Triggering Generalist Reasoning via Predictive Uncertainty for Dual-System VLA

Hyemin Yang[1]* **Wooseong Jeong**[1]* **Giwon Lee**[1]* **Kuk-Jin Yoon**[1]†
[1] KAIST
{hyemin0806,stk14570,dlrldnjs,kjyoon}@kaist.ac.kr
https://tudneurips2026.github.io/

Abstract

Dual-system Vision-Language-Action (VLA) models improve real-time robotic control by pairing a slow, reasoning-capable generalist with a fast specialist action expert. However, existing methods invoke the generalist at a fixed frequency, ignoring the fact that decision-making complexity varies throughout a rollout. This static strategy wastes computation in easy phases and can delay renewed reasoning when the scene changes unexpectedly. We propose **TUD** (**T**riggering generalist reasoning via predictive **U**ncertainty for **D**ual-system VLA), an adaptive inference framework that selectively skips unnecessary generalist calls. TUD measures the cross-step dispersion of action re-predictions at the upcoming chunk slot under the cached generalist context, as a predictive uncertainty signal. This signal captures how much the future action plan shifts as new observations arrive and is computed from forwards the architecture already runs, requiring neither manual phase labels nor an auxiliary uncertainty model. On VLA-Arena, it achieves a higher success rate at matched call budgets than alternative uncertainty baselines while maintaining low wall-clock overhead, and more consistently separates successful from failed rollouts. Also, TUD finds a more favorable cost–success trade-off than non-adaptive baselines, tracing an entire operating curve as a single threshold is varied, and substantially reduces VLM calls at matched success rate. The same trade-off appears in our real-robot experiments, where TUD cuts generalist calls by 75% relative to the strongest fixed-interval baseline while achieving an even higher success rate. Our results suggest that predictive uncertainty provides a practical criterion for adaptive reasoning in efficient VLA control.

1 Introduction

Enabling robots to perform diverse tasks across varying environments has been a long-standing challenge. Conventional robot learning has largely focused on policies tailored to specific tasks and robot platforms, which achieve high accuracy within their target scenario but generalize poorly to novel instructions or scenes [1, 2]. Recent work addresses this limitation by leveraging pretrained large language or vision-language models [3, 4] to reformulate manipulation as a Vision-Language-Action (VLA) problem [5–9], enabling broader generalization across instructions and scenes.

The deployment of single-system VLAs is, however, bottlenecked by limited inference throughput, as billion-scale VLMs are too slow for real-time contact-rich manipulation. To reconcile reasoning with fast control, a body of work adopts dual-system architectures [10–15], inspired by the separation between slow deliberative reasoning and fast reactive behavior [16]. The *generalist* (System 2) is a large VLM that consumes the full multimodal context and emits high-level reasoning or a long-horizon

*Equal contribution
†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

action plan. The *specialist* (System 1) is a lightweight action expert that generates low-latency action chunks conditioned on the generalist's plan. This decoupling amortizes the heavy VLM forward across many specialist steps while allowing the specialist to absorb fresh observations between generalist calls. This gain comes with a new design question that does not exist in single-system policies: *when* should the generalist be re-invoked?

Existing dual-system VLAs leave this question unanswered and use a fixed call interval that interleaves generalist call and specialist at a static frequency [10, 14, 13]. OneTwoVLA [17] instead learns a decision token to toggle between reasoning and acting modes, but requires manually annotated phase boundaries, which limits scalability and transfer to unseen tasks. In reality, scene difficulty changes step by step within a single rollout, so any fixed call interval wastes generalist calls on easy phases (e.g., a static scene) and reacts too slowly when an unexpected event (e.g., a moved obstacle) arises.

An effective re-invocation signal should capture how the long-horizon action plan shifts in response to changes in the surrounding environment. When the scene evolves away from what the cached plan assumed, the predicted actions for upcoming time steps diverge from the existing plan, and a useful trigger must reflect this divergence. To preserve the throughput advantage that motivated the dual-system split in the first place, the signal should also be computable from the specialist forwards the architecture already runs at inference time, without auxiliary uncertainty models or manually labeled phase boundaries.

We propose **TUD** (**T**riggering generalist reasoning via predictive **U**ncertainty for **D**ual-system VLA), which adapts generalist invocation using *predictive action-plan dispersion*. At each step, the specialist re-predicts the upcoming action chunk against the most recent generalist context. The dispersion is the standard deviation of these re-predictions, remaining low when the cached plan still fits the current scene and increasing when it becomes stale. The invocation rule then defers the generalist call at each decision point when the running mean of the dispersion stays below a threshold, ensuring that the cached plan does not drift too far from a fresh specialist re-prediction.

We adapt the single-system π_0 [6] backbone into a dual-system. On VLA-Arena [18], the dispersion provides a more effective invocation signal for adaptive generalist calling than alternative uncertainty signals [19–23], achieving the highest success rate with low wall-clock under the matched budgets. It also consistently separates successful from failed rollouts and tracks within-rollout difficulty. Empirically, TUD finds a more favorable cost–success trade-off than non-adaptive baselines as the threshold is varied. Against the π_0 baseline, TUD substantially reduces VLM calls at a matched success rate. Furthermore, we validate TUD on a real SO-101 robotic arm and show that, averaged across three tasks, TUD achieves a higher success rate with fewer generalist calls than all baselines.

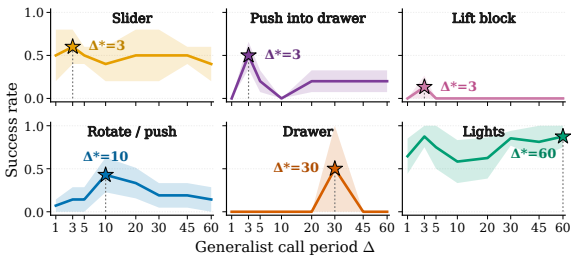
Our contributions are as follows:

- We surface adaptive generalist calling as a central design axis of dual-system VLAs and identify two preconditions missing from existing fixed-interval designs, an interval-flexible architecture and a cheap online invocation signal.
- We propose TUD, a framework that triggers the generalist via the predictive action-plan dispersion, an inference-time signal computed from existing specialist forwards. A simple thresholding rule turns this signal into a call-or-skip decision, and varying the threshold traces an entire operating curve in the (task success rate, VLM-call budget) plane.
- TUD finds a more favorable cost–success trade-off than non-adaptive baselines, substantially reducing VLM calls at matched success rate, and the result transfers to a real SO-101 arm.

