Reflective VLA: In-Context Action Consequences Make VLAs Generalize

Qing Lian[1] **Kent Yu**[1,2] **Lei Zhang**[1,2]
[1]Futian Lab [2]IDEA

Abstract

Most vision-language-action (VLA) models are reactive: they predict the next action from the current instruction and observation, implicitly assuming that the current observation fully specifies the action-relevant state. In embodied control, however, embodiment-specific factors such as camera-to-robot geometry, robot calibration, or systematic actuation bias are often hard to identify from a single observation. As a result, reactive policies cannot reliably disambiguate these factors in general, overfitting to training environments and generalizing poorly at deployment. We propose Reflective VLA, which conditions each decision on a context of observation–action–consequence triplets. Each triplet records not only what the robot observed and executed, but also how the scene changed afterward, exposing the deployment-specific mapping from actions to observed effects. Architecturally, Reflective VLA routes all observation modalities through the VLM under shared attention, so the action expert reasons directly over past triplets and the current observation. A block-causal mask enables parallel multi-frame training without leakage and supports KV-cached real-time inference. On standard LIBERO and SimplerEnv-Bridge, Reflective VLA preserves strong in-distribution performance. Under distribution shift on LIBERO-Plus and the harder LIBERO-Plus-Hard, it improves average success rate by 4.6 and 8.7 percentage points over a matched reactive baseline. Ablations with a matched history-only baseline further show that action consequences—rather than additional context length alone—are the key to cross-environment generalization.

1 Introduction

Vision-language-action (VLA) systems build on pretrained vision-language backbones Bai et al. [2025a,b], Beyer et al. [2024] and fine-tune on large-scale robot demonstrations Open X-Embodiment Collaboration et al. [2024], AgiBot-World-Contributors et al. [2025] to produce language-conditioned control policies Zitkovich et al. [2023], Kim et al. [2024], Black et al. [2025], Li et al. [2024a], NVIDIA et al. [2025]. By unifying scene understanding, instruction following, and low-level control in one model, they have substantially broadened the range of manipulation tasks a generalist policy can perform. Yet despite training on large and diverse datasets, current VLAs still generalize poorly to deployment environments unseen during training Fei et al. [2025], Zhang et al. [2025a].

Cross-environment generalization in embodied control often requires inferring embodiment-specific factors that are hard to identify from single observations, such as camera geometry, robot calibration, and systematic actuation bias. In contrast, the *interaction context*—an observation, an executed action, and the resulting observation—exposes these factors through how actions translate into observed changes: a commanded motion reveals camera extrinsics through its pixel displacement and calibration offsets through its pose residual. Existing VLAs condition on a single frame, omitting this evidence, so the policy must instead memorize the embodiments seen during training. Recent temporal-context and memory-based VLAs Liu et al. [2025], Shi et al. [2026] improve state tracking,

Preprint.

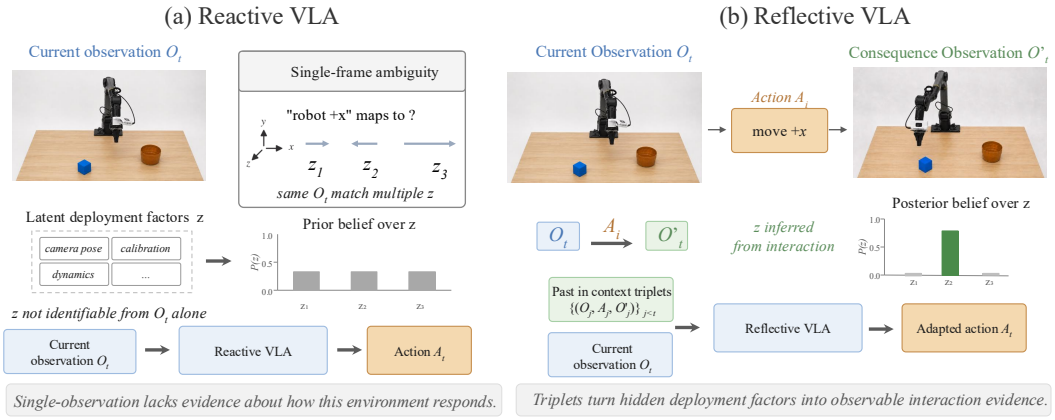


Figure 1: **From reactive to reflective control. (a) Reactive VLA.** Identifying embodiment-specific latent factors z—camera pose, calibration, etc.—from a single observation O_t is ill-posed: many z are consistent with the same frame, so A_t overfits training embodiments. **(b) Reflective VLA.** Conditioning on past causal triplets $\{(O_j, A_j, O'_j)\}_{j<t}$, where O'_j is the action-aligned observation after A_j, rules out deployments inconsistent with the evidence, sharpening $P(z \mid \text{context})$ and yielding an A_t adapted to the current embodiment.

but lack explicit action–consequence binding, which is critical for identifying deployment-specific sensing and control factors.

We therefore cast cross-environment generalization in VLAs as an in-context learning (ICL) problem over causal interaction triplets. Given a small prompt of such triplets, the policy can infer the current deployment online from interaction feedback, analogous to how large language models infer task structure from few-shot demonstrations Brown et al. [2020], Xie et al. [2022]. Instantiating this in a dual-system VLA raises two challenges. First, the prompt interleaves heterogeneous modalities—images, proprioception, and continuous action chunks—which must be packed into a causal sequence without bottlenecking the action decoder. Second, dual-system VLAs must propagate context through both the VLM prefix and the action expert; naive training repeats forward passes across target frames, while naive inference recomputes the history prefix at every step, making real-time control costly.

We address these challenges with **Reflective VLA**, an ICL framework for dual-system VLAs. The current observation is augmented with a small set of past triplets, each consisting of an observation, an executed action chunk, and the resulting observation. To pack heterogeneous modalities into a single causal sequence without bottlenecking the action decoder, all modalities share the VLM token space and a continuous action expert attends densely to the full prompt under shared attention. To make ICL training tractable, a block-causal mask supervises all K context frames in a single forward pass instead of K separate ones. The same causal structure supports KV-cached inference, so the extended context does not compromise real-time control at deployment.

We evaluate Reflective VLA on standard LIBERO and SimplerEnv, as well as on LIBERO-Plus Fei et al. [2025] and our extension, LIBERO-Plus-Hard, which target deployment shifts in sensing, embodiment, and layout. Reflective VLA preserves strong standard-benchmark performance, improving over a matched reactive baseline by 1.1 and 5.3 points on LIBERO and SimplerEnv, respectively. Under deployment shifts, it improves held-out generalization by 4.6 and 8.7 points on LIBERO-Plus and LIBERO-Plus-Hard, respectively, without test-time fine-tuning. Ablations show that these gains come from observation–action–consequence evidence rather than longer context alone.

In summary, our main contributions are:

1. We identify observation–action–consequence interaction context as a key signal for cross-environment VLA generalization, and formulate it as in-context learning over causal triplets.
2. We introduce **Reflective VLA**, a dual-system VLA that places multimodal observations and historical action consequences in a shared VLM token space, with block-causal training and KV-cached real-time inference.
3. Across LIBERO, SimplerEnv, LIBERO-Plus, and LIBERO-Plus-Hard, Reflective VLA improves standard-benchmark performance and held-out deployment generalization over

matched reactive baselines; history-only ablations confirm that aligned consequence observations are the critical ingredient.

