged over the four tasks.

Task (SR, %)	Initial policy	*Autonomous*			*Human-guided*		
		Action-AWR	RLT	**NSL-AWR**	DAgger	UniSteer	**NSL-AWR + human**
Cup stacking	75.0	100.0	100.0	100.0	100.0	100.0	100.0
Toy fishing	55.0	70.0	75.0	80.0	70.0	90.0	90.0
Sequential insertion	15.0	50.0	75.0	70.0	70.0	85.0	90.0
Bead placement	5.0	15.0	30.0	35.0	25.0	20.0	55.0
Avg.	37.5	58.8	70.0	71.3	66.3	73.8	83.8
Training time (min)	—	52.4	23.2	1.53	61.6	49.8	2.04

Figure 4 illustrates cup stacking, toy fishing, sequential insertion, and bead placement. Table 3 shows that autonomous NSL-AWR improves mean success from the initial policy's 37.5% to 71.3%, exceeding Action-AWR's 58.8% and performing comparably to RLT's 70.0%. Performance varies across individual tasks, with RLT performing better on sequential insertion. Overall, learning a conditional noise distribution improves real-robot behavior with the VLA frozen and achieves performance comparable to generator and residual adaptation.

With human guidance, NSL-AWR's mean success further increases to 83.8%, exceeding 66.3% for DAgger and 73.8% for UniSteer. The largest gains occur on sequential insertion and bead placement, where success rises from 70% and 35% to 90% and 55%, respectively. These improvements illustrate how human corrections complement autonomous exploration by supplying useful behaviors that are difficult to obtain through autonomous collection alone, enabling noise space learning to further improve complex manipulation.

NSL-AWR also has a lower training cost. Mean cumulative training time per task is 1.53 minutes for autonomous NSL-AWR, compared with 52.4 minutes for Action-AWR and 23.2 minutes for RLT. With human guidance, NSL-AWR takes 2.04 minutes. Figure 5 in Appendix B.2 further shows that NSL-AWR with human guidance uses fewer human trajectories per round than UniSteer on every task while achieving higher mean success. These results demonstrate NSL-AWR's practical value on real robots: improving policies at low update cost and further enhancing task performance through human corrections.

5 Discussion and Conclusion

We study Noise Space Learning, which improves task performance by learning a conditional noise distribution while keeping the VLA frozen. Our theoretical analysis characterizes the relationship between noise reweighting and changes in behavior distributions, and establishes conditions under which NSL attains the common behavioral information lower bound. Simulation and real-robot experiments further show that this adaptation interface effectively uses pretrained action generators, achieving performance competitive with generator and residual adaptation at a low gradient-update cost.

Different optimizers suit different resource conditions. NSL-AWR quickly uses observed successful experience, performs well with limited collection and update budgets, and is less sensitive to training components. NSL-SAC benefits more from additional data and updates, achieving higher success rates in simulation when larger budgets are available. NSL-PPO also improves the base policies, but its sustained gains are more limited. These findings suggest choosing the optimizer according to the available interaction and computation budgets. Real-robot experiments demonstrate the practicality of NSL-AWR for online learning with small collection batches, while human corrections further improve performance on complex manipulation tasks.

The adaptation capacity of NSL remains constrained by the fixed generator and the expressiveness of the noise actor. Richer noise parameterizations and combinations of noise learning with selective generator updates are promising directions for further study. Future work should also examine generalization to new tasks and robot embodiments, retention of existing skills, and execution reliability in complex contact interactions.

AI USE STATEMENT

We used GPT-5.6 and GPT-6 through GitHub Copilot and the Copilot SDK to assist with writing mathematical proofs and implementing the proposed methods. Generative AI was not used for the other research tasks requiring disclosure under the ICLR 2027 AI Policy for Authors. All AI-assisted content was reviewed by the authors, who take full responsibility for the final work.

ETHICS STATEMENT

This work does not involve human subjects or personal or sensitive data. Real-robot experiments were conducted in controlled environments with human supervision, emergency stops, and constraints on robot motion and force. The proposed methods are not intended for unsupervised or safety-critical deployment.

REPRODUCIBILITY STATEMENT

The main paper describes the proposed algorithms. The appendix provides complete proofs, ablation studies, and additional experimental and implementation details. The authors intend to release the code after publication.

REFERENCES

Lars Ankile, Zhenyu Jiang, Rocky Duan, Guanya Shi, Pieter Abbeel, and Anusha Nagabandi. Residual Off-Policy RL for Finetuning Behavior Cloning Policies, 2025. URL https://arxiv.org/abs/2509.19301.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control, 2024. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, et al. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control, 2023. URL https://arxiv.org/abs/2307.15818.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, Tiejun Huang, Yu Wang, and Chao Yu. π_{RL}: Online RL Fine-tuning for Flow-based Vision-Language-Action Models, 2025a. URL https://arxiv.org/abs/2510.25889.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan-ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A Scalable Data Generator and Benchmark with Strong Domain Randomization for Robust Bimanual Robotic Manipulation, 2025b. URL https://arxiv.org/abs/2506.18088.

Meipo Dai, Qiyuan Zhuang, He-Yang Xu, Ying-Jie Shuai, Yijun Wang, Qi Dou, and Xiu-Shen Wei. Where Should Action Generation Begin? A Learnable Source Prior for Generative Robot Policies, 2026. URL https://arxiv.org/abs/2606.17408.

Perry Dong, Kuo-Han Hung, Tian Gao, Dorsa Sadigh, and Chelsea Finn. EXPO-FT: Sample-Efficient Reinforcement Learning Finetuning for Vision-Language-Action Models, 2026. URL https://arxiv.org/abs/2605.25477.

Varun Giridhar, Anant Khandelwal, Jeremy A. Collins, Ignat Georgiev, and Animesh Garg. Beyond Imitation: Self-Improving Robot Policies via Off-Policy Q-Planning, 2026. URL https://arxiv.org/abs/2608.21204.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, 2018. URL https://arxiv.org/abs/1801.01290.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models, 2021. URL https://arxiv.org/abs/2106.09685.

Jiaheng Hu, Jay Shim, Chen Tang, Yoonchang Sung, Bo Liu, Peter Stone, and Roberto Martin-Martin. Simple Recipe Works: Vision-Language-Action Models are Natural Continual Learners with Reinforcement Learning, 2026. URL https://arxiv.org/abs/2603.11653.

Suning Huang, Jiaqi Shao, Ke Wang, Qianzhong Chen, Jiankai Sun, Yanjiang Guo, Mac Schwager, and Jeannette Bohg. Breaking Lock-In: Preserving Steerability under Low-Data VLA Post-Training, 2026. URL https://arxiv.org/abs/2604.23121.

Yuhua Jiang, Junjie Lu, Xinyao Qin, Xiaoyu Chen, Kaixin Wang, Feifei Gao, and Li Zhao. VLA-GSE: Boosting Parameter-Efficient Fine-Tuning in VLA with Generalized and Specialized Experts, 2026. URL https://arxiv.org/abs/2605.06175.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An Open-Source Vision-Language-Action Model, 2024. URL https://arxiv.org/abs/2406.09246.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success, 2025. URL https://arxiv.org/abs/2502.19645.

Sergey Levine. Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review, 2018. URL https://arxiv.org/abs/1805.00909.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning, 2023. URL https://arxiv.org/abs/2306.03310.

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.

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.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An Open-Source Generalist Robot Policy, 2024. URL https://arxiv.org/abs/2405.12213.