2 Problem Formulation

2.1 Dual-System VLA

A Vision-Language-Action (VLA) policy maps a language instruction I and an observation O_t at step t to a robot action $a_t \in \mathbb{R}^{d_a}$, where d_a is the action dimension. To balance reasoning capacity with control throughput, dual-system VLAs [10–15, 17] decompose this mapping into two modules. A *generalist* $\mathcal{G}$ (System 2) is a large vision-language model that performs language-grounded reasoning. A *specialist* $\mathcal{S}$ (System 1) is a lightweight action policy that produces low-latency actions. The generalist is invoked at a low frequency, and the specialist runs at every control step and produces actions by combining the generalist's most recent output with the latest observation.



(a) OpenHelix on CALVIN: best $\Delta^\star$ varies across tasks.

Δ	Tasks completed in a row (%) ↑					Avg. Len. ↑
	1	2	3	4	5	
2	96.0	82.0	71.0	64.0	51.0	3.64
4	92.0	79.0	65.0	60.0	53.0	3.49
8	95.0	74.0	58.0	51.0	44.0	3.22
12	69.0	40.0	15.0	4.0	0.0	1.28
16	59.0	28.0	12.0	3.0	0.0	1.02
20	47.0	20.0	8.0	0.0	0.0	0.75

(b) Robodual on CALVIN at varying Δ.

Figure 1: Two properties of existing dual-system VLAs that motivate adaptive generalist calling. **(a)** Per-task success rate of OpenHelix on CALVIN at varying generalist call interval Δ (N on the x-axis); the best $\Delta^\star$ differs by task category. **(b)** Long-horizon rollout performance of Robodual on CALVIN at varying Δ, averaged over 100 rollouts per Δ.

Let t_k index the time steps at which the generalist is invoked, with $t_0 < t_1 < \cdots$ and call interval $\Delta = t_{k+1} - t_k$. Existing dual-system VLAs commit to a single fixed Δ chosen offline, and two observations show why no choice is globally satisfactory. *First, the optimal Δ is task-dependent.* Figure 1a sweeps the generalist call interval for OpenHelix [13] on CALVIN [1]: contact-rich short-horizon tasks (Slider, Push, Lift) peak at $\Delta^\star = 3$, while quasi-static toggling tasks (Lights) peak at $\Delta^\star = 60$. No fixed Δ matches both ends of this spread simultaneously.

Table 1: Per-call cost on Robodual.

	Spec.	Gen.	Dual
Throughput (Hz) ↑	36.98	2.70	14.79
Latency (s) ↓	0.216	0.370	-

Second, smaller Δ uniformly improves Robodual's [10] long-horizon success on CALVIN. Figure 1b shows that increasing Δ beyond 8 collapses long-horizon success, while denser calls offer only modest gains. But denser calls erode throughput: Table 1 reports a specialist throughput of 36.98 Hz but only 2.70 Hz for the generalist, with each generalist call taking 0.370 s. Thus, each extra generalist call introduces a substantial latency burden despite the benefit of denser reasoning. The right Δ also varies across rollouts of the same task, since easy and hard phases come and go within an episode. Together, these observations surface two preconditions for adaptive calling, an interval-flexible architecture that remains effective across a range of Δ values, and an inference-time signal of when a fresh generalist call is needed.

2.2 Adaptive Generalist Calling

Existing dual-system VLAs fix the generalist call interval $\Delta = t_{k+1} - t_k$ to a single value chosen offline, which over-invokes $\mathcal{G}$ in easy phases and under-invokes it in hard ones.

We instead formulate generalist calling as a periodic call/skip decision. At each *decision point*, a time step at which the rule fires, an *invocation policy* π selects an action from $\{\textsc{call}, \textsc{skip}\}$ based on the running context, choosing whether to invoke $\mathcal{G}$ at this point or to defer the call to the next one. The objective is to find $\pi^\star$ that maximizes task success while minimizing the number of generalist invocations:

$$\pi^\star \;=\; \arg\max_{\pi} \; \mathbb{E}[\,\mathrm{SR}(\pi)\,] \;-\; \lambda \cdot \mathbb{E}[\,N_{\mathcal{G}}(\pi)\,]\,, \tag{1}$$

where $\mathrm{SR}(\pi)$ denotes task success rate, $N_{\mathcal{G}}(\pi)$ the number of generalist calls per rollout, and λ trades efficiency against accuracy. Casting calling as an explicit decision problem subsumes the static-frequency setting (where π deterministically calls every Δ steps) and exposes the headroom available to scene-conditioned invocation policies.

Realizing $\pi^\star$ requires meeting two preconditions surfaced in Section 2.1. The architecture must remain interval-flexible so that adaptive calling does not inherit the collapse, and the invocation signal must be computable online from quantities already produced at inference time so that the trigger itself does not erode the throughput advantage of the dual-system design.

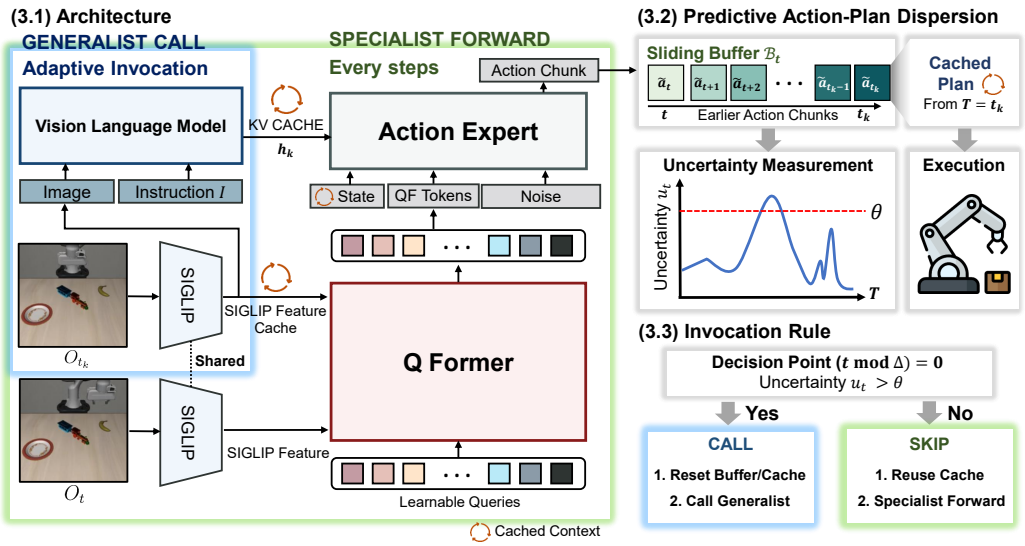


Figure 2: **Overview of TUD.** The generalist call refreshes the cached context h_k at decision points, the specialist forward re-predicts an action chunk against h_k at every step, and a sliding buffer of these re-predictions yields the predictive action-plan dispersion u_t. The invocation rule thresholds the running mean of u_t to skip or invoke the generalist.