2 Related Work

Vision-language-action models. Recent generalist policies build on pretrained vision-language backbones and train on large-scale robot datasets to produce language-conditioned control Reed et al. [2022], Brohan et al. [2023], Zitkovich et al. [2023], Kim et al. [2024], Team et al. [2024]. Architecturally, recent VLAs follow two main designs. One line casts control as autoregressive prediction over discretized or tokenized actions Zitkovich et al. [2023], Kim et al. [2024, 2025], Pertsch et al. [2025]. A more recent line decouples high-level reasoning from low-level control by pairing the VLM backbone with a dedicated continuous action expert based on diffusion or flow matching—a *dual-system* design that preserves action fidelity at high control rates Black et al. [2025], Li et al. [2024a], Wu et al. [2026], Zheng et al. [2026], NVIDIA et al. [2025]. We adopt this dual-system design and take an orthogonal direction: rather than scaling data or refining the action representation, we equip a fixed VLA with in-context interaction evidence as a new axis for cross-environment generalization, without any test-time updates.

Temporal context and memory for robot control. Temporal context is a natural response to partial observability in robot control. Imitation policies such as ACT, Diffusion Policy, and RDT use short observation histories and action chunks to stabilize visuomotor control Zhao et al. [2023], Chi et al. [2023], Liu et al. [2025]; VLA systems such as RoboFlamingo, 4D-VLA, and MemoryVLA further incorporate visual history, spatiotemporal representations, or explicit memory for language-conditioned manipulation Li et al. [2024b], Zhang et al. [2025b], Shi et al. [2026]. These methods improve state estimation, task progress tracking, and robustness to transient perceptual ambiguity. However, temporal history alone is not equivalent to interaction feedback: existing histories do not explicitly bind an executed action chunk to its aligned consequence. Reflective VLA therefore distinguishes temporal context from action-conditioned consequence context, and tests this distinction with history-only ablations that remove consequence observations from the same historical milestones.

In-context adaptation for sequential decision making. In-context learning provides a complementary view of adaptation: a sequence model can infer the relevant task or environment from examples in its context Brown et al. [2020], Xie et al. [2022]. This idea connects to meta-RL Duan et al. [2016], Finn et al. [2017], Rakelly et al. [2019] and to transformer policies that adapt behavior from contextual experience at test time, including Decision Transformer, Prompt-DT, and algorithm distillation Chen et al. [2021], Xu et al. [2022], Laskin et al. [2023]. Reflective VLA brings this viewpoint to embodied VLA control, but uses a different adaptation signal: structured multimodal observation–action–consequence triplets, without rewards, textual self-reflection, a separate system-identification module, or test-time parameter updates.

3 Method

Reflective VLA extends a reactive dual-system VLA with an in-context interface over observation–action–consequence triplets. We first define the reactive formulation, then describe how triplets are constructed and packed, how the model is trained on them in parallel using a block-causal mask, and how they are reused during online inference.

3.1 Preliminary: Standard Reactive VLA Formulation

At control step t, let $\mathcal{L}$ denote the language instruction, $\mathcal{O}_t$ the current multimodal observation, and $A_t = [a_t, \ldots, a_{t+C-1}]$ an action chunk of horizon C. The observation may include third-person images, wrist images, and proprioceptive states. A standard reactive VLA predicts the next action chunk from only the current instruction and observation:

$$A_t \sim \pi_\theta(\cdot \mid \mathcal{L}, \mathcal{O}_t). \tag{1}$$

This interface covers recent dual-system VLAs that pair a VLM prefix with a continuous diffusion or flow-matching action expert. Although their action generation objectives differ, they share the same limitation for our purpose: the action expert conditions on the current observation, but not on past executed actions and their observed consequences.

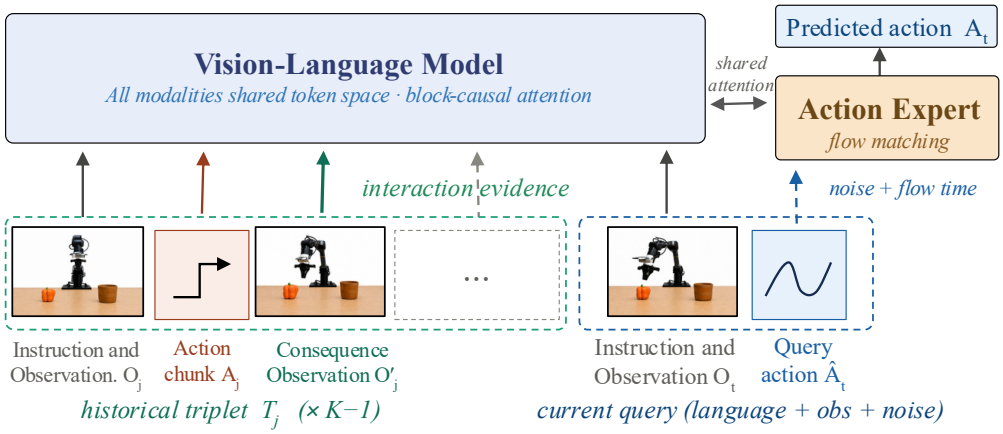


Figure 2: **Reflective VLA: observation–action–consequence in-context learning.** Past triplets $\{(O_j, A_j, O'_j)\}_{j=1}^{K-1}$ and the current observation O_t share a single VLM token sequence; a flow-matching action expert attends to this prefix and denoises $\hat{A}_t$ into the predicted chunk A_t. The aligned consequence O'_j carries the *interaction evidence* that exposes environment factors.

3.2 From Reactive to In-Context Generalization

A reactive VLA learns $\pi(A_t \mid \mathcal{L}, \mathcal{O}_t)$, predicting the next action from the current instruction and observation. Different deployments may share task semantics but differ in camera-to-robot geometry, robot calibration, or systematic actuation bias. We summarize these embodiment-specific factors as an environmental latent variable z, on which the appropriate action depends. Inferring z from a single observation $\mathcal{O}_t$ is ill-posed: the mapping from z to $\mathcal{O}_t$ is many-to-one, leaving the policy unable to disambiguate the current sensing and control conditions.

This missing evidence can be recovered by interaction feedback: when the robot executes an action and observes the resulting change, the observation–action–consequence relation reveals how the current deployment responds to commands. Formally, an adaptive policy can be written as a marginal over the posterior on z:

$$\pi(A_t \mid \mathcal{L}, \mathcal{O}_t, \mathcal{H}) = \int \pi(A_t \mid \mathcal{L}, \mathcal{O}_t, z)\, P(z \mid \mathcal{H})\, dz, \tag{2}$$

where $\mathcal{H}$ is the interaction context accumulated so far. We do not explicitly estimate z or compute this integral; the formulation serves only as a motivating abstraction, since transformers can implicitly approximate such posterior updates via in-context inference Xie et al. [2022].

The key requirement is that $\mathcal{H}$ is diagnostic of z. Under an idealized Markov view, if $\mathcal{H}$ consists of interaction triplets, the posterior factorizes as

$$P(z \mid \mathcal{H}) \propto P(z) \prod_i P_{\text{env}}(\mathcal{O}'_i \mid \mathcal{O}_i, A_i, z), \tag{3}$$

where $\mathcal{O}'_i$ is the observation after executing action chunk A_i. The likelihood depends explicitly on how the environment responds to executed actions, so a context containing only $(\mathcal{O}_i, A_i)$ pairs cannot expose the response term and provides limited evidence for identifying deployment-specific factors.

Reflective VLA therefore conditions each decision on interaction context with explicit consequences:

$$A_t \sim \pi_\theta\big(A_t \mid \mathcal{L}, \mathcal{H}, \mathcal{O}_t\big), \quad \mathcal{H} = \big\{(\mathcal{O}_i, A_i, \mathcal{O}'_i)\big\}_{i=1}^{K-1}, \tag{4}$$

where $\mathcal{H}$ is the observation–action–consequence context.

3.3 Reflective VLA

Observation–action–consequence context. Reflective VLA represents recent interaction history as structured observation–action–consequence triplets. Each triplet stores the observation before an action chunk, the action chunk executed from that observation, and the resulting observation after the chunk completes. Since the policy predicts C-step action chunks, we define the consequence observation as $\mathcal{O}'_i = \mathcal{O}_{i+C}$ rather than the immediate next frame $\mathcal{O}_{i+1}$. This action-aligned consequence

better captures the visible effect of the executed action, such as end-effector displacement, residual control error and etc.