Omkar Patil, Ondrej Biza, Thomas Weng, Karl Schmeckpeper, Wil Thomason, Xiaohan Zhang, Kausik Sivakumar, Robin Walters, Nakul Gopalan, Sebastian Castro, Stephen Hart, and Eric Rosen. You've Got a Golden Ticket: Improving Generative Robot Policies With A Single Noise Vector, 2026. URL https://arxiv.org/abs/2603.15757.

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

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.

Yifan Ruan, Chenyang Cao, Andreas Burger, Ali Pesaranghader, Kaveh Kamali, Jaehong Kim, Nandita Vijaykumar, Alan Aspuru-Guzik, Igor Gilitschenski, and Nicholas Rhinehart. QPILOTS: Efficient Test-Time Q-Steering for Flow Policies, 2026. URL https://arxiv.org/abs/2606.14801.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Kexin Shi, Junyao Shi, Poorvi Hebbar, Zhuolun Zhao, Tarun Amarnath, Yifan Su, Shikhar Bahl, and Deepak Pathak. FlowDPG: Deterministic Policy Gradient on Flow Matching Policies for Real-World Manipulation, 2026. URL https://arxiv.org/abs/2606.22303.

Hikmet Simsir and Ozgur S. Oguz. Lagrangian Perturbation Diffusion Steering: Latent Reinforcement Learning for Generative Policies, 2026. URL https://arxiv.org/abs/2606.01151.

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.

Tim van Erven and Peter Harremoës. Rényi Divergence and Kullback-Leibler Divergence, 2012. URL https://arxiv.org/abs/1206.2459.

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, 2025. URL https://arxiv.org/abs/2506.15799.

Siyuan Yang, Yang Zhang, Haoran He, Ling Pan, Xiu Li, Chenjia Bai, and Xuelong Li. Steering Vision-Language-Action Models as Anti-Exploration: A Test-Time Scaling Approach, 2025. URL https://arxiv.org/abs/2512.02834.

Chuheng Zhang, Rushuai Yang, Xiaoyu Chen, Kaixin Wang, Li Zhao, Yi Chen, and Jiang Bian. How Do VLAs Effectively Inherit from VLMs?, 2025a. URL https://arxiv.org/abs/2511.06619.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. ReinFlow: Fine-tuning Flow Matching Policy with Online Reinforcement Learning, 2025b. URL https://arxiv.org/abs/2505.22094.

A Information-Cost Analysis

A.1 Definitions and Scope

All logarithms are natural, so KL is measured in nats. For probability measures P, Q, define $D_{\mathrm{KL}}(P\|Q) = \int \log(dP/dQ)\, dP$ if $P \ll Q$, and $+\infty$ otherwise. Standard Borel spaces ensure regular conditional distributions exist. We condition on a fixed context s unless an expectation over the externally fixed d is displayed. Assume the divergences used in equalities are finite, avoiding subtraction of infinite terms.

For stochastic inference, replace the deterministic map in Eq. (1) with a fixed conditional kernel $K_\phi(da \mid s, z)$. The base and adapted action policies are

$$\mu_\phi(da \mid s) = \int K_\phi(da \mid s, z)\rho(dz), \qquad \pi^a_{\theta,\phi}(da \mid s) = \int K_\phi(da \mid s, z)\pi^z_\theta(dz \mid s). \tag{10}$$

For NSL, $A = A^{\mathrm{gen}}$ because the correction kernel is the identity. Marginalizing Eq. (3) over A^{gen} gives the following laws on (Z, A), with P_0 also denoting the reference marginal and P_π denoting the marginal of P_{NSL}:

$$P_0(dz, da \mid s) = \rho(dz)K_\phi(da \mid s, z), \qquad P_\pi(dz, da \mid s) = \pi^z(dz \mid s)K_\phi(da \mid s, z). \tag{11}$$

Their action marginals are μ_ϕ and $\pi^a = q^a_{\mathrm{NSL}}$. This analysis concerns the law supplied to the *fixed* kernel. If a stochastic decoder uses additional independent randomness, that randomness remains part of the kernel. Allowing an actor to control all decoder randomness defines a different adaptation interface.

A.2 Noise and Behavioral Information

Apply the KL chain rule in two orders to Eq. (11). Conditioning first on Z, the equal kernels cancel:

$$D_{\mathrm{KL}}(P_\pi\|P_0) = D_{\mathrm{KL}}(\pi^z\|\rho) + \mathbb{E}_{Z\sim\pi^z} D_{\mathrm{KL}}(K_\phi(\cdot \mid s, Z)\|K_\phi(\cdot \mid s, Z)) \tag{12}$$

$$= D_{\mathrm{KL}}(\pi^z\|\rho). \tag{13}$$

Conditioning first on A gives

$$D_{\mathrm{KL}}(\pi^z\|\rho) = D_{\mathrm{KL}}(\pi^a\|\mu_\phi) + \mathbb{E}_{A\sim\pi^a} D_{\mathrm{KL}}(P_\pi(Z \mid s, A)\|P_0(Z \mid s, A)). \tag{14}$$

Integrating with respect to d gives

$$\mathcal{C}_{\mathrm{NSL}} = \mathcal{B}_{\mathrm{NSL}} + \mathbb{E}_{s\sim d,\, a\sim\pi^a} D_{\mathrm{KL}}\big(P_\pi(Z \mid s, a)\|P_0(Z \mid s, a)\big). \tag{15}$$

Nonnegativity gives the data-processing inequality, with equality exactly when the adapted and reference noise posteriors agree for $d(ds)\pi^a(da \mid s)$-almost every (s, a). No differentiability or invertibility of the generator is needed for this decomposition.

A.3 Proof of the Minimum-Information Lift

Proof of Proposition 1. Write $G = G_{\phi,s}$, $\mu = G_\#\rho$, and $f = dq/d\mu$. Define π^* by $d\pi^*/d\rho(z) = f(G(z))$. It is a probability measure since

$$\int f(G(z))\rho(dz) = \int f(a)\mu(da) = 1. \tag{16}$$

For every measurable set $E \subseteq \mathcal{A}$,

$$\pi^*(G^{-1}(E)) = \int_E f(a)\mu(da) = q(E), \tag{17}$$

so $G_\#\pi^* = q$. Its information cost is

$$D_{\mathrm{KL}}(\pi^*\|\rho) = \int f(G(z)) \log f(G(z))\rho(dz) = \int f(a) \log f(a)\mu(da) = D_{\mathrm{KL}}(q\|\mu), \tag{18}$$

using $0 \log 0 = 0$. Every other noise distribution with pushforward q has cost at least $D_{\mathrm{KL}}(q\|\mu)$ by Eq. (14); singular candidates have infinite cost and cannot improve the finite minimum. Thus π^* attains the infimum. Its weighting is constant on each fiber $G^{-1}(a)$, preserving the reference noise posterior conditioned on a. □

Why deterministic generation matters. The data-processing inequality holds for stochastic kernels, but the constructive equality need not. For example, let $K(A = 0 \mid z) = 1/4$ for every z. Every noise distribution induces the same action marginal $(1/4, 3/4)$. The target $(1/2, 1/2)$ is absolutely continuous with respect to this marginal but is unreachable by noise reweighting. For a deterministic generator, absolute continuity with respect to $G_{\#}\rho$ suffices for the unrestricted construction; this statement is not transferable to an arbitrary stochastic channel.