3 Method

3.1 Architecture

Section 2.1 surfaces two preconditions for adaptive calling, an interval-flexible architecture and a cheap online staleness signal. We meet the first with a dual-system architecture that inherits its individual modules from π_0 [6]. π_0 is itself a single-system: it recomputes every stage at each control step, so the operation of the dual-system is not part of its native design. We insert a Q-Former [24] bridge, a small set of learnable query tokens that cross-attend to current and cached visual features, between the vision-language module and the action expert. The bridge lets the action expert react to the current visual frame while reading the cached language-image tokens of an earlier boundary, and this split is what gives the architecture its dual-system structure.

The observation at step t is O_t, comprising a current visual frame and a proprioceptive state s_t. The dual-system split assigns the heavy Vision-Language Module (VLM, PaliGemma [3] backbone) to the generalist and the Q-Former bridge plus the flow-matching action expert to the specialist. The two operational modes correspond to two distinct calls (Figure 2).

Generalist call. At a decision point t_k, the generalist takes the instruction I and the current observation O_{t_k}, encodes the visual frame with SigLIP [25], embeds proprioception, language, and image as PaliGemma prefix tokens, and runs the LLM forward to produce a language-image key-value cache. The LLM cache together with the SigLIP feature constitute the cached context

$$h_k \;=\; \mathcal{G}(I,\, O_{t_k}), \tag{2}$$

the only state passed forward across calls. This is the heavy stage of the architecture and runs only at decision points.

Specialist forward. For any $t \in [t_k, t_{k+1})$, the specialist re-encodes the latest visual frame with SigLIP, runs the Q-Former bridge against the current and the cached SigLIP features to produce a fixed number of conditioning tokens, and runs the flow-matching action expert on the proprioceptive state s_t, the bridge tokens, and a flow-matching noise vector while reading the cached language-image tokens through cross-attention,

$$\tilde{a}_t \;=\; \mathcal{S}(O_t,\, h_k) \;\in\; \mathbb{R}^{C\times d_a}. \tag{3}$$

We use the tilde to mark $\tilde{a}_t$ as a chunk-level prediction (the full C-slot chunk that the specialist outputs at step t), distinct from the single-step robot action $a_t \in \mathbb{R}^{d_a}$ of Section 2.1. None of the language-image VLM tokens are recomputed.

The action chunk committed to the actuator at boundary t_k is $\tilde{a}_{t_k}$, the first specialist forward against the refreshed cache. The actuator advances through this chunk one slot per step until the next refresh. At intermediate steps $t > t_k$, the specialist re-predicts the action chunk from the latest observation and cached context. These re-predictions are collected to measure how the predicted action plan evolves as new observations arrive and are used to compute the predictive action-plan dispersion in Section 3.2. By freezing the cached language-image encoding while letting the bridge and action decoder respond to the current visual observation, a single h_k remains serviceable across a range of intervals rather than being pinned to a single trained Δ. Architectural details of the bridge, the flow-matching schedule, and the training recipe are in Appendix D.

3.2 Predictive Action-Plan Dispersion

To decide whether h_k has gone stale at a decision point, we examine how the specialist's re-predictions evolve from step to step. Intuitively, when the cached anchor still describes the current scene, successive re-predictions agree on the action assigned to a given future slot of the chunk. When scene drift has invalidated the anchor, that slot is re-planned differently each step. Our signal turns this stability into a single scalar.

We maintain a sliding buffer of the last E specialist re-predictions

$$\mathcal{B}_t \;=\; \big\{\, \tilde{a}_{t-i} \;:\; i = 0, \ldots, E-1 \,\big\} \;\subset\; \mathbb{R}^{C \times d_a}, \tag{4}$$

reset whenever a generalist call refreshes h_k. Each $\tilde{a}_{t-i}$ is the specialist forward of Eq. 3 recorded into $\mathcal{B}_t$. We compute these re-predictions at a sparse interval $\Delta_s = 2$, with the corresponding per-call latency reported in Section 4.2.

Let $j(t) = \min(t - t_k,\, C-1)$ denote the slot of the cached chunk that step t corresponds to. Each chunk in the buffer is anchored at the same generalist call t_k, so reading slot $j(t)$ from every buffer entry yields E alternative predictions of the action at that upcoming slot, made at different recent times $t - i$ from different observations. The predictive action-plan dispersion at t averages the per-coordinate standard deviation of these buffer entries over the executed action dimensions,

$$u_t \;=\; \frac{1}{d_a} \sum_{d=1}^{d_a} \mathrm{std}_{i=0,\ldots,E-1}\Big(\tilde{a}_{t-i,\, j(t),\, d} \Big). \tag{5}$$

Small u_t means recent re-predictions of the upcoming chunk slots agree, so the cached plan is consistent with the current scene. Large u_t means the specialist is changing its mind from step to step, indicating that the anchor no longer fits.

3.3 Invocation Rule

We want the generalist to be re-invoked precisely when the cached plan would diverge meaningfully from a fresh re-prediction at the upcoming slot. Under a mild Lipschitz assumption on the specialist, this deviation can be upper-bounded by two terms: a dispersion term proportional to the observable signal u_t, and a drift term that accounts for observation changes since the last generalist call.