At training time, A_i is the demonstration action chunk and $\mathcal{O}'_i$ the observation after the chunk horizon; at deployment, A_i is the chunk actually executed and $\mathcal{O}'_i$ the subsequent observation collected from the environment. Thus, each stored triplet represents a realized interaction rather than only a planned transition. We embed each previous action chunk into eight tokens in the VLM token space via a learned projection g_A, and write the resulting prefix tokens simply as A_i. The i-th context element is then $T_i = (\mathcal{L}, \tau_i, \mathcal{O}_i, A_i, \mathcal{O}'_i)$, where the language instruction $\mathcal{L}$ is re-inserted at the start of every triplet so that each unit is a self-contained $(\mathcal{L}, \mathcal{O}, A, \mathcal{O}')$ block, matching the per-triplet structure shown in Figure 3. We refer to these as causal triplets because the action token links a pre-action observation to its observed post-action consequence.

Multimodal prompt construction. Given historical milestones $m_1, \ldots, m_{K-1}$ and the current query timestep t, Reflective VLA packs past triplets and the current observation—each prefixed by the language instruction $\mathcal{L}$—into a single multimodal sequence:

$$X_{\text{in}} = \Big[T_{m_1}, \ldots, T_{m_{K-1}},\ (\mathcal{L}, \tau_t, \mathcal{O}_t)\Big]. \tag{5}$$

The historical triplets provide evidence about how the current deployment responds to actions, while the final $(\mathcal{L}, \tau_t, \mathcal{O}_t)$ block serves as the query for predicting the next action chunk. The current target action is not inserted into the prefix; it is generated by the continuous action expert.

Shared-attention dual-system architecture. For the action decoder to use interaction context, all observation modalities must be visible to it. Some dual-system VLAs route only part of the observation through the VLM backbone or condition the action module on a compressed prefix NVIDIA et al. [2025], Zheng et al. [2026], which can bottleneck reasoning over (O, A, O') structure. Reflective VLA therefore routes all observation modalities—third-person images, wrist images, and proprioception—into a shared VLM token sequence. Images are encoded by the visual backbone, while proprioceptive states and previous action chunks are projected into the token space with a learned non-linear head. A continuous flow-matching action expert is attached as a suffix that shares attention with the VLM prefix at every layer, following recent MoT-style VLA designs Liang et al. [2025], Black et al. [2025], Wu et al. [2026]. The action expert can thus attend directly to prior observations, prior actions, observed consequences, temporal indices, and the current observation when generating A_t.

Block-causal training. In-context conditioning lengthens each training sequence by a factor of K, so supervising each context position with an independent forward pass would scale training cost as $\mathcal{O}(K)$. We instead borrow the packed-sequence idea from LM training and supervise all K sampled frames jointly under a *block-causal* attention mask, which lets every sampled frame act as both context for later targets and as a prediction target itself within a single forward pass. Because the same mask supervises positions with $0, 1, \ldots, K-1$ preceding triplets, training naturally covers the full range of context lengths the model will encounter at deployment.

Sampling and packing. We sample K ordered frames $t_1 < \cdots < t_K$ from a trajectory and pack them into a single sequence: the first $K-1$ form the historical triplets and the K-th is the current query, but the block-causal mask supervises all K frames jointly. As reflected in T_i, $\mathcal{L}$ is re-inserted at the start of every triplet (and the query block) so each unit is a self-contained $(\mathcal{L}, \mathcal{O}, A, \mathcal{O}')$ structure, and special tokens demarcate triplet boundaries so the action expert can unambiguously parse where each unit begins and ends. Because adjacent action chunks are temporally smooth, a fixed inter-frame stride invites the policy to extrapolate A_{t_k} from past actions rather than learn from observed consequences; we therefore randomize the stride within a bounded range during training to break this shortcut.

Mask and objective. Each target frame t_k is realized by a query slot $\hat{A}_{t_k}$ in the suffix flow-matching expert; Figure 3 visualizes the resulting attention pattern as a row in the mask. The query attends only to

$$\mathcal{V}_{t_k} = \{T_{t_j} : j < k\} \cup \{(\mathcal{L}, \tau_{t_k}, \mathcal{O}_{t_k})\}, \tag{6}$$

i.e., all completed prior triplets (each carrying its own copy of $\mathcal{L}$) together with the current language-prefixed observation. It is masked from the prefix action token A_{t_k} and consequence $\mathcal{O}'_{t_k}$ within its own triplet—both observed only after A_{t_k} is executed—and from every subsequent triplet $\{T_{t_j} : j >$

Table 1: **In-distribution evaluation on LIBERO and SimplerEnv-Bridge.** Success rates (%) across the four standard LIBERO task suites and the SimplerEnv-Bridge benchmark. † denotes our reproduced reactive baseline $\pi_{0.5}$. "–" indicates the result is not provided.

	LIBERO					SimplerEnv-Bridge				
Method	Spatial	Object	Goal	Long	Avg	Spoon	Carrot	Cube	Eggplant	Avg
OpenVLA Kim et al. [2024]	84.7	88.4	79.2	53.7	75.9	4.2	0.0	8.3	45.8	14.6
CoT-VLA Zhao et al. [2025]	87.5	91.6	87.6	69.0	81.1	–	–	–	–	–
4D-VLA Zhang et al. [2025b]	93.8	92.8	95.6	86.5	92.2	–	–	–	–	–
ThinkAct Huang et al. [2025]	–	–	–	–	–	37.5	8.7	58.3	70.8	43.8
CogACT Li et al. [2024a]	97.2	98.0	90.2	88.8	93.2	71.7	50.8	15.0	67.5	51.3
InternVLA-M1 Intern Robotics [2025]	98.0	**99.0**	93.8	92.6	95.9	87.5	67.9	31.3	**100.0**	71.7
π_0 Black et al. [2025]	96.8	98.8	95.8	85.2	94.2	–	–	–	–	–
MemoryVLA Shi et al. [2026]	98.4	98.4	96.4	93.4	96.5	75.0	75.0	37.5	**100.0**	71.9
GR00T N1.5 NVIDIA et al. [2025]	–	–	–	–	–	82.0	72.0	54.0	63.0	67.8
$\pi_{0.5}$ Physical Intelligence [2025]	**98.8**	98.2	98.0	92.4	96.9	–	–	–	–	–
Reactive baseline $\pi^{\dagger}_{0.5}$	97.5	98.2	97.8	94.0	96.9	91.7	79.2	70.8	50.0	72.9
Reflective VLA (ours)	98.4	**99.0**	**98.2**	**94.6**	**98.0**	**95.8**	**83.3**	79.2	54.2	**78.2**

Table 2: **Robustness under perturbations on LIBERO-Plus and LIBERO-Plus-Hard.** Success rates (%) on LIBERO-Plus (7 standard perturbation categories) and LIBERO-Plus-Hard (2 additional harder shifts: Multi-camera shift (Camera†) and Robot calibration shift (Rob. Calib†)).