Fixed noise and coverage. A constant-noise policy, as considered by Golden Ticket (Patil et al., 2026), is a useful special case of the control interface but is singular relative to a continuous Gaussian prior. Its KL divergence from that prior is infinite; it lies outside the finite-information proposition. Moreover, a Gaussian prior can have full mathematical support while allocating negligible probability to useful regions. Reachability, high-probability coverage, and learnability under a finite budget are different properties.

A.4 Comparing Intervention Locations

For the joint laws on (Z, A^{gen}, A) in Eq. (3), absolutely continuous factors and finite costs give

$$\mathcal{C}_{\mathrm{NSL}} = \mathbb{E}_d D_{\mathrm{KL}}(\pi^z(\cdot \mid S) \| \rho), \tag{19}$$

$$\mathcal{C}_{\mathrm{gen}} = \mathbb{E}_{S\sim d, Z\sim\rho} D_{\mathrm{KL}}(K_\psi(\cdot \mid S, Z) \| K_\phi(\cdot \mid S, Z)), \tag{20}$$

$$\mathcal{C}_{\mathrm{corr}} = \mathbb{E}_{S\sim d, A^{\mathrm{gen}}\sim\mu_\phi} D_{\mathrm{KL}}(L_\omega(\cdot \mid S, A^{\mathrm{gen}}) \| L_0(\cdot \mid S, A^{\mathrm{gen}})). \tag{21}$$

When $\mathcal{C}_M < \infty$, the KL chain rule (van Erven & Harremoës, 2012) with $U = (Z, A^{\mathrm{gen}})$ gives, for $M \in \{\mathrm{NSL}, \mathrm{gen}, \mathrm{corr}\}$,

$$\begin{aligned} \mathcal{C}_M &= \mathcal{B}_M + \mathcal{M}_M \;\geq\; \mathcal{B}_M, \\ \mathcal{M}_M &= \mathbb{E}_{s\sim d,\, a\sim q_M^a} D_{\mathrm{KL}}\big(P_M(U \mid s, a) \| P_0(U \mid s, a)\big). \end{aligned} \tag{22}$$

The conditional mechanism divergence $\mathcal{M}_M$ measures redistribution of the internal variables given the executed action. This identifies a common behavioral lower bound, not a strict ordering of intervention costs. Hybrid interventions require the full chain rule with expectations under their adapted upstream factors. In particular, EXPO-FT's generator fitting, action edits, and candidate selection cannot be substituted for the single-factor correction law without redefining the internal variables.

Deterministic-kernel singularity. Distinct point masses have infinite KL. If deterministic G_ψ and G_ϕ differ on a set of positive probability under $d(ds)\rho(dz)$, then $\mathcal{C}_{\mathrm{gen}} = +\infty$. With $L_0 = \delta_{a^{\mathrm{gen}}}$, a correction assigning positive joint probability to $A \neq A^{\mathrm{gen}}$ likewise gives $\mathcal{C}_{\mathrm{corr}} = +\infty$. For finite-cost comparisons, a common Gaussian relaxation with $\sigma > 0$, such as $K_\phi^\sigma = \mathcal{N}(G_\phi, \sigma^2 I)$ and $K_\psi^\sigma = \mathcal{N}(G_\psi, \sigma^2 I)$, gives conditional KL $\|G_\psi - G_\phi\|^2/(2\sigma^2)$. The expected cost is finite when the squared displacement is integrable. An analogous shared scale applies to deterministic correction kernels. All marginals and decomposition terms are computed for the relaxed system, and the resulting costs depend on the chosen scale. Proposition 1 applies directly to the original NSL system because its deterministic map is unchanged.

A.5 Utility-Based Information Costs

For a bounded measurable fixed-context utility $u(s, a)$ and threshold τ, define

$$\mathfrak{R}_{\mathrm{beh}}(\tau) = \inf_{q:\mathbb{E}_{d,q}u\geq\tau} \mathbb{E}_d D_{\mathrm{KL}}(q(\cdot \mid S) \| \mu_\phi(\cdot \mid S)). \tag{23}$$

For each interface, use the same d, u, τ, and reference law P_0. Let $\xi_{\mathrm{NSL}} = \pi^z$, $\xi_{\mathrm{gen}} = K_\psi$, and $\xi_{\mathrm{corr}} = L_\omega$ range over their admissible adapted factors. Define

$$\mathfrak{R}_M(\tau) = \inf_{\xi_M:\mathbb{E}_{d,q_M^a}u\geq\tau} \mathcal{C}_M(\xi_M), \qquad M \in \{\mathrm{NSL}, \mathrm{gen}, \mathrm{corr}\}, \tag{24}$$

with value $+\infty$ for an empty feasible set.

Proof of Proposition 2. Every feasible intervention induces a feasible action policy. Data processing gives

$$\mathcal{C}_M(\xi_M) \geq \mathbb{E}_d D_{\mathrm{KL}}(q_M^a \| \mu_\phi) \geq \mathfrak{R}_{\mathrm{beh}}(\tau). \tag{25}$$

Taking infima over the admissible factors proves $\mathfrak{R}_M(\tau) \geq \mathfrak{R}_{\mathrm{beh}}(\tau)$ for all three interfaces, including empty feasible sets and infinite intervention costs.

To prove the reverse inequality for NSL, fix $\varepsilon > 0$. Since $\mathfrak{R}_{\mathrm{beh}}(\tau) < \infty$, there is a feasible measurable action kernel q_ε with

$$\mathbb{E}_d D_{\mathrm{KL}}(q_\varepsilon(\cdot \mid S) \| \mu_\phi(\cdot \mid S)) \leq \mathfrak{R}_{\mathrm{beh}}(\tau) + \varepsilon. \tag{26}$$

Write $Q_\varepsilon(ds, da) = d(ds) q_\varepsilon(da \mid s)$ and $Q_0(ds, da) = d(ds)\mu_\phi(da \mid s)$. Finite integrated KL implies $Q_\varepsilon \ll Q_0$. Choose a nonnegative jointly measurable Radon–Nikodym derivative $f_\varepsilon = dQ_\varepsilon / dQ_0$. The function $h_\varepsilon(s) = \int f_\varepsilon(s, a)\mu_\phi(da \mid s)$ is measurable and equals 1 for d-almost every s, since both joint laws have context marginal d. Define

$$\pi_\varepsilon^z(dz \mid s) = \begin{cases} f_\varepsilon(s, G_\phi(s, z))\rho(dz), & h_\varepsilon(s) = 1, \\ \rho(dz), & h_\varepsilon(s) \neq 1. \end{cases} \tag{27}$$

Measurability of G_ϕ makes this a measurable probability kernel, normalized at every context. Its action pushforward, integrated against d, equals Q_ε. Uniqueness of conditional distributions on standard Borel spaces therefore gives $(G_{\phi,s})_\# \pi_\varepsilon^z(\cdot \mid s) = q_\varepsilon(\cdot \mid s)$ for d-almost every s. This conditional noise distribution is admissible by the unrestricted-family assumption and has the same utility as q_ε.