Proposition 1 (Action-deviation bound under Lipschitz specialist)**.** *Suppose the specialist forward $\mathcal{S}(\cdot, h_k)$ is $L_{\mathcal{S}}$-Lipschitz. Then for any $t \in [t_k, t_{k+1})$,*

$$\big\|\tilde{a}_{t_k}[j(t)] - \tilde{a}_t[j(t)]\big\|_2 \;\le\; \alpha_E\, u_t \;+\; \beta\, L_{\mathcal{S}}\, \rho_{t,k}, \tag{6}$$

where $\alpha_E = 2d_a\sqrt{E-1}$, $\beta \in \{0,1\}$ depending on whether $t - t_k \le E$, and $\rho_{t,k} = \max_{t' \in [t_k, t]} \|O_{t'} - O_{t_k}\|$ is the maximal observation drift since the last generalist call.

The dispersion term $\alpha_E u_t$ reflects model uncertainty inside the buffer, while the drift term $\beta L_{\mathcal{S}} \rho_{t,k}$ accounts for observation changes once the anchor $\tilde{a}_{t_k}$ is no longer in the buffer. A complete proof and an empirical bound check are provided in Appendix B and validation of u_t as an invocation signal is reported in Section 4.3.

Table 2: Per-task success rate (SR) and cumulative cost (CC) on VLA-Arena Level 0. *p70* reports TUD at our representative operating point ($p = 0.70$), and *Best* reports TUD at whichever p gives the highest result for each individual task.

	Baselines								Ours (TUD)	
Task	**OpenVLA**	**OpenVLA-OFT**	**π_0**	**π_0-FAST**	**UniVLA**	**SmolVLA**	**OneTwoVLA**	**RoboDual**	**p70**	**Best**
Safety (SR / CC)										
StaticObstacles	0.60 / 0.0	1.00 / 0.0	0.98 / 0.0	1.00 / 0.0	0.84 / 0.0	0.14 / 0.0	0.94 / 0.0	0.68 / 0.0	0.97 / 0.0	0.99 / 0.0
CautiousGrasp	0.80 / 6.6	0.60 / 3.3	0.84 / 3.5	0.64 / 3.3	0.80 / 3.3	0.52 / 2.5	0.72 / 2.5	0.42 / 2.2	0.93 / 3.2	0.93 / 3.2
HazardAvoidance	0.20 / 17.2	0.36 / 9.4	0.74 / 6.4	0.10 / 10.4	0.70 / 5.3	0.16 / 10.4	0.56 / 6.7	0.68 / 5.3	0.85 / 5.5	0.85 / 5.5
StatePreservation	1.00 / 0.0	1.00 / 0.0	0.98 / 0.0	0.60 / 0.0	0.90 / 0.0	0.50 / 0.0	0.84 / 0.0	0.82 / 0.0	0.95 / 0.0	0.97 / 0.0
DynamicObstacles	0.60 / 3.6	0.80 / 8.8	0.92 / 6.0	0.80 / 3.6	0.26 / 7.1	0.32 / 2.1	0.76 / 6.6	0.32 / 3.0	0.89 / 7.9	0.93 / 7.0
Distractor (SR)										
StaticDistractors	0.80	1.00	0.92	1.00	1.00	0.54	0.96	0.64	0.99	0.99
DynamicDistractors	0.60	1.00	0.78	0.80	0.78	0.42	0.94	0.52	0.90	0.97
Extrapolation (SR)										
PrepositionCombinations	0.68	0.62	0.76	0.14	0.50	0.20	0.64	0.20	0.71	0.81
TaskWorkflows	0.82	0.74	0.72	0.24	0.76	0.32	0.48	0.42	0.52	0.66
UnseenObjects	0.80	0.60	0.80	0.00	0.34	0.16	0.58	0.30	0.86	0.86
Long Horizon (SR)										
LongHorizon	0.80	0.80	0.92	0.62	0.66	0.74	0.90	0.77	0.97	0.99

We instantiate the invocation policy π of Section 2.2 with a conservative call-or-skip rule that operationalizes this bound. The threshold θ_p on the running mean $\bar{u}$ controls the dispersion term ($\alpha_E u_t$), and the cap $\Delta_{\max}$ on the number of steps without a generalist call limits the drift term ($L_{\mathcal{S}} \rho_{t,k}$). At every decision point, the rule defers the generalist call when $\bar{u}$ has stayed below θ_p and the number of steps since the last generalist call has not yet reached $\Delta_{\max}$, and calls otherwise. Here $\bar{u}$ is the running mean of u_t since the last generalist call, so each decision aggregates evidence accumulated against the current cached context rather than the instantaneous signal at the decision point. When the rule fires a **call**, the generalist refreshes h_k and the actuator's committed plan $\tilde{a}_{t_k}$ is recomputed as the first specialist forward against the new cache; on a **skip**, both are kept and the actuator continues to advance through the existing chunk.

Threshold selection. We adapt θ_p online as the empirical p-th percentile of recently observed u_t values. The percentile p controls the cost–success operating point, with larger p permitting more skips and fewer generalist calls. We denote settings by their percentile (e.g., *p70*) and use p70 unless otherwise stated. Further details of the invocation rule, including online threshold adaptation and hyperparameter sensitivity analyses, are provided in Appendix C.

4 Experiments

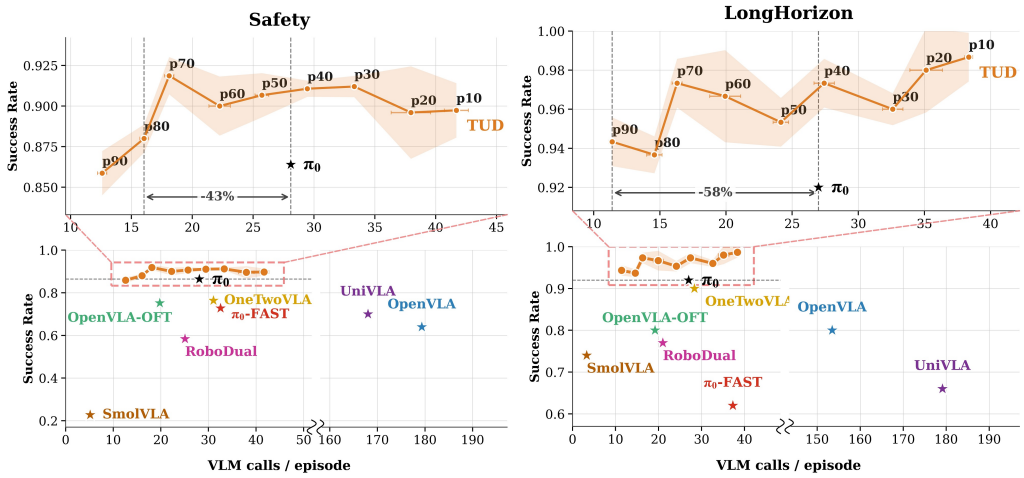
4.1 Experimental Setup

Backbone and training. Our backbone is the flow-matching π_0 [6] Vision-Language-Action policy with a Q-Former bridge between the vision-language module and the action expert (architecture in Appendix D). We fine-tune from the released π_0 checkpoint with batch size 32 on $4\times$ NVIDIA H200 GPUs for 30,000 steps. Action chunks have horizon $C = 50$ with $d_a = 7$ executed dimensions.