	LIBERO-Plus								LIBERO-Plus-Hard		
Method	Camera	Robot	Lang	Light	Bg	Noise	Layout	Avg	Camera†	Rob. Calib†	Avg
UniVLA Bu et al. [2025]	1.8	46.2	69.6	69.0	81.0	21.2	31.9	42.9	–	–	–
π_0 Black et al. [2025]	13.8	6.0	58.8	85.0	81.4	79.0	68.9	53.6	–	–	–
OpenVLA-OFT Fei et al. [2025]	92.8	30.3	85.8	94.9	93.9	89.3	**77.6**	79.6	75.2	43.1	59.2
MemoryVLA Shi et al. [2026]	93.1	42.1	84.2	95.1	92.7	89.1	77.5	82.0	76.1	49.2	62.7
Reactive baseline $\pi^{\dagger}_{0.5}$	90.0	50.0	94.9	92.0	85.8	90.2	75.0	82.6	75.0	50.2	62.6
Reflective VLA (ours)	**96.0**	**60.2**	**95.0**	**96.1**	**94.1**	**91.3**	**77.6**	**87.2**	**80.1**	**62.5**	**71.3**

$k\}$; sibling query slots $\{\hat{A}_{t_j}\}_{j\neq k}$ are also mutually masked, so each prediction depends solely on prefix evidence. The training objective sums the flow-matching action loss over all valid targets:

$$\mathcal{L}_{\text{train}} = \sum_{k=1}^{K} \mathcal{L}_{\text{act}}\big(A_{t_k}; \mathcal{V}_{t_k}\big), \tag{7}$$

where $\mathcal{L}_{\text{act}}$ is the same flow-matching objective used by the base VLA. This yields dense multi-frame supervision in one forward pass while ensuring that each prediction is conditioned only on past completed interactions and the current observation. The first target t_1 has no preceding triplets and provides reactive supervision, while $t_2, \ldots, t_K$ provide ICL supervision with progressively longer context, as the staircase pattern along the diagonal of Figure 3 shows.

At deployment, Reflective VLA keeps a rolling buffer of the most recent $K-1$ triplets. Unlike training, where ground-truth action chunks populate the historical context, at inference each triplet stores the policy's own predicted chunk $\hat{A}_t$ together with the observation $\mathcal{O}_{t+C}$ actually reached after executing it—so the in-context prefix reflects the realized rollout rather than a teacher trajectory. At each step, only the new observation (or triplet, after execution) is encoded; VLM-side keys and values for past triplets are cached once and reused, keeping the per-step inference cost roughly constant.

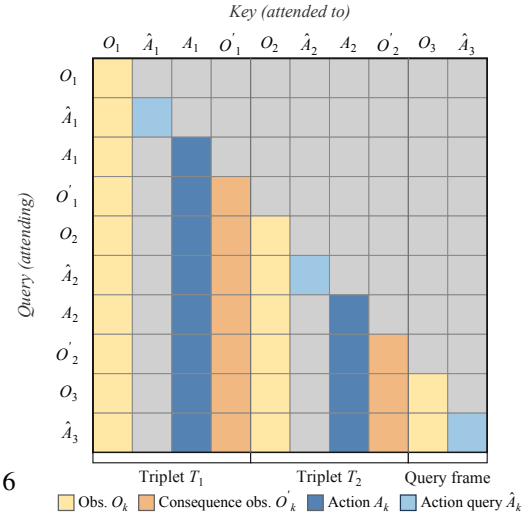


Figure 3: **Block-causal mask.** Each query $\hat{A}_k$ attends to $\mathcal{L}$, prior triplets $T_{<k}$, and $\mathcal{O}_k$, while its

Table 3: **Ablations on the interaction context.** Success rates (%) on Camera, Camera†, and Rob. Calib†; Avg denotes the mean. (a) varies what each context element contains, with K fixed; (b) varies the number of context elements K with the full (O, A, O') structure. $K{=}1$ corresponds to the reactive baseline without historical triplets.

(a) Context composition (fixed K).

Context	Camera	Camera†	Rob. Calib†	Avg
Reactive (no history)	90.0	75.0	50.2	71.7
O	88.8	74.2	50.0	71.0
O, A	89.1	75.0	54.4	72.8
O, A, O'	**96.0**	**80.1**	**62.5**	**79.5**

(b) Context length (full (O, A, O')).

K	Camera	Camera†	Rob. Calib†	Avg
1	90.0	75.0	50.2	71.7
2	93.8	77.2	57.0	76.7
4	96.1	79.2	60.4	78.6
8	**96.0**	**80.1**	**62.5**	**79.5**

4 Experiments

We evaluate Reflective VLA along three axes. First, we test whether adding interaction context preserves strong in-distribution performance on standard manipulation benchmarks. Second, we evaluate robustness under perturbations that change visual appearance, sensing, or embodiment-specific properties. Third, we ablate the designed components to illustrate the contribution of the observation–action–consequence structure rather than from longer context alone.

4.1 Experimental Setup

Simulation benchmarks. We evaluate on four simulation settings: *(i)* LIBERO Liu et al. [2023], the standard suite of 40 language-conditioned manipulation tasks across *Spatial*, *Object*, *Goal*, and *Long*, with a fixed camera and embodiment, used as our nominal in-distribution testbed; *(ii)* SimplerEnv-Bridge Li et al. [2024c], which evaluates VLAs trained on the BridgeDataV2 Walke et al. [2023] in the ManiSkill2 Gu et al. [2023] simulator, providing a real-to-sim transfer setting; *(iii)* LIBERO-Plus Fei et al. [2025], which extends LIBERO with seven perturbation categories spanning sensing, language, and robot–environment configuration; and *(iv)* LIBERO-Plus-Hard, our diagnostic extension targeting shifts in the action-to-observation mapping.

A context-diagnostic benchmark. While LIBERO-Plus provides broad perturbation coverage, not all of its categories are equally diagnostic of interaction-conditioned adaptation: shifts in language, or background can largely be absorbed by invariances in the pretrained VLM backbones. We therefore introduce LIBERO-Plus-Hard with two perturbations designed so that single-frame evidence is insufficient and the action-to-observation mapping must be inferred from interaction:

- **Multi-camera shift** (Camera† in Table 2). We jointly perturb the extrinsics of all camera views (third-person and wrist), so no single view preserves the nominal calibration; the camera-to-robot geometry must be recovered from the pixel-space displacement induced by past action chunks.
- **Robot calibration shift** (Rob. Calib† in Table 2). We inject an episode-level systematic offset between commanded and achieved end-effector motion, simulating calibration error, actuation bias, or mechanical backlash; this offset is unobservable from a static frame but recoverable from the residual between commanded actions and their consequences.

Baselines. Our primary baseline, denoted $\pi^\dagger_{0.5}$, is a reproduced *reactive* $\pi_{0.5}$ that uses the same backbone, training data, and network parameters as Reflective VLA but with context length $K{=}1$, conditioning each prediction only on the current instruction and observation. Where available and protocol-compatible, we additionally report published results for OpenVLA Kim et al. [2024], OpenVLA-OFT Kim et al. [2025], UniVLA Bu et al. [2025], CoT-VLA Zhao et al. [2025], 4D-VLA Zhang et al. [2025b], ThinkAct Huang et al. [2025], CogACT Li et al. [2024a], InternVLA-M1 Intern Robotics [2025], π_0 Black et al. [2025], $\pi_{0.5}$ Physical Intelligence [2025], MemoryVLA Shi et al. [2026], and GR00T N1.5 NVIDIA et al. [2025].

Implementation details. Following $\pi_{0.5}$ Physical Intelligence [2025], Reflective VLA adopts a Mixture-of-Transformers Liang et al. [2025] dual-system design, pairing a pretrained VLM prefix

with a flow-matching continuous action expert as the suffix under shared attention. Unless otherwise stated, we use action chunk size $C{=}10$, context length $K{=}8$, corresponding to seven historical triplets and one query observation, and bounded randomized milestone sampling. Additional architectural details, hyperparameters, and training settings are provided in Sections A and C.1.