The density construction is the minimum-information lift of Proposition 1 at almost every context, so

$$\mathcal{C}_{\mathrm{NSL}}(\pi_\varepsilon^z) = \mathbb{E}_d D_{\mathrm{KL}}(q_\varepsilon(\cdot \mid S) \| \mu_\phi(\cdot \mid S)) \leq \mathfrak{R}_{\mathrm{beh}}(\tau) + \varepsilon. \tag{28}$$

Taking the infimum over noise distributions and then letting $\varepsilon \downarrow 0$ yields $\mathfrak{R}_{\mathrm{NSL}}(\tau) \leq \mathfrak{R}_{\mathrm{beh}}(\tau)$. Together with the shared lower bound, this proves Eq. (6). The argument does not require an attained utility optimizer. □

The comparison establishes that NSL with unrestricted noise distributions matches the common behavioral requirement in the deterministic system. Strict separation from another interface would require a positive gap in its own infimum. For the stochastic relaxations in Appendix A.4, the shared lower bound uses the relaxed reference marginal and induced policies throughout; equality for the unchanged deterministic generator is a separate result.

A.6 Regularized Noise Actor Improvement and AWR

Fix s, a reference noise distribution η, a measurable value score $Q(z)$, and $\beta > 0$. Assume $0 < \Lambda = \int e^{Q(z)/\beta}\eta(dz) < \infty$ and the required expectations exist. Define $q^*(dz) = e^{Q(z)/\beta}\eta(dz)/\Lambda$. For any admissible $q \ll \eta$,

$$\mathbb{E}_q Q - \beta D_{\mathrm{KL}}(q \| \eta) = \beta \log \Lambda - \beta D_{\mathrm{KL}}(q \| q^*). \tag{29}$$

Nonnegativity proves optimality of q^* for the regularized objective. Subtracting a context-only baseline from Q leaves the normalized conditional target unchanged. This is the standard regularized-improvement argument underlying weighted regression (Peng et al., 2019; Levine, 2018), not a new NSL-specific theorem.

For replay law $P_{\mathcal{D}}(ds, dz) = d_{\mathcal{D}}(ds)\eta_{\mathcal{D}}(dz \mid s)$, take any nonnegative weight w with $0 < \overline{\Lambda} = \mathbb{E}_{P_{\mathcal{D}}} w < \infty$. Set

$$\Lambda(s) = \mathbb{E}_{\eta_{\mathcal{D}}} w(s, Z), \quad q^*(dz \mid s) = \frac{w(s, z)\eta_{\mathcal{D}}(dz \mid s)}{\Lambda(s)}, \quad \widetilde{d}(ds) = \frac{\Lambda(s) d_{\mathcal{D}}(ds)}{\overline{\Lambda}}. \tag{30}$$

The conditional target is needed only where $\Lambda(s) > 0$. On the weighted context distribution,

$$\mathbb{E}_{\widetilde{d}} D_{\mathrm{KL}}(q^*(\cdot \mid S) \| \pi_\theta^z(\cdot \mid S)) = \mathrm{const} - \frac{1}{\overline{\Lambda}} \mathbb{E}_{P_{\mathcal{D}}}[w(S, Z) \log \pi_\theta^z(Z \mid S)]. \tag{31}$$

Thus weighted likelihood is an exact forward-KL projection of the population target under the *reweighted* context marginal, not generally under the original $d_{\mathcal{D}}$. The identity assumes finite cross-entropies. For a discrete empirical replay measure fitted by a continuous density, their measure-level KL is infinite; the finite empirical weighted-log-density objective is instead a Monte Carlo approximation to this population projection, not a literal finite empirical KL.

Filtering and clipping. With random success Y and advantage estimate $\widehat{A}$, the effective population weight is

$$\overline{w}(s,z) = \mathbb{E}\left[Y \min\{e^{\widehat{A}/\beta}, w_{\max}\} \mid s, z\right]. \tag{32}$$

Replacing w by $\overline{w}$ gives the filtered target. Filtering and clipping generally change the unclipped Gibbs optimum. A statewise advantage baseline cancels in an *unclipped conditional* target, but can change context weights in a shared-parameter empirical fit; clipping can also change conditional relative weights. With no successful replay, the weighted objective supplies no actor-learning signal. These qualifications prevent conflating success-only BC, standard AWR, and the authors' filtered variant.

A.7 Why Behavior Anchoring Is Not a Safety Bound

For $\pi^z = \mathcal{N}(m, \mathrm{diag}(\sigma^2))$ and $\rho = \mathcal{N}(0, I)$ in dimension d_z,

$$D_{\mathrm{KL}}(\pi^z \| \rho) = \tfrac{1}{2}\sum_{j=1}^{d_z}(m_j^2 + \sigma_j^2 - 1 - \log\sigma_j^2), \qquad \mathcal{H}(\pi^z) = \tfrac{1}{2}\sum_{j=1}^{d_z}\log(2\pi e\sigma_j^2). \tag{33}$$

Entropy contains no mean penalty and therefore does not control mean drift relative to the pretrained prior. Conversely, weighted maximum likelihood can fit a replay distribution whose mean is already large; a Gaussian fit can also assign mass to regions absent from finite replay. A moving policy reference does not provide a fixed-prior bound either: for $\pi_k = \mathcal{N}(kv, I)$, $D_{\mathrm{KL}}(\pi_{k+1}\|\pi_k) = \|v\|^2/2$ but $D_{\mathrm{KL}}(\pi_K\|\rho) = K^2\|v\|^2/2$. This is an illustration about local versus global constraints, not a literal guarantee supplied by PPO clipping.

No inequality above translates noise moments into collision, force, or joint-limit guarantees. Such a guarantee would require additional assumptions on the generator, environment, controller, and safe set. Physical safety remains an experimental and engineering requirement.

B Experimental Details and Additional Results

B.1 Implementation Details

Network architecture and noise transformations. For LIBERO, all three noise actors take pooled VLA hidden features concatenated with proprioception as input. Each actor has four hidden layers of width 512, with LayerNorm and ReLU after each linear layer, and separate 32-dimensional mean and log-standard-deviation heads. The heads are initialized to zero, and log standard deviations are clamped to $[-20, 2]$. The sampled noise vector is repeated across the 50-step prediction horizon. In the main SAC setting, the noise is bounded to $(-1, 1)$ after the initial Gaussian exploration batch.

PPO and AWR each use a separate value MLP with four 512-unit ReLU hidden layers and a 16-bin categorical output over $[0, 1]$, trained with cross-entropy loss. SAC uses two independent scalar-output critics, each with four 512-unit hidden layers, LayerNorm, and ReLU; their inputs concatenate the context features and the noise vector.

Hyperparameters. Table 4 lists the LIBERO settings. Actors and value/critic networks use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon_{\mathrm{Adam}} = 10^{-8}$, weight decay 0.01, and gradient-norm clipping at 1.0. AWR and SAC use log probabilities averaged over noise dimensions. For the PPO ratio in Eq. (7), joint log probabilities are obtained by summing over noise dimensions. PPO entropy and reference-KL terms use per-dimension averages.

Definition of the update multiplier. Let N be the number of new chunk transitions in a collection batch, $B = 2{,}048$ the optimization minibatch size, and $u \in \{1, 2, 4\}$ the update multiplier. AWR and PPO schedule $10uN/B$ joint actor–value updates and $10uN/B$ additional value-only updates, rounded to the nearest integer. SAC schedules $20uN/B$ twin-critic updates, with one actor update every two critic updates; fractional critic-update budgets are carried forward. Thus, $1\times$ corresponds to approximately $10N/B$ actor updates and $20N/B$ value/critic updates. For $N = 20{,}480$, this gives 100 actor updates and 200 value/critic updates. Table 5 gives the corresponding counts for the main configurations.

Table 4: Hyperparameters for the main LIBERO experiments. SAC learns its entropy temperature, initialized at 0.006, with Adam at learning rate 3×10^{-4}. Its target entropy is specified per noise dimension.