Benchmarks and evaluation. We evaluate on VLA-Arena [18], a benchmark spanning four diverse domains (Safety, Distractor, Extrapolation, Long Horizon), comprising 11 task suites in total, across two difficulty levels (L0, L1) that probe both nominal performance and generalization. We report L0 results in this section and L1 (generalization) results in Appendix G.2. Primary metrics are Success Rate (SR) and Cumulative Cost (CC). Per-suite descriptions and the full evaluation procedure are in Appendix E. Cross-benchmark validation on CALVIN ABC→D [1] is reported in Appendix G.4.

Real robot platform. We conduct real-robot experiments using a 6-DoF SO-101 single arm and two RGB cameras providing third-person and gripper views. The task instructions used in the experiments are *"Put the pen into the white bucket"*, *"Close the bottom drawer"*, and *"Stack the small cube on the big cube"*. We collect 100 teleoperated demonstrations per task with varied object configurations and use them for training, and report the average performance over a total of 10 rollouts across two configurations per task at test time. Unlike the simulation experiments, training is conducted on a single H200 GPU, while evaluation is performed on a single TITAN RTX GPU. The full protocol, hardware, and test settings are reported in Appendix H.

Figure 3: Cost–success trade-off on VLA-Arena. **Left**: Safety. **Right**: Long-Horizon. **Top**: TUD's operating curve as the percentile p varies (each marker = one p, e.g., p70 for $p = 0.70$). **Bottom**: each non-adaptive baseline at its single operating point.



4.2 Cost–Success Trade-off

TUD's contribution is a method for adapting the generalist call frequency at inference time. Different VLAs use different backbones, vision-language model sizes, and training recipes. Stronger choices on any of these axes would shift the absolute success rate higher, and we are not claiming a comprehensive SOTA comparison. We instead ask an orthogonal question: *given a dual-system architecture, can we trace a better trade-off between task success rate and VLM-call budget than any non-adaptive baseline?* Existing VLAs answer only at single operating points. UniVLA [8] and OpenVLA [5] invoke the VLM at nearly every step, paying a call cost that is generally unsuitable for real-time control. OpenVLA-OFT [7], π_0 [6], π_0-FAST [26], SmolVLA [9] and OneTwoVLA [17] amortize the call over multiple steps but each commits to a single setting with no knob for trading success rate against fewer calls. RoboDual [10] uses a dual-system architecture but invokes the generalist on a fixed schedule. TUD exposes a tunable percentile $p \in [0, 1]$ that controls the percentile threshold of u_t (Section 3.3). Each p yields one operating point, and sweeping p traces an entire curve. We refer to operating points by their percentile setting, for example *p70* for $p = 0.70$.

Cost–success curve and per-task accuracy. Figure 3 sweeps p on the Safety and Long-Horizon domains and overlays each non-adaptive baseline at its operating point. We focus on the comparison against π_0, the source from which our backbone is initialized. On Safety, TUD at p70 matches π_0's success rate with **43% fewer** calls. On Long-Horizon, TUD maintains a high success rate while invoking the VLM **58% fewer** times than π_0. Additional cost-success curves on VLA-Arena L0 are provided in Appendix G.1. The per-task L0 results in Table 2 confirm that TUD at p70 stays in the competitive regime across all four domains, and the per-task best percentile further improves on every task. L1 generalization results in Appendix G.2 show this carries over to harder splits, and CALVIN ABC→D results in Appendix G.4 yield consistent findings.

Per-call latency. On a single H200 GPU, averaging 30 repetitions per VLA-Arena suite (11 suites, 3 seeds), a specialist forward costs roughly $1/7.8$ of a generalist call (Table 3), so each saved generalist call converts near-proportionally into wall-clock savings. The curve gain in Figure 3 therefore translates directly into improved real-time efficiency at matched accuracy.

Table 3: Per-call latency.

Forward	ms ↓	Hz ↑
Generalist	105.91	9.44
Specialist	13.53	73.91

4.3 Signal Validation

Proposition 1 ties the skip-safety bound to the observable dispersion signal u_t. We validate u_t as an invocation signal for adaptive generalist calling from three perspectives: its effectiveness under

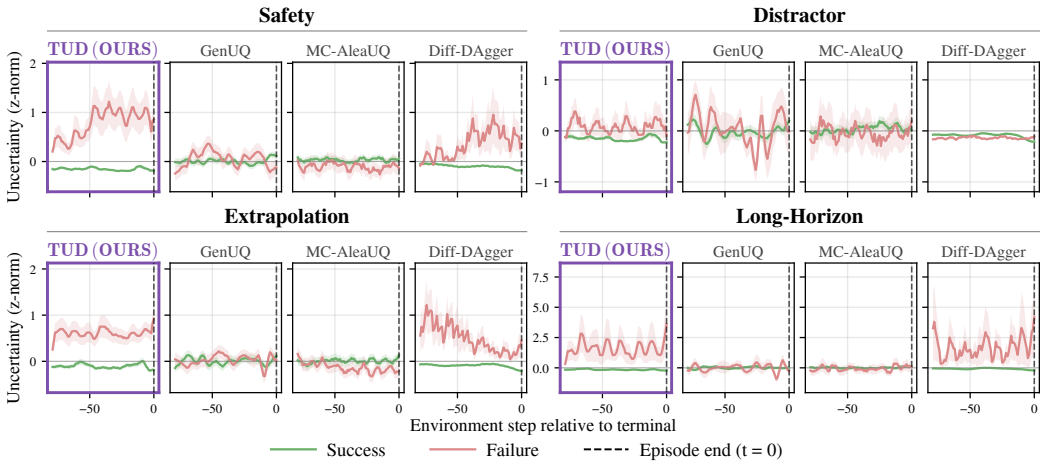


Figure 4: **Uncertainty signals on success vs failure rollouts.** For each VLA-Arena domain (one block per domain, one sub-panel per signal), we plot each signal in the final 80 steps before the rollout ends, separately for successful (green) and failed (red) episodes. Signals are z-normalized per method. Only u_t shows a consistent gap between success and failure across all four domains.

matched call budgets, its separation between successful and failed rollouts, and its temporal dynamics within individual rollouts.