4.2 Main Results

In-distribution performance. Table 1 shows that Reflective VLA preserves strong in-distribution performance on both LIBERO and SimplerEnv-Bridge. On LIBERO it reaches 98.0% average success, achieving state-of-the-art—ahead of the published $\pi_{0.5}$ (96.9%) and memory-based MemoryVLA (96.5%)—and outperforming the matched reactive $\pi^{\dagger}_{0.5}$ baseline by 1.1 points. On SimplerEnv-Bridge, it reaches 78.2% average success, again achieving state-of-the-art and improving over the reactive baseline by 5.3 points; since BridgeData V2 Walke et al. [2023] spans diverse scenes, embodiments, and viewpoints, this gain reflects the model's ability to exploit in-context interaction evidence under embodiment-level variability already present in the training distribution. Performance on the *Long* suite is also slightly improved (94.0% $\rightarrow$ 94.6%), indicating that interaction context does not hurt temporally extended manipulation. Together, these results show that adding structured interaction context does not compromise the base policy's nominal task-solving ability.

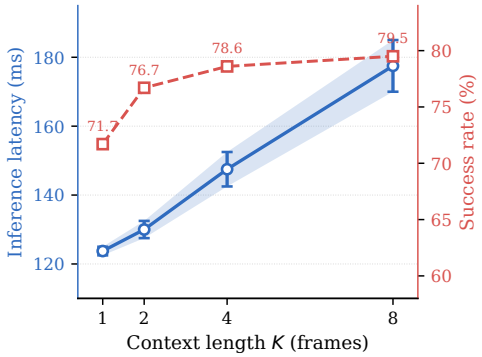
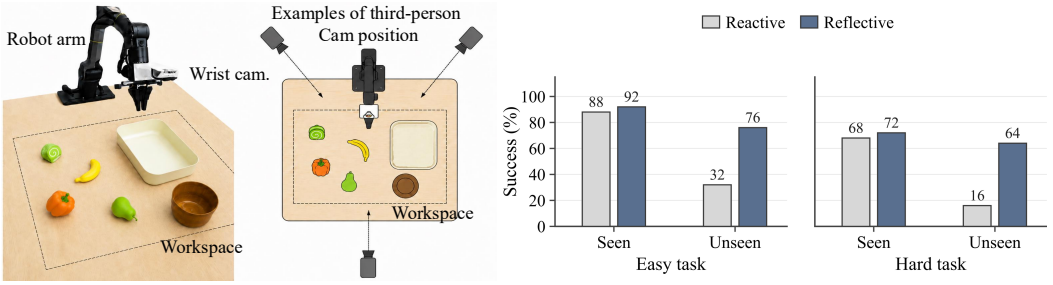


Figure 4: Latency–accuracy trade-off across context length K on the perturbation subset.

Robustness under perturbations. Table 2 evaluates robustness on the seven standard LIBERO-Plus perturbation categories and two harder shifts in LIBERO-Plus-Hard; MemoryVLA is reproduced on the same training data as ours for a fair comparison. On LIBERO-Plus, Reflective VLA achieves 87.2% average success, outperforming the matched reactive baseline (82.6%) and strong prior methods such as OpenVLA-OFT (79.6%). The gains span six of the seven categories, with the largest improvements under *Robot* (+10.2 pp), *Background* (+8.0 pp), and *Camera* (+6.0 pp); language perturbations are already saturated for both methods (95.0%) and leave no room for further improvement. The categories with the largest gains directly affect either the sensing interface or the robot's spatial configuration, both of which are difficult to identify from a single frame but exposed through how past actions translate into observed consequences.

Discussion. The two LIBERO-Plus-Hard shifts further stress-test this hypothesis. Under *Multi-camera shift* (Camera†), where the extrinsics of both third-person and wrist views are perturbed, Reflective VLA improves from 75.0% to 80.1%. Under *Robot calibration shift* (Rob. Calib†), where the achieved end-effector motion deviates systematically from the commanded action, it improves from 50.2% to 62.5%. Together, these two shifts raise the average success rate from 62.6% to 71.3%, an 8.7-point gain over the matched reactive baseline. Unlike language or appearance perturbations, both shifts change the relationship between actions and their observed effects, making them difficult to identify from any single frame. The largest gain appears on Rob. Calib†, where the only diagnostic signal is the residual between commanded actions and their consequences—directly supporting the central design of Reflective VLA.

Context composition. Table 3a isolates the effect of context composition with K fixed. Adding observation history alone (O) yields no improvement over the reactive baseline (71.0% vs. 71.7%), and adding observation–action pairs without the resulting consequence (O,A) provides only a marginal gain (72.8%). In contrast, the full observation–action–consequence context (O,A,O$'$) improves the average from 71.7% to 79.5%, a 7.8-point gain that is most pronounced on Rob. Calib† (50.2% $\rightarrow$ 62.5%). This supports our hypothesis that the action-aligned consequence $\mathcal{O}'$ provides the key adaptation signal: temporal context alone, even when paired with the executed action, leaves the action-to-observation mapping unidentified.



(a) Real robot setup (b) Camera placement protocol (c) Results on seen and unseen camera setups

Figure 5: **Real-world setup.** (a) An Agilex Piper arm with RealSense D435i cameras over a tabletop workspace, with two tasks (place-into-box, place-into-bowl). (b) Third-person camera-placement protocol: ten placements span the left, front, and right of the workspace; demonstrations cover all ten, while evaluation uses five seen and five held-out placements drawn from the same regions.

Context length. Table 3b studies the effect of context length. Increasing K consistently improves robustness, from 71.7% at $K{=}1$ to 76.7% at $K{=}2$, 78.6% at $K{=}4$, and 79.5% at $K{=}8$. Most of the gain is captured by $K{=}4$ (+6.9 points over the reactive baseline), with diminishing returns thereafter, suggesting that a small number of recent interaction triplets already captures most of the useful deployment-specific evidence.

Latency–accuracy trade-off. Figure 4 reports per-step latency against success rate during inference. Since past triplets are encoded once and reused, latency grows sub-linearly in K: $K{=}8$ is only $1.43\times$ slower than $K{=}1$ (178 vs. 124 ms) despite an $8\times$ longer prompt, while $K{=}4$ already recovers most of the accuracy gain at $1.19\times$ baseline latency—a practical operating point when tighter control rates are required. We cap K at 8 because longer contexts exceed our GPU memory budget during block-causal training; KV-cached inference itself can scale further.

4.3 Real-World Experiments

We further evaluate Reflective VLA under real-world cross-camera generalization. Following the protocol in Figure 5, demonstrations cover ten third-person camera placements spanning the left, front, and right sides of the workspace; at test time we evaluate on five seen and five held-out placements ($N = 25$ trials each) without test-time fine-tuning. We consider two tabletop tasks—placing an object into a large square box (primarily grasping) and into a small bowl (additionally requires camera-dependent placement). Full protocol details are in Section D.

Reflective VLA preserves performance on seen viewpoints while substantially improving generalization to unseen ones. On seen placements, success improves only slightly over the reactive baseline (88%→92% on box, 68%→72% on bowl), indicating that interaction context does not compromise nominal performance when camera geometry is covered by training. Under unseen placements, the reactive baseline degrades sharply (32% box, 16% bowl) while Reflective VLA reaches 76% and 64%. Although the bowl task is harder overall due to its tighter placement tolerance, the gain on unseen viewpoints (+48 pp) matches the box task (+44 pp), suggesting that reflective context benefits both grasping robustness and camera-dependent spatial alignment.

5 Conclusion

We cast cross-environment generalization in VLAs as in-context inference over causal triplets $(\mathcal{O}, A, \mathcal{O}')$, enabling adaptation from interaction feedback without test-time fine-tuning. Two findings support this view: a matched history-only ablation that omits the action aligned consequence $\mathcal{O}'$ recovers little of the gain, showing that context length alone is insufficient; and Reflective VLA closes a substantial portion of the OOD gap on the simulation and real world generalization environments. These gains come at constant model size—only a change in how the policy uses its context window. Scaling context length, extending reflection to longer horizons, and applying this framework to diverse embodiments are natural next steps.