Parameter	NSL-AWR	NSL-PPO	NSL-SAC
Actor learning rate	10^{-5}	10^{-5}	10^{-5}
Value/critic learning rate	2×10^{-5}	2×10^{-5}	2×10^{-5}
Training minibatch size	2,048	2,048	2,048
Discount per chunk	0.99	0.99	0.99
Replay success sampling fraction	0.5	0.5	0.5
Random seed	42	42	42
AWR temperature β	0.5	—	—
AWR maximum weight $w_{\max}$	20	—	—
GAE λ	—	0.95	—
PPO clipping range ϵ	—	0.2	—
Actor entropy coefficient	0	0.01	Learned
Reference-KL coefficient	—	0.005	—
SAC target entropy	—	—	0.5
SAC target-update coefficient τ	—	—	0.005
SAC Monte Carlo auxiliary weight	—	—	0.5

Table 5: Scheduled updates for a collection batch containing 20,480 new chunk transitions. Counts scale with the actual number of collected transitions. Each SAC critic update jointly optimizes both Q networks.

Method	Main update multiplier	Actor updates	Value/critic updates
NSL-AWR	$1\times$	100	200
NSL-PPO	$2\times$	200	400
NSL-SAC	$4\times$	400	800

Chunk-level rewards and discounts. Index policy decisions by $n = 0, 1, \ldots$, and let c_n be the number of low-level controls actually executed before the next decision. Decision n starts at environment step $t_n = \sum_{j<n} c_j$, with $t_0 = 0$. With per-control rewards r_t, the LIBERO experiments sum rewards within each chunk and apply a fixed discount between chunks:

$$R_n = \sum_{j=0}^{c_n-1} r_{t_n+j}, \qquad \Gamma_n = \gamma_{\text{chunk}} = 0.99. \tag{34}$$

Targets distinguish environment termination from time-limit truncation. Each recorded transition contains the noise that generated the chunk, context, reward, successor context, executed duration, and termination and truncation indicators.

Optimizer specification. Section 3 gives the actor objectives. PPO advantages use the chunk rewards R_n and continuation discounts Γ_n. For a nonterminal chunk, a SAC soft backup is

$$R_n + \Gamma_n \mathbb{E}_{z' \sim \pi_\theta^z(\cdot|s')} \left[\min_{i=1,2} Q_{\bar{\chi}_i}^z(s', z') - \alpha \log \pi_\theta^z(z' \mid s') \right], \tag{35}$$

where $Q_{\bar{\chi}_i}^z$ are target critics. In the LIBERO configuration, the critic target omits the entropy term; the critic loss also includes Monte Carlo return regression with the weight in Table 4.

If replay contains no successful samples, the success-filtered actor update is skipped.

Evaluation modes. In the LIBERO-10 comparison, NSL methods use deterministic actor-mean evaluation, while the base-policy reference uses native Gaussian noise sampling. Stochastic training-rollout success is recorded separately. Learning curves are indexed by training epoch.

B.2 Real-Robot Protocol

Policy and execution. Each task uses a separately fine-tuned $\pi_{0.5}$ base policy, shared by the compared methods. Observations comprise primary-view and wrist-camera images, the task instruction, and a seven-dimensional proprioceptive vector containing end-effector position, Euler-angle orientation, and gripper state. Primary-view crops have resolution 224×224. The action representation

likewise contains six pose coordinates and a gripper command. Policies predict 50-step chunks and execute the first ten steps before replanning.

Table 6 summarizes the demonstration counts and SFT steps used to initialize each task's base policy.

Table 6: Task-specific $\pi_{0.5}$ SFT data and training steps. Bead placement uses 20 complete demonstration trajectories.

Task	Demonstrations	SFT steps
Cup stacking	25	30,000
Toy fishing	35	30,000
Sequential insertion	35	50,000
Bead placement	20	140,000

Task objectives. *Cup stacking* requires placing the medium cup inside the largest cup and then the smallest cup inside the medium cup. *Toy fishing* requires catching a red toy fish with a fishing rod among differently colored fish. *Sequential insertion* requires inserting four pieces into their matching slots in the right-to-left order numbered in Figure 4. *Bead placement* requires arranging 16 colored beads into the four colored blocks of the Microsoft logo.

Online collection. Adaptation alternates between trajectory collection and model updates. Action-AWR, RLT, and autonomous NSL-AWR use robot-collected experience without human corrective demonstrations during online adaptation. DAgger uses human trajectories, while UniSteer and NSL-AWR with human guidance combine model and human data. Figure 5 gives mean trajectory counts per round. Human counts are included in the corresponding totals; model counts equal total minus human counts.

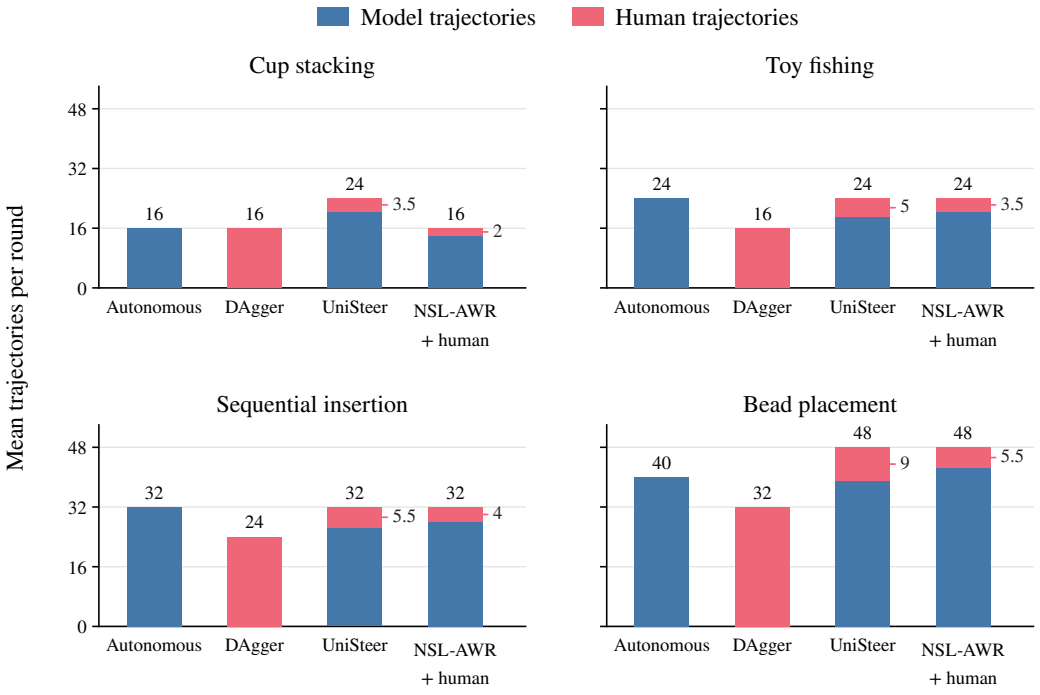


Figure 5: Mean trajectories collected per adaptation round. Bars separate model and human trajectories; labels above bars give totals, and side labels give human counts. Autonomous denotes the shared schedule for Action-AWR, RLT, and NSL-AWR.

Human-guided noise updates. For NSL-AWR with human guidance, we use the fixed-point inversion procedure of UniSteer (Lu et al., 2026) to recover noise corresponding to human action chunks under the frozen VLA. The recovered context–noise pairs are combined with robot-collected samples and used to update the noise actor with the AWR objective in Eq. (9).