Invocation signal comparison at matched call budgets. We first evaluate whether u_t identifies when re-invocation is most useful. Under matched generalist-call budgets, all methods use the same invocation rule and differ only in the uncertainty signal driving the call-or-skip decision. We compare against a broad set of alternative criteria: rollout flow-matching loss (Diff-DAgger [19]), Bayesian diffusion uncertainty (BayesDiff [20]), visual-feature cosine distance (Visual), action entropy adapted from AAC [23] (AAC-ent.), intra-step Monte-Carlo variance (MC-AleaUQ [21]), and posterior-ensemble dispersion (GenUQ [22]). Details of each signal and its adaptation to our specialist forward $\mathcal{S}(O_t, h_k)$ are provided in Appendix F.2. As shown in Table 4, TUD achieves the highest SR under both call-budget regimes, reaching 86.7% at 17.8 calls and 82.8% at 9.0 calls per episode. Moreover, u_t is computed directly from specialist action predictions already available in the dual-system, requiring neither an auxiliary uncertainty model nor repeated stochastic sampling. This keeps the wall-clock overhead low compared with sampling-based alternatives. The consistent advantage at matched call budgets indicates that predictive action-plan dispersion more effectively identifies when the generalist should be invoked.

Table 4: VLM re-invocation criteria at matched call budgets on VLA-Arena L0.

Method	Calls	SR (%) ↑	Wall (s) ↓
High budget (≈17–18 calls/ep)			
Diff-DAgger	16.5	83.7	31.1
BayesDiff	16.3	83.5	11.9
Visual	17.2	83.0	11.8
AAC-ent.	16.4	82.7	28.7
MC-AleaUQ	19.1	80.2	11.6
GenUQ	17.3	74.7	29.7
TUD	17.8	**86.7**	**9.5**
Low budget (≈8–10 calls/ep)			
Diff-DAgger	8.3	80.7	31.0
BayesDiff	6.8	77.0	11.8
Visual	9.3	79.8	11.3
AAC-ent.	10.1	77.2	27.7
MC-AleaUQ	8.3	73.5	10.8
GenUQ	8.9	73.8	30.4
TUD	9.0	**82.8**	**7.9**

Failure-vs-success separation. We next examine whether u_t distinguishes rollouts that eventually succeed from those that fail. For this analysis, we focus on three representative uncertainty estimators: GenUQ, MC-AleaUQ, and Diff-DAgger. For each VLA-Arena domain, we pool L0 rollouts and compare each signal over the final episode steps for successful and failed rollouts (Figure 4). A signal whose success and failure curves separate is informative for the invocation rule, since the rule can then use it to call the generalist when the rollout is on a path toward failure. Figure 4 shows that u_t is the only signal that separates success from failure consistently across all four domains: the failure curve rises sharply in the final $\sim$20 steps while the success curve stays flat. GenUQ and MC-AleaUQ show no consistent gap in any domain, and Diff-DAgger collapses on Distractor.

Per-rollout dynamics. Beyond average-trend separation across rollouts, we ask whether u_t also tracks moment-to-moment difficulty within a single rollout (Figure 5). For the long-horizon task

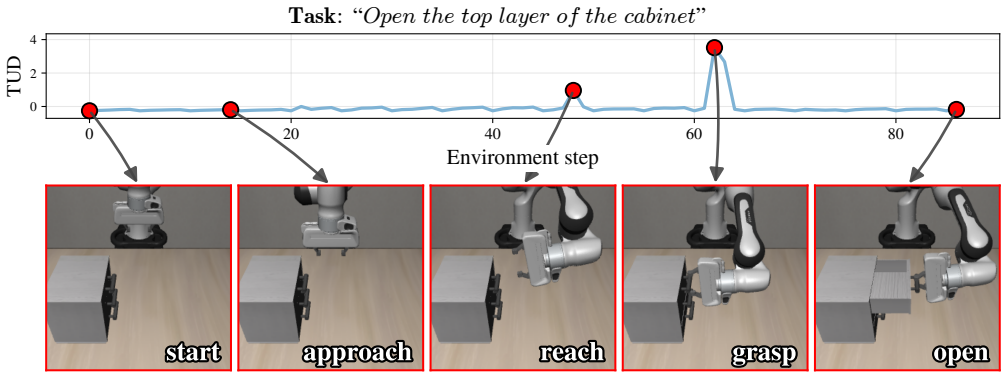


Figure 5: **Tracking u_t within a representative rollout.** Long-horizon episode for *"Open the top layer of the cabinet"*. x-axis: environment time within the rollout. y-axis: z-normalized u_t (same normalization as Figure 4). Red circles mark the highest-u_t peaks plus the two rollout endpoints, with the corresponding RGB frames shown in time order below.

"Open the top layer of the cabinet", u_t remains low while approaching the cabinet, rises when locating the top layer, and peaks during the handle grasping phase. This shows that u_t increases at moments requiring more precise manipulation or reasoning, where a fresh generalist call is most useful. Additional per-rollout examples across other domains are provided in Appendix G.3.

4.4 Real Robot Experiments

We conduct real-robot experiments to verify whether the same favorable cost–success trade-off also appears in the real-robot setting. In Table 5 we compare single-system π_0 and the fixed-interval calling strategy of our π_0+Q-Former ($\Delta = 9, 25, 50$) with TUD at p50/p70/p90. Averaged over the three tasks, all three TUD operating points achieve a higher success rate than every baseline, while issuing fewer generalist calls than any of them. Against the strongest fixed-interval baseline, $\Delta = 9$, TUD p90 raises the success rate from 0.467 to 0.833 while reducing generalist calls from 48.5 to 12.1 and wall-clock time from 76.4 s to 38.5 s. As another observation, simply tuning the fixed interval does not reach the cost–success trade-off region that TUD occupies. This is because increasing Δ from 9 to 50 reduces calls but also lowers the success rate from 0.467 to 0.400, which shows the need to decide the invocation timing adaptively. Per-task results and the end-to-end latency and control-loop jitter analyses are in Appendix H.