References

AgiBot-World-Contributors, Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, et al. AgiBot World Colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025b.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, Thomas Unterthiner, Daniel Keysers, Skanda Koppula, Fangyu Liu, Adam Grycner, Alexey Gritsenko, Neil Houlsby, Manoj Kumar, Keran Rong, Julian Eisenschlos, Rishabh Kabra, Matthias Bauer, Matko Bošnjak, Xi Chen, Matthias Minderer, Paul Voigtlaender, Ioana Bica, Ivana Balazevic, Joan Puigcerver, Pinelopi Papalampidi, Olivier Henaff, Xi Xiong, Radu Soricut, Jeremiah Harmsen, and Xiaohua Zhai. PaliGemma: A versatile 3B VLM for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

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. In *RSS*, 2025. doi: 10.15607/RSS.2025.XXI.010.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. In *RSS*, 2023. doi: 10.15607/RSS.2023.XIX.025.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, volume 33, 2020.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. In *RSS*, 2025. doi: 10.15607/RSS.2025.XXI.014.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *NeurIPS*, 2021.

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. In *RSS*, 2023. doi: 10.15607/RSS.2023.XIX.026.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. LIBERO-Plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhaoyuan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yuzhe Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. ManiSkill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023.

Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Yu-Chiang Frank Wang, and Fu-En Yang. ThinkAct: Vision-language-action reasoning via reinforced visual latent planning. In *NeurIPS*, 2025.

Intern Robotics. InternVLA-M1: A spatially guided vision-language-action framework for generalist robot policy. *arXiv preprint arXiv:2510.13778*, 2025.

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. In *CoRL*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *RSS*, 2025. doi: 10.15607/RSS.2025.XXI.017.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, Maxime Gazeau, Himanshu Sahni, Satinder Singh, and Volodymyr Mnih. In-context reinforcement learning with algorithm distillation. In *ICLR*, 2023.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, Xiaofan Wang, Bei Liu, Jianlong Fu, Jianmin Bao, Dong Chen, Yuanchun Shi, Jiaolong Yang, and Baining Guo. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024a.

Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, Hang Li, and Tao Kong. Vision-language foundation models as effective robot imitators. In *ICLR*, 2024b.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024c.

Weixin Liang, Lili Yu, Liang Luo, Srini Iyer, Ning Dong, Chunting Zhou, Gargi Ghosh, Mike Lewis, Wen tau Yih, Luke Zettlemoyer, and Xi Victoria Lin. Mixture-of-transformers: A sparse and scalable architecture for multi-modal foundation models. *Transactions on Machine Learning Research*, 2025.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *NeurIPS*, 2023.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: A diffusion foundation model for bimanual manipulation. In *ICLR*, 2025.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, et al. Open x-embodiment: Robotic learning datasets and rt-x models. In *ICRA*, 2024.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

Physical Intelligence. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *ICML*, 2019.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *Transactions on Machine Learning Research*, 2022.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. In *ICLR*, 2026.

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. In *RSS*, 2024. doi: 10.15607/RSS.2024.XX.090.

Homer Rich Walke, Kevin Black, Tony Z. Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, Abraham Lee, Kuan Fang, Chelsea Finn, and Sergey Levine. BridgeData V2: A dataset for robot learning at scale. In *Conference on Robot Learning*, volume 229, pages 1723–1736, 2023.

Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, et al. A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692*, 2026.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *ICLR*, 2022.

Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua B. Tenenbaum, and Chuang Gan. Prompting decision transformer for few-shot policy generalization. In *ICML*, 2022.

Hongyin Zhang, Shuo Zhang, Junxi Jin, Qixin Zeng, Runze Li, and Donglin Wang. Robustvla: Robustness-aware reinforcement post-training for vision-language-action models. *arXiv preprint arXiv:2511.01331*, 2025a.

Jiahui Zhang, Yurui Chen, Yueming Xu, Ze Huang, Yanpeng Zhou, Yu-Jie Yuan, Xinyue Cai, Guowei Huang, Xingyue Quan, Hang Xu, and Li Zhang. 4d-vla: Spatiotemporal vision-language-action pretraining with cross-scene calibration. In *NeurIPS*, 2025b.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Tsung-Yi Lin, Gordon Wetzstein, Ming-Yu Liu, and Donglai Xiang. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *CVPR*, pages 1702–1713, 2025.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *RSS*, 2023. doi: 10.15607/RSS.2023.XIX.016.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Ya-Qin Zhang, Jiangmiao Pang, Jingjing Liu, Tai Wang, and Xianyuan Zhan. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *ICLR*, 2026.

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 *CoRL*, 2023.

Supplementary Material

This supplementary material provides the implementation and experimental details needed to reproduce Reflective VLA. Section A describes the architecture components, triplet construction, context-buffer behavior, block-causal masking, and flow-matching inference procedure. Section C details the training data construction, optimization settings, and evaluation protocol for LIBERO, LIBERO-Plus, LIBERO-Plus-Hard, and SimplerEnv-Bridge. Section B specifies the two diagnostic perturbation families used in LIBERO-Plus-Hard. Section D documents the real-world robot setup, control interface, tasks, and camera-placement protocol. Section E summarizes reproducibility resources, external assets, and current limitations.

A Implementation Details

A.1 Architecture components

Reflective VLA uses the same model family and training budget as the matched reactive baseline, and changes only the conditioning interface. We implement it with a dual-system Mixture-of-Transformers (MoT) architecture: a pretrained VLM encodes the language and multimodal observation prefix, and a continuous flow-matching transformer action expert is attached as a suffix with shared self-attention to the prefix. We instantiate the VLM with either PaliGemma-3B Beyer et al. [2024] or Qwen3-VL-2B Bai et al. [2025a], and use an action expert with hidden dimension 1024. Third-person and wrist images are processed through the native VLM visual pathway, while proprioceptive states and historical action chunks are projected into the token space by lightweight two-layer nonlinear fully connected projectors. Each historical action chunk is represented by eight learned tokens. The matched reactive baseline uses the same backbone, projectors, action expert, optimizer, and data, but sets $K{=}1$ so that no historical triplets are provided.

A.2 Triplet construction and context buffer

For each control step t, the query input contains the instruction, the current multimodal observation, and a bounded history of completed interaction triplets. The current observation comprises the third-person view, left/right wrist views (when available), and the proprioceptive state. For a chunk horizon C, each historical triplet is stored as

$$T_i = \big(\tau_i, \mathcal{O}_i, A_i, \mathcal{O}_{i+C}\big), \tag{8}$$

where $A_i = [a_i, \ldots, a_{i+C-1}]$ is the executed action chunk, projected by g_A into eight learned action tokens. We use $C{=}10$ for LIBERO and SimplerEnv-Bridge, and $C{=}30$ for the real-world experiments.

During training, triplets are sampled only from completed past interactions of the same episode, so each consequence $\mathcal{O}_{i+C}$ strictly precedes the query. To prevent the model from extrapolating the target action from a fixed temporal pattern of neighboring actions, we add a randomized stride of $[0, 15]$ environment steps to the backward spacing.

At inference, the policy maintains a rolling FIFO buffer of completed triplets from the current episode: after each executed chunk, the newly formed triplet is appended and the oldest is evicted once the buffer is full. Triplets are drawn from the buffer with a fixed backward stride for deterministic input layout, and at the start of an episode the model simply conditions on whatever context is available.

A.3 Prompt packing details

VLM input template

```
<|im_start|>user {instruction}
# historical triplets
<|frame_start|> <Third-Person OBS_i> <Wrist OBS_i> <PROPRIO_i>
<|action_start|><ACTION_i><|action_end|>
<Third-Person OBS_{i+C}> <Wrist OBS_{i+C}> <PROPRIO_{i+C}> <|frame_end|>
<|frame_sep|>   ···   <|frame_sep|>
# current query, no target action or consequence
<|frame_start|> <Third-Person OBS_t> <Wrist OBS_t> <PROPRIO_t>
```

B LIBERO-Plus-Hard Specification

B.1 Perturbation overview

LIBERO-Plus-Hard extends LIBERO-Plus with two diagnostic perturbation families targeting factors that are hard to identify from a single frame but directly govern the action-to-observation mapping: camera geometry and robot calibration. We denote them Camera† and Rob. Calib† in the result tables. Task semantics, objects, and language instructions are unchanged.