Evaluation and timing. Each method is evaluated on 40 attempts per task. Task success rate is the fraction of successful attempts, and the reported average weights all four tasks equally. Training time is the measured cumulative time spent on model updates for each task, averaged over the four tasks.

B.3 Additional Ablation Curves

Figures 6 and 7 regroup the top and bottom rows of Figure 3 by algorithm.

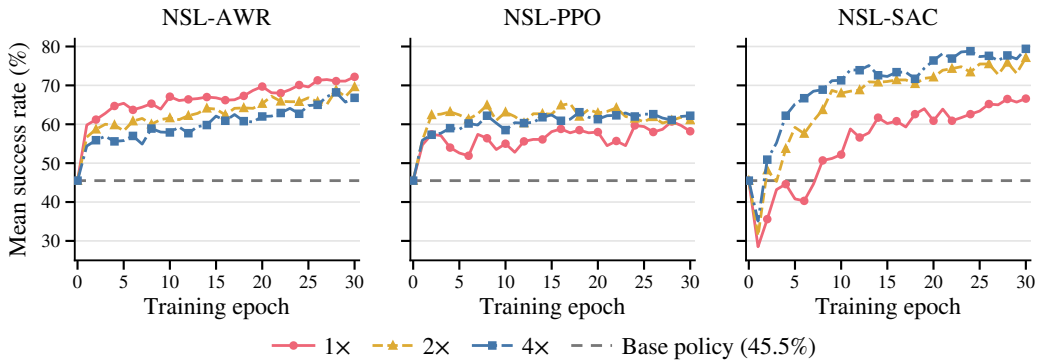


Figure 6: Update-budget comparisons within each algorithm, averaged equally over SmolVLA-Full and $\pi_{0.5}$-One. Dashed horizontal lines show the mean base-policy success rate (45.5%).

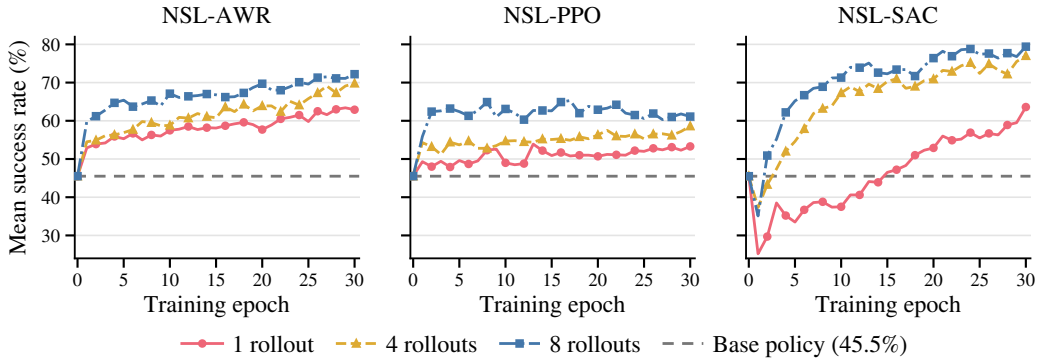


Figure 7: Rollout-count comparisons within each algorithm, averaged equally over SmolVLA-Full and $\pi_{0.5}$-One. Update ratios remain AWR $1\times$, PPO $2\times$, and SAC $4\times$. Dashed horizontal lines show the mean base-policy success rate (45.5%).

B.4 NSL-AWR Noise Evolution on Fixed Observations

Figures 8 and 9 track the four main LIBERO NSL-AWR 1× runs from epoch 0 to 60. Each run uses the same 500 fixed observations throughout the analysis, with 50 observations per task. The actors start with zero means and unit standard deviations.

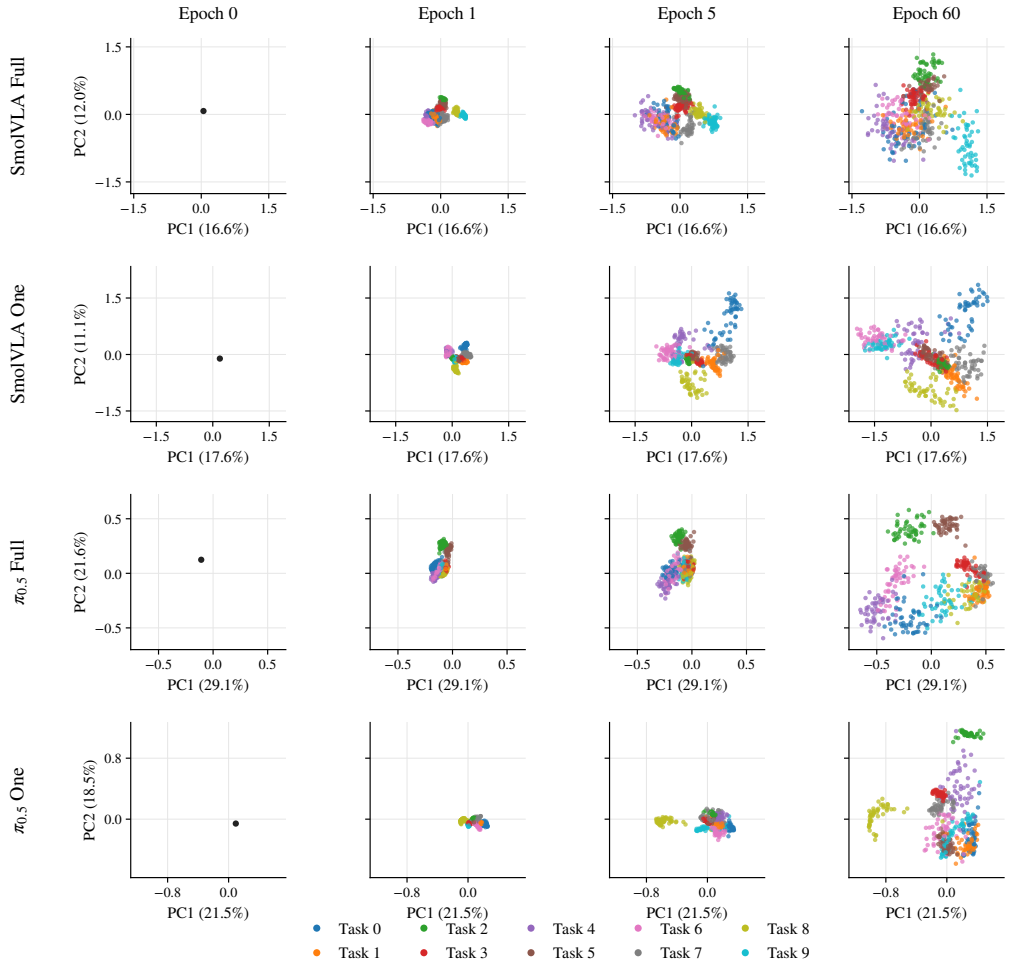


Figure 8: NSL-AWR actor-mean evolution on 500 fixed LIBERO observations. Rows show the four backbone–data settings; colors denote tasks. Each point is a projected 32-dimensional conditional actor mean. Each model uses one PCA basis fitted to all 30,500 means from 61 checkpoints, with fixed axis limits across its panels. Axis percentages give the explained variance over those checkpoints. The 500 initial means overlap and are shown as one black point.