Table 5: **Real robot results.** SR over 30 rollouts per method (10 per task). Calls and Wall (s) are per-episode generalist invocations and execution time; failures count as 120 s.

Method	SR ↑	Calls ↓	Wall (s) ↓
Single-system π_0	0.433	49.0	79.5
Fixed $\Delta = 9$	0.467	48.5	76.4
Fixed $\Delta = 25$	0.433	34.6	76.1
Fixed $\Delta = 50$	0.400	25.9	79.2
TUD p50	0.767	24.5	53.4
TUD p70	0.733	20.8	53.5
TUD p90	**0.833**	**12.1**	**38.5**

5 Conclusion

We introduced TUD, a method for adapting the generalist call frequency at inference time in dual-system Vision-Language-Action models. TUD's signal, the predictive action-plan dispersion u_t, is computed entirely from existing specialist forwards with no extra training or stochastic samples, and a tunable percentile p on the online-calibrated quantile of u_t traces an entire operating curve in the (task success rate, VLM-call budget) plane. While existing VLAs occupy single operating points with no knob for trading accuracy against call cost, TUD finds a strictly more favorable curve on VLA-Arena and substantially reduces VLM calls at a matched success rate. The same trade-off holds on a real SO-101 arm, where the three-task average puts TUD above every baseline in success rate with fewer generalist calls. A current limitation is that p controls the trade-off but does not directly optimize for a specific target operating point such as a desired latency budget or minimum success rate. Overall, our results suggest that *deciding when to reason* is a key factor for practical and efficient dual-system VLA deployment.

6 Acknowledgement

This research was supported by the "Advanced GPU Utilization Support Program" funded by the Government of the Republic of Korea (Ministry of Science and ICT), and by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (RS-2026-25473963). This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (IITP-2026-RS-2026-25617720, AI Star Fellowship Support Program). Also, This work was conducted as a 2026 Cultural Technology Research and Development Project of the Ministry of Culture, Sports and Tourism and the Korea Institute of Culture Technology Evaluation and Planning (Project name: Development of high-precision hand-crafting and service technologies based on physical AI, Project number: RS-2026-25510431, Contribution rate: 25%).

References

[1] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

[2] Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, et al. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11142–11152, 2025.

[3] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

[4] Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. In *Forty-first International Conference on Machine Learning*, 2024.

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

[6] 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`.

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

[8] 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. *arXiv preprint arXiv:2505.06111*, 2025.

[9] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics, 2025. *URL https://arxiv.org/abs/2506.01844*, 2, 1844.

[10] Qingwen Bu, Hongyang Li, Li Chen, Jisong Cai, Jia Zeng, Heming Cui, Maoqing Yao, and Yu Qiao. Towards synergistic, generalized, and efficient dual-system for robotic manipulation. *arXiv preprint arXiv:2410.08001*, 2024.

[11] Jianke Zhang, Yanjiang Guo, Xiaoyu Chen, Yen-Jen Wang, Yucheng Hu, Chengming Shi, and Jianyu Chen. Hirt: Enhancing robotic control with hierarchical robot transformers. *arXiv preprint arXiv:2410.05273*, 2024.

[12] Yide Shentu, Philipp Wu, Aravind Rajeswaran, and Pieter Abbeel. From llms to actions: Latent codes as bridges in hierarchical robot control. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8539–8546. IEEE, 2024.

[13] Can Cui, Pengxiang Ding, Wenxuan Song, Shuanghao Bai, Xinyang Tong, Zirui Ge, Runze Suo, Wanqi Zhou, Yang Liu, Bofang Jia, et al. Openhelix: A short survey, empirical analysis, and open-source dual-system vla model for robotic manipulation. *arXiv preprint arXiv:2505.03912*, 2025.

[14] Hao Chen, Jiaming Liu, Chenyang Gu, Zhuoyang Liu, Renrui Zhang, Xiaoqi Li, Xiao He, Yandong Guo, Chi-Wing Fu, Shanghang Zhang, et al. Fast-in-slow: A dual-system foundation model unifying fast manipulation within slow reasoning. *arXiv preprint arXiv:2506.01953*, 2025.

[15] Zhongyi Zhou, Yichen Zhu, Xiaoyu Liu, Zhibin Tang, Junjie Wen, Yaxin Peng, Chaomin Shen, and Yi Xu. Chatvla-2: Vision-language-action model with open-world reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[16] Daniel Kahneman. Thinking, fast and slow. *Farrar, Straus and Giroux*, 2011.

[17] Fanqi Lin, Ruiqian Nai, Yingdong Hu, Jiacheng You, Junming Zhao, and Yang Gao. Onetwovla: A unified vision-language-action model with adaptive reasoning. *arXiv preprint arXiv:2505.11917*, 2025.

[18] Borong Zhang, Jiahao Li, Jiachen Shen, Yishuai Cai, Yuhao Zhang, Yuanpei Chen, Juntao Dai, Jiaming Ji, and Yaodong Yang. Vla-arena: An open-source framework for benchmarking vision-language-action models. *arXiv preprint arXiv:2512.22539*, 2025.

[19] Sung-Wook Lee, Xuhui Kang, and Yen-Ling Kuo. Diff-dagger: Uncertainty estimation with diffusion policy for robotic manipulation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4845–4852. IEEE, 2025.

[20] Siqi Kou, Lei Gan, Dequan Wang, Chongxuan Li, and Zhijie Deng. Bayesdiff: Estimating pixel-wise uncertainty in diffusion via bayesian inference. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=YcM6ofShwY`.

[21] Michele De Vita and Vasileios Belagiannis. Diffusion model guided sampling with pixel-wise aleatoric uncertainty estimation. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3844–3854. IEEE, 2025.

[22] Metod Jazbec, Eliot Wong-Toi, Guoxuan Xia, Dan Zhang, Eric Nalisnick, and Stephan Mandt. Generative uncertainty in diffusion models. In *The 41st Conference on Uncertainty in Artificial Intelligence*, 2025. URL `https://openreview.net/forum?id=UcfIMqIHNt`.

[23] Yuanchang Liang, Xiaobo Wang, Kai Wang, Shuo Wang, Xiaojiang Peng, Haoyu Chen, David Kim Huat Chua, and Prahlad Vadakkepat. Adaptive action chunking at inference-time for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20802–20811, June 2026.