For each rollout, the perturbation parameters are sampled once at the start of the episode and held fixed throughout. All methods share the same task initializations, seeds, and sampled perturbation instances, and the success criterion follows the original LIBERO task-completion signal. The latent perturbation is not exposed to the model but can be inferred from observation–action–consequence triplets collected during the current episode.

B.2 Multi-camera shift

The multi-camera shift perturbs the visual sensing interface while leaving task, object layout, and robot dynamics unchanged. For the third-person view, we sample an episode-level camera transform with azimuth in $[-75°, 75°]$, elevation in $[0°, 15°]$, distance scale in $[1.0, 2.0]$, and endpoint rotations in $[-10°, 10°]$. For the wrist view, we sample a field of view in $[60°, 90°]$ and one of four image transforms (identity, horizontal flip, vertical flip, $180°$ rotation), applied consistently throughout the rollout. Both views remain visually valid but their geometry differs from the nominal LIBERO setting, so the action-to-pixel mapping must be re-identified online.

B.3 Robot calibration shift

The robot calibration shift perturbs the mapping from commanded actions to the motion executed by the simulator, simulating systematic calibration error, actuation bias, or mechanical offset. At the start of each rollout, we sample a fixed calibration bias and apply it to every absolute end-effector command before stepping the environment. Under the default moderate setting, the translational bias is sampled independently along each Cartesian axis from $[-10, 10]$ mm, and the rotational bias is a fixed axis-angle offset with magnitude up to $3°$; the gripper command is unchanged. We only collect the replayed trajectories that successfully complete the task.

The bias is unobservable from any single frame but recoverable from completed triplets, since the residual between the commanded action stored in context and the realized next observation directly reveals it.

C Experimental Protocol

C.1 Training details

Training data construction. For the standard LIBERO and SimplerEnv-Bridge experiments, we use the official LIBERO demonstrations and BridgeData V2 trajectories, with action targets converted to absolute end-effector control and a chunk horizon of $C{=}10$.

Table 4: Training hyperparameters. Batch size is reported per GPU; all reported runs use 8 H20 GPUs.

Setting	Steps	Batch/GPU	K
LIBERO reactive baseline	90k	32	1
LIBERO Reflective VLA	50k	4	8
LIBERO-Plus / Hard reactive	50k	32	1
LIBERO-Plus / Hard Reflective VLA	90k	4	8
SimplerEnv-Bridge reactive	80k	32	1
SimplerEnv-Bridge Reflective VLA	160k	6	4

For LIBERO-Plus, we use the released LIBERO-Plus training set and convert the trajectories to absolute end-effector control via replay. For LIBERO-Plus-Hard, we follow the same replay-based generation pipeline on the original LIBERO demonstrations, additionally injecting our two diagnostic perturbations during replay: camera-pose shifts and commanded-action biases. We merge the two sets and train a single model on the union, yielding 43,546 trajectories in total.

Optimization. All models are trained with AdamW (β_1=0.9, β_2=0.95, weight decay 0.01, gradient clipping at norm 1.0) under bf16 mixed precision with DeepSpeed ZeRO-2. The action expert and projectors use a peak learning rate of 10^{-4}; the VLM uses a $0.1\times$ multiplier and is held frozen for the initial period in Table 4, after which it follows the same linear warm-up and cosine decay schedule as the rest of the model.

For Reflective VLA on LIBERO/LIBERO-Plus we use K=8 context frames, and K=5 on SimplerEnv-Bridge. Adjacent context frames are separated by one action-chunk stride plus a random gap, sampled from $[0, 15]$ environment steps for LIBERO-family experiments and $[0, 5]$ for SimplerEnv-Bridge; at evaluation the gap is set to zero for deterministic fixed-stride selection.

C.2 Evaluation protocol

All models are evaluated from a fixed checkpoint without test-time fine-tuning, using the same pipeline for the reactive baseline and Reflective VLA. For Reflective VLA, the context buffer is reset per episode and populated online from the policy's own executed chunks and reached observations.

LIBERO. We evaluate the four official suites (*Spatial*, *Object*, *Goal*, *Long*) under absolute end-effector control, with 50 rollouts per task (seed 42) and a horizon of 800 steps (900 for *Long*). Success follows the environment's task-completion signal.

LIBERO-Plus and LIBERO-Plus-Hard. For LIBERO-Plus we evaluate the seven standard perturbation categories with one rollout per task; for LIBERO-Plus-Hard we additionally evaluate the multi-camera shift and the robot-calibration shift (default moderate level) on the LIBERO base suites. Perturbations are generated deterministically from task and rollout ids, so all methods see identical instances.

SimplerEnv-Bridge. We evaluate the four WidowX tasks (spoon on towel, carrot on plate, cube stacking, eggplant in basket) with 24 rollouts per task, reporting per-task success and the unweighted average.

All tables report success rates in percent, averaged over the reported suites or perturbation categories.

C.3 More qualitative results

In Figure 6, we show the qualitative results of Reflective VLA on LIBERO-Plus-Hard dataset.