Task-related mean structure appears after the first epoch, alongside differences between observations within each task. At epoch 60, variation between task centroids accounts for 52.8–79.9% of the total actor-mean variation in the original 32-dimensional space. SmolVLA exhibits larger mean shifts than $\pi_{0.5}$ and a larger reduction in average conditional standard deviation, ending near 0.80 compared with 0.95 for $\pi_{0.5}$. For either backbone, One has a larger final mean displacement than Full, while their average standard deviations are similar.

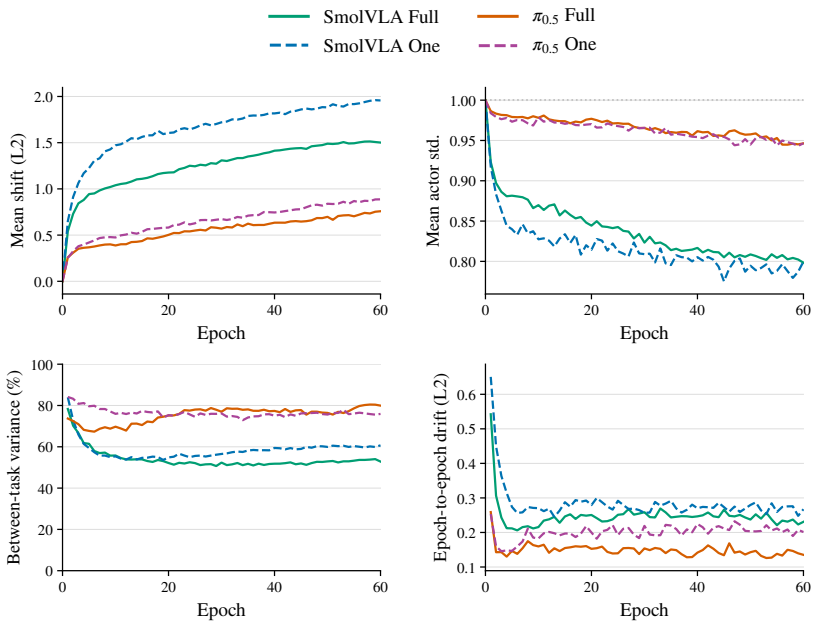


Figure 9: NSL-AWR output statistics on the same fixed observations. Mean shift and epoch-to-epoch drift average per-observation L2 distances of actor means relative to initialization and the preceding checkpoint, respectively. Actor standard deviation is averaged over observations and noise coordinates. Between-task variance is the fraction of total actor-mean squared variation explained by task centroids, computed in the original 32-dimensional space. Curves use all 61 checkpoints without smoothing; the initial drift and between-task fraction are undefined and omitted.

The outputs continue to change in later epochs. For $\pi_{0.5}$-One, the mean adjacent-epoch displacement over epochs 51–60 is 0.205, compared with 0.171 over epochs 1–5. Task-related structure thus coexists with continued refinement of observation-dependent outputs. These results illustrate how a shared noise actor develops contextual noise preferences while the action generator remains frozen.

B.5 Noise-Space SAC Diagnostics in Simulation

Figure 10 provides diagnostics for the noise-limit ablation on SmolVLA-Full discussed in Section 4.6.

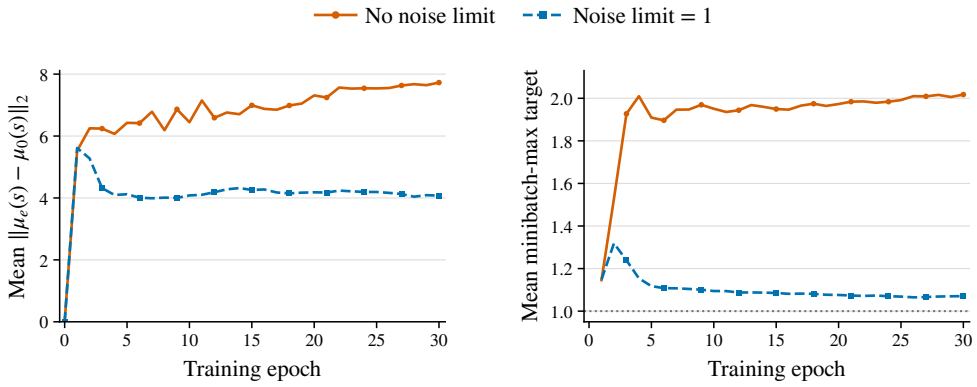


Figure 10: Effect of the noise limit on NSL-SAC with SmolVLA-Full. Left: mean L2 displacement of the underlying Gaussian mean from initialization over 500 fixed contexts per run, measured before any tanh transformation. Right: mean minibatch-maximum Bellman target, weighted by the number of critic updates. The dotted line marks the return upper bound of 1 under terminal binary rewards without entropy backup.

B.6 Noise-Space SAC Diagnostics on the Real Robot

We analyze an online real-robot bead-placement SAC experiment that collected 32 robot trajectories, yielding 2,789 replay transitions. The policy used 32-dimensional noise transformed by $5\tanh(\cdot)$ and executed ten action steps before replanning. Unintended table contact occurred during this experiment. Figure 11 shows 1,870 gradient updates from the recorded training segment; the horizontal axis indexes updates within online training.

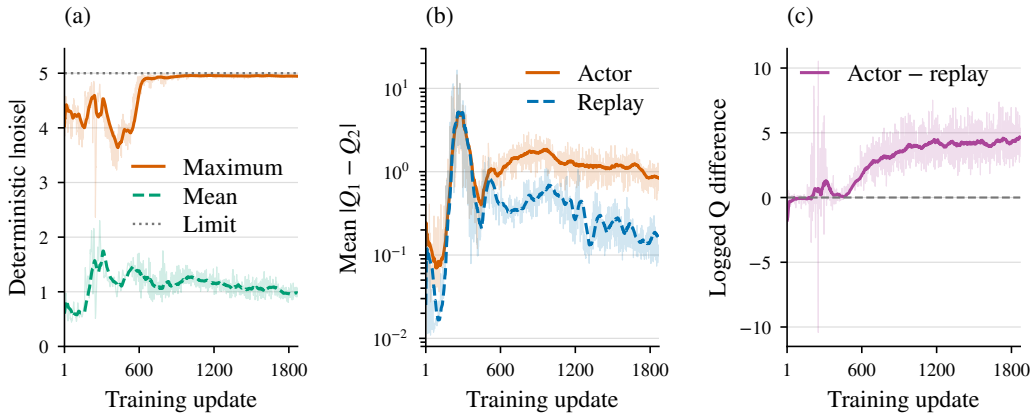


Figure 11: SAC training diagnostics for real-robot bead placement. (a) Mean and maximum absolute deterministic noise output $5\tanh(\mu)$ over minibatch contexts and coordinates; the dotted line marks the amplitude limit. (b) Mean absolute twin-critic disagreement for actor-proposed and replay noise. (c) Logged actor-proposal minus replay Q value, using the arithmetic mean of the two critics for both. Thin lines show minibatch statistics; thick lines show trailing 50-update means. The Q difference compares the separately logged actor and replay evaluations.

Some deterministic noise coordinates approach the amplitude limit during training. The final minibatch maximum absolute output is 4.96, while the mean absolute output increases from 0.60 to 1.04. This pattern reflects boundary proximity in a subset of outputs. Critic disagreement is also larger on actor proposals than on replay noise: over the last 200 updates, mean absolute twin-Q disagreement is 0.916 versus 0.151, a ratio of approximately 6.1. Over the same interval, the logged mean Q value for actor proposals exceeds that for replay by 4.43.

The actor proposes noises with higher predicted Q values and larger critic disagreement than replay noise, with some outputs approaching the amplitude boundary. The unintended table contact shows that limiting the noise amplitude alone does not ensure reliable decoded actions. These results motivate grounding actor updates in observed successful noise samples when robot data are limited.

C RoboTwin Experiments

C.1 Experimental Setup

We evaluate NSL-AWR, NSL-PPO, and NSL-SAC on eight RoboTwin (Chen et al., 2025b) tasks under Easy and Hard settings. For each task and difficulty, fixed manifests specify 50 training scenarios and 50 held-out evaluation scenarios with disjoint reset seeds, including the instruction and episode horizon for each scenario. The same requested manifests are used across methods.

Unlike our shared-policy LIBERO experiments, we train an independent noise actor for each task, without sharing RL parameters or replay across tasks. We reuse the NSL objectives and frozen-feature interface: the VLA remains frozen, and the noise actor conditions on pooled VLA features and proprioception. Replay balances successful and failed experience; AWR restricts actor regression to successful samples while fitting its value model to both outcomes. Noise actors start with zero Gaussian mean and log standard deviation -0.4; SAC uses a $\tanh$ noise limit of 1.0.

The main experiments run for 30 training epochs with eight rollouts and update multipliers of $4\times$, $2\times$, and $4\times$ for AWR, PPO, and SAC, respectively. Additional Easy experiments vary the update

multiplier over $1\times$, $2\times$, $4\times$, and $8\times$, and the rollout count over one, four, and eight. These ablations are compared at epoch 20. We denote the update multiplier by U and the rollout count by R.

All methods start from the same publicly released RoboTwin SmolVLA SFT checkpoint, obtained by fine-tuning SmolVLA on 27,500 RoboTwin demonstration trajectories. The mean success rate of the SFT reference is 31.50% on Easy and 32.75% on Hard.

C.2 Main Results

Table 7 compares the three methods against SFT. All three improve both settings, with NSL-SAC achieving the highest mean success rate. At epoch 30, it reaches 84.25% on Easy and 77.25% on Hard, improving over SFT by 52.75 and 44.50 percentage points. NSL-AWR and NSL-PPO also provide substantial gains, with their relative ranking depending on the setting.

Table 7: RoboTwin main results with $R = 8$. SR (%) is the equal-weight mean over all eight tasks at the same epoch. Δ is epoch-30 SR minus SFT, in percentage points. Best is the highest mean SR within epochs 1–30; parentheses give its first occurrence.

Setting	Method	SFT	Epoch 30	Δ	Best (epoch)
Easy	NSL-AWR $4\times$	31.50	73.50	$+42.00$	73.50 (30)
	NSL-PPO $2\times$	31.50	70.50	$+39.00$	73.50 (29)
	NSL-SAC $4\times$	31.50	**84.25**	$+52.75$	**84.25** (30)
Hard	NSL-AWR $4\times$	32.75	65.00	$+32.25$	67.00 (29)
	NSL-PPO $2\times$	32.75	69.25	$+36.50$	72.25 (27)
	NSL-SAC $4\times$	32.75	**77.25**	$+44.50$	**78.50** (26)

C.3 Per-Task Results

Table 8 reports individual task results. At epoch 30, every method exceeds SFT on all eight tasks in both settings. Improvements extend to tasks with low initial success: NSL-SAC raises Easy `move_can_pot` from 4% to 86% and Hard `place_phone_stand` from 2% to 62%. The best method varies by task: NSL-AWR leads on Easy dual-bottle picking and microphone handover, while NSL-PPO leads on Hard dual-bottle picking. The task `place_a2b_left` remains the weakest across all six main runs.

Table 8: Per-task SR (%) on RoboTwin Easy and Hard at epoch 30. All runs use $R = 8$, with update multipliers of $4\times$ for NSL-AWR, $2\times$ for NSL-PPO, and $4\times$ for NSL-SAC. AWR, PPO, and SAC columns denote the NSL methods. Bold marks the best entry within each difficulty block for each row, including ties.

	Easy				Hard			
Task	SFT	AWR	PPO	SAC	SFT	AWR	PPO	SAC
`handover_block`	10.0	62.0	62.0	**96.0**	10.0	40.0	56.0	**86.0**
`move_can_pot`	4.0	66.0	60.0	**86.0**	8.0	60.0	58.0	**78.0**
`place_a2b_left`	20.0	38.0	34.0	**40.0**	20.0	**34.0**	**34.0**	26.0
`place_phone_stand`	12.0	64.0	62.0	**76.0**	2.0	50.0	**62.0**	**62.0**
`pick_dual_bottles`	58.0	**96.0**	88.0	90.0	60.0	88.0	**94.0**	86.0
`handover_mic`	66.0	**100.0**	96.0	98.0	58.0	94.0	90.0	**98.0**
`place_empty_cup`	56.0	94.0	96.0	**100.0**	46.0	80.0	78.0	**92.0**
`beat_block_hammer`	26.0	68.0	66.0	**88.0**	58.0	74.0	82.0	**90.0**
Mean	31.50	73.50	70.50	**84.25**	32.75	65.00	69.25	**77.25**

C.4 Analysis of Update Budget

Table 9 shows that the preferred update budget depends on the optimizer. NSL-SAC improves by 17.00 percentage points when increasing U from one to four, but a further increase to eight reduces its epoch-20 SR. NSL-PPO performs best at $2\times$ and deteriorates with larger budgets; its $8\times$ run declines from 60.75% at epoch 4 to 50.50% at epoch 20. NSL-AWR shows comparatively stable performance across the tested update budgets, achieving its best result at $2\times$ and maintaining similar SR at $4\times$ and $8\times$. A modest update budget is therefore sufficient for strong NSL-AWR performance.

Table 9: Update-budget ablation on RoboTwin Easy. Entries are eight-task mean SR (%) at epoch 20 with $R = 8$. Bold indicates the best multiplier for each method.

Method	$U = 1\times$	$U = 2\times$	$U = 4\times$	$U = 8\times$
NSL-AWR	63.75	**70.75**	68.75	68.25
NSL-PPO	69.25	**72.25**	65.50	50.50
NSL-SAC	63.75	71.50	**80.75**	77.00

C.5 Analysis of Rollout Count

Increasing the rollout count improves all three methods (Table 10). From one to eight rollouts, epoch-20 SR increases by 7.75 percentage points for NSL-AWR, 16.75 for NSL-PPO, and 20.50 for NSL-SAC. With one rollout per scenario, AWR and SAC finish at similar performance and both exceed PPO. Thus, SAC's advantage is strongly reduced under limited collection, while SAC benefits more from additional collection, gaining 9.75 percentage points from four to eight rollouts. This pattern is consistent with value-guided optimization benefiting from richer experience.

Table 10: Rollout-count ablation on RoboTwin Easy at epoch 20. Entries are eight-task mean SR (%). Update multipliers remain at $4\times$, $2\times$, and $4\times$ for AWR, PPO, and SAC, respectively. Δ is the increase from $R = 1$ to $R = 8$, in percentage points.

Method	$R = 1$	$R = 4$	$R = 8$	Δ
NSL-AWR	61.00	67.50	**68.75**	$+7.75$
NSL-PPO	55.50	68.00	**72.25**	$+16.75$
NSL-SAC	60.25	71.00	**80.75**	$+20.50$