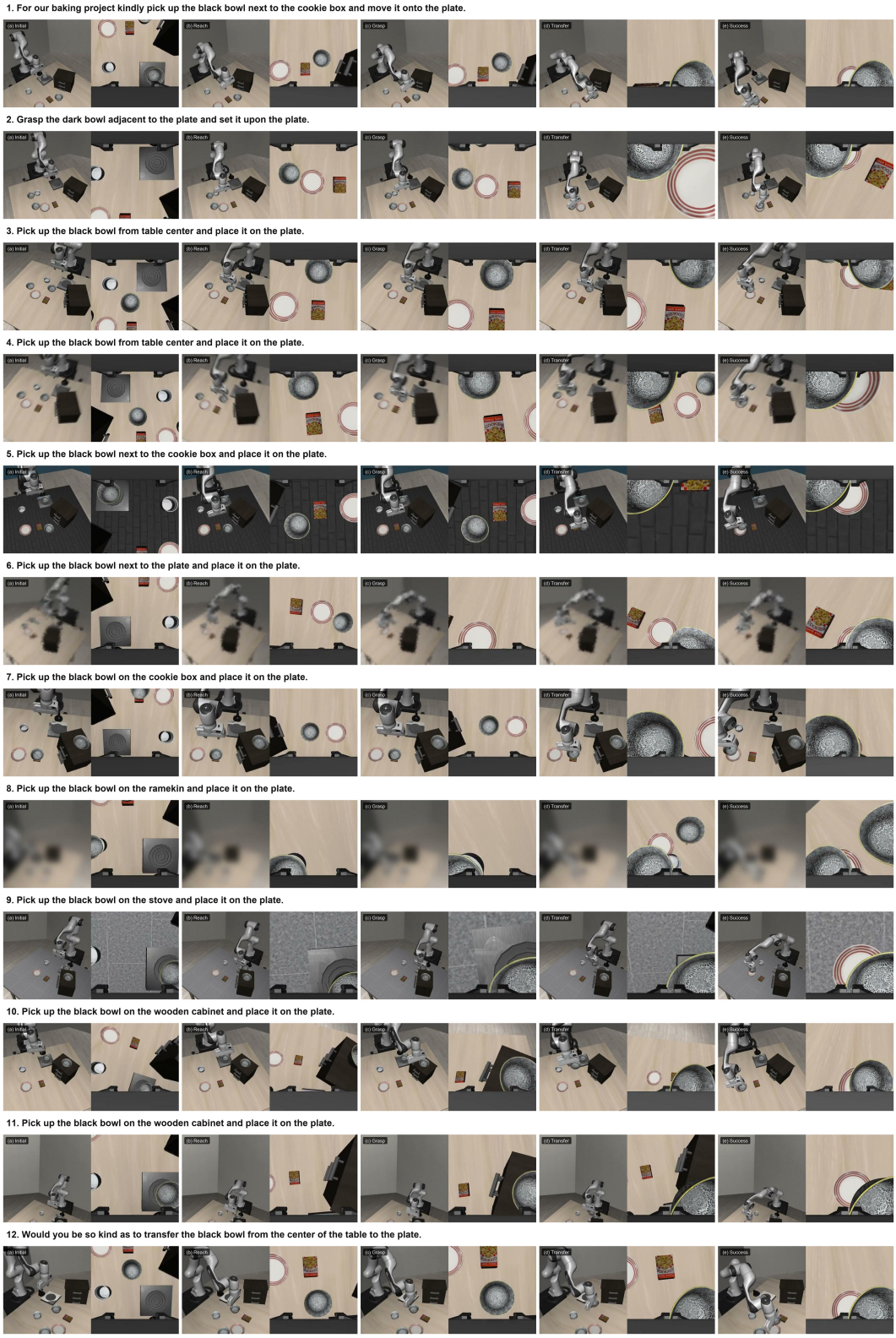


Figure 6: Qualitative results of reflective VLA on LIBERO-Plus-Hard dataset.

D Real-World Protocol

D.1 Hardware and control interface

We use a tabletop setup with an Agilex Piper arm and Intel RealSense D435i cameras. Each policy receives the language instruction, the current third-person view, and the proprioceptive state, and predicts chunked delta end-effector pose and gripper commands with horizon $C{=}30$, executed through the same interface for the reactive baseline and Reflective VLA. For Reflective VLA, the context buffer is reset per trial and populated online from executed chunks and their resulting observations; the reactive baseline conditions only on the current observation and instruction. All hardware, controller, camera streams, and task initializations are identical across methods, with no test-time fine-tuning or camera-specific calibration.

D.2 Tasks and camera placements

We use two language-conditioned pick-and-place tasks: a *box* task (placement into a large square box) and a *bowl* task (placement into a smaller bowl, requiring tighter spatial alignment). We collect 500 demonstrations across ten third-person camera placements spanning the left, front, and right sides of the workspace. Evaluation uses both seen and held-out placements from the same regions, preserving task semantics and hardware while changing camera-to-robot geometry. For each task and condition, we run five trials per placement ($N = 25$ total) with matched initial object configurations; success is judged by a human (target object inside the container at rollout end).

Statistical significance. We report Wilson 95% confidence intervals in Section 4.3. On the *box* task, the reactive baseline reaches 88% [70.0, 95.8] (seen) and 32% [17.2, 51.6] (held-out), versus 92% [75.0, 97.8] and 76% [56.6, 88.5] for Reflective VLA. On the *bowl* task, the corresponding numbers are 68% [48.4, 82.8] / 16% [6.4, 34.7] for the baseline and 72% [52.4, 85.7] / 64% [44.5, 79.8] for Reflective VLA. On both tasks, the held-out intervals do not overlap, indicating that the cross-camera generalization gain is significant at the 95% level; seen-placement intervals overlap, consistent with the small in-distribution gap.

E Reproducibility, Assets, and Limitations

Reproducibility. We provide the codebase for training, evaluation, and perturbation generation in the supplementary material. Refer to the README for setup instructions, dependency versions, and dataset preparation steps; benchmarks depending on external assets are linked to their official downloads rather than redistributed.

Assets. Our simulated experiments build on public assets from LIBERO, LIBERO-Plus, BridgeData V2, and SimplerEnv. Pretrained vision-language backbones are obtained from their official releases under their respective licenses. The LIBERO-Plus-Hard perturbations are specified procedurally in our evaluation scripts (Section B). Real-world experiments use the hardware described in Section D.

Limitations and Future work. Reflective VLA predicts the first chunk reactively, and assumes consequences are observable within the chunk horizon—delayed effects or contact-rich dynamics may need longer contexts. Three further constraints stem from our compute and data budget: (i) we cap context at $K{=}8$ frames due to training memory; (ii) providing history can induce a shortcut where the policy extrapolates from past action chunks instead of reasoning from the current observation, partially mitigated by frame-boundary tokens and stride randomization; (iii) in-context generalization benefits from data diversity, so scaling training data should yield further gains. Our real-world study is also limited to tabletop cross-camera generalization with few trials per condition.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Yes. The abstract and Section 1 state the main claim that observation–action–consequence context enables in-context adaptation for VLA generalization. Section 3 defines the method and Section 4 evaluates the claims on LIBERO, SimplerEnv-Bridge, LIBERO-Plus, and LIBERO-Plus-Hard.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes. The appendix discusses limitations, including the focus on manipulation benchmarks, dependence on observable action consequences, finite context length, additional inference latency, and the limited scale of real-world evaluation.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: N/A. The paper does not present formal theoretical results or proofs. The probabilistic equations in Section 3.2 are used as motivating abstractions for interaction-conditioned adaptation.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Yes. Section 4.1 and the appendices specify the datasets, benchmark splits, perturbation protocols, model architecture, training settings, evaluation metrics, and ablation protocols needed to reproduce the main results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Yes. An anonymized supplementary package includes training and evaluation code, environment/config files, perturbation scripts for LIBERO-Plus-Hard/LIBERO-ID, and raw result tables. Data access and preprocessing instructions for existing benchmarks are documented in the appendix.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes. Section 4.1 and the appendix describe the training and test settings, including datasets, task splits, success metrics, optimizer, learning rate schedule, batch size, training steps, action chunk size, context length, milestone sampling, and evaluation episodes.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Yes. We follow the official evaluation protocol of each benchmark, which prescribes between 20 and 100 trials per scenario; all reported success rates are aggregated over at least 20 trials per scenario. The appendix reports the number of evaluation trials and Wilson binomial confidence intervals for success rates over benchmark episodes, using the same protocol for the main tables and ablations.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes. The appendix reports the compute used for training and evaluation, including GPU type/count, memory, precision, training time, estimated GPU-hours, evaluation hardware, and inference latency.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: Yes. The authors reviewed the NeurIPS Code of Ethics. The experiments use public simulation benchmarks and controlled robot manipulation settings, avoid personal data collection, and follow standard safety procedures for physical robot evaluation.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Yes. The appendix discusses potential positive impacts on robust robot deployment and potential negative impacts, including unsafe physical actions, misuse of autonomous manipulation systems, and over-reliance on adaptation under distribution shift. It also discusses mitigation through controlled evaluation, safety monitoring, and limited release scope.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: N/A. The paper does not release high-risk scraped datasets, pretrained language models, or image generators. The released artifacts are limited to evaluation/training code, benchmark perturbation scripts, and configuration files.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes. The appendix lists all existing assets used in the paper, including LIBERO, LIBERO-Plus, BridgeDataV2, ManiSkill2/SimplerEnv, pretrained backbones, and baseline implementations, with citations, versions, licenses, and terms of use where available.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: N/A. The paper does not introduce new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: N/A. The paper does not involve crowdsourcing experiments or human-subject studies. Robot demonstrations and evaluations do not collect personal, biometric, or behavioral data from study participants.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: N/A. The paper does not involve human-subject research, crowdsourcing, or participant studies requiring IRB or equivalent approval.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: Yes. The core method uses a pretrained vision-language backbone as part of the VLA architecture; its role and interface with the flow-matching action expert are described in Section 3 and the appendix. No LLMs were used to generate experimental data or make evaluation decisions.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.