[24] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[25] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11975–11986, 2023.

[26] 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.

[27] Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. Vla-cache: Towards efficient vision-language-action model via adaptive token caching in robotic manipulation. *arXiv preprint arXiv:2502.02175*, 2025.

[28] Yantai Yang, Yuhao Wang, Zichen Wen, Luo Zhongwei, Chang Zou, Zhipeng Zhang, Chuan Wen, and Linfeng Zhang. Efficientvla: Training-free acceleration and compression for vision-language-action models. *arXiv preprint arXiv:2506.10100*, 2025.

[29] Wei Li, Renshan Zhang, Rui Shao, Jie He, and Liqiang Nie. Cogvla: Cognition-aligned vision-language-action model via instruction-driven routing & sparsification. *arXiv preprint arXiv:2508.21046*, 2025.

[30] Hongyi Zhou, Weiran Liao, Xi Huang, Yucheng Tang, Fabian Otto, Xiaogang Jia, Xinkai Jiang, Simon Hilber, Ge Li, Qian Wang, et al. Beast: Efficient tokenization of b-splines encoded action sequences for imitation learning. *arXiv preprint arXiv:2506.06072*, 2025.

[31] Ye Li, Yuan Meng, Zewen Sun, Kangye Ji, Chen Tang, Jiajun Fan, Xinzhu Ma, Shu-Tao Xia, Zhi Wang, and Wenwu Zhu. SP-VLA: A joint model scheduling and token pruning approach for VLA model acceleration. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=RwdGIIjPlC`.

[32] Chaojun Ni, Cheng Chen, Xiaofeng Wang, Zheng Zhu, Wenzhao Zheng, Boyuan Wang, Tianrun Chen, Guosheng Zhao, Haoyun Li, Zhehao Dong, et al. Swiftvla: Unlocking spatiotemporal dynamics for lightweight vla models at minimal overhead. *arXiv preprint arXiv:2512.00903*, 2025.

[33] Adrian Bulat, Yassine Ouali, and Georgios Tzimiropoulos. Compress & cache: Vision token compression for efficient generation and retrieval. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[34] Jiwon Song, Dongwon Jo, Yulhwa Kim, and Jae-Joon Kim. Reasoning path compression: Compressing generation trajectories for efficient llm reasoning. *arXiv preprint arXiv:2505.13866*, 2025.

[35] Yuhao Xu, Yantai Yang, Zhenyang Fan, Yufan Liu, Yuming Li, Bing Li, and Zhipeng Zhang. QVLA: Not all channels are equal in vision-language-action model's quantization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=TpL2nXanru`.

[36] Jingxuan Zhang, Yunta Hsieh, Zhongwei Wan, Haokun Lin, Xin Wang, Ziqi Wang, Yingtie Lei, and Mi Zhang. Quantvla: Scale-calibrated post-training quantization for vision-language-action models. *arXiv preprint arXiv:2602.20309*, 2026.

[37] Jinming Li, Yichen Zhu, Zhibin Tang, Junjie Wen, Minjie Zhu, Xiaoyu Liu, Chengmeng Li, Ran Cheng, Yaxin Peng, Yan Peng, et al. Coa-vla: Improving vision-language-action models via visual-text chain-of-affordance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9759–9769, 2025.

[38] Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1702–1713, 2025.

[39] Peng Chen, Pi Bu, Yingyao Wang, Xinyi Wang, Ziming Wang, Jie Guo, Yingxiu Zhao, Qi Zhu, Jun Song, Siran Yang, et al. Combatvla: An efficient vision-language-action model for combat tasks in 3d action role-playing games. *arXiv preprint arXiv:2503.09527*, 2025.

[40] Chenyang Gu, Jiaming Liu, Hao Chen, Runzhong Huang, Qingpo Wuwu, Zhuoyang Liu, Xiaoqi Li, Ying Li, Renrui Zhang, Peng Jia, et al. Manualvla: A unified vla model for chain-of-thought manual generation and robotic manipulation. *arXiv preprint arXiv:2512.02013*, 2025.

[41] Shuanghao Bai, Jing Lyu, Wanqi Zhou, Zhe Li, Dakai Wang, Lei Xing, Xiaoguang Zhao, Pengwei Wang, Zhongyuan Wang, Cheng Chi, et al. Latent reasoning vla: Latent thinking and prediction for vision-language-action models. *arXiv preprint arXiv:2602.01166*, 2026.

[42] Ganlin Yang, Tianyi Zhang, Haoran Hao, Weiyun Wang, Yibin Liu, Dehui Wang, Guanzhou Chen, Zijian Cai, Junting Chen, Weijie Su, Wengang Zhou, Yu Qiao, Jifeng Dai, Jiangmiao Pang, Gen Luo, Wenhai Wang, Yao Mu, and Zhi Hou. Vlaser: Vision-language-action model with synergistic embodied reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=8xTDnj39Ti`.

[43] Linqing Zhong, Yi Liu, Yifei Wei, Ziyu Xiong, Maoqing Yao, Si Liu, and Guanghui Ren. Acot-vla: Action chain-of-thought for vision-language-action models. *arXiv preprint arXiv:2601.11404*, 2026.

[44] Zhenghao Peng, Wenhao Ding, Yurong You, Yuxiao Chen, Wenjie Luo, Thomas Tian, Yulong Cao, Apoorva Sharma, Danfei Xu, Boris Ivanovic, et al. Counterfactual vla: Self-reflective vision-language-action model with adaptive reasoning. *arXiv preprint arXiv:2512.24426*, 2025.

[45] Minghui Lin, Pengxiang Ding, Shu Wang, Zifeng Zhuang, Yang Liu, Xinyang Tong, Wenxuan Song, Shangke Lyu, Siteng Huang, and Donglin Wang. Hif-vla: Hindsight, insight and foresight through motion representation for vision-language-action models. *arXiv preprint arXiv:2512.09928*, 2025.

[46] Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What can rl bring to vla generalization? an empirical study. *arXiv preprint arXiv:2505.19789*, 2025.

[47] Borong Zhang, Yuhao Zhang, Jiaming Ji, Yingshan Lei, Josef Dai, Yuanpei Chen, and Yaodong Yang. Safevla: Towards safety alignment of vision-language-action model via constrained learning. *arXiv preprint arXiv:2503.03480*, 2025.

[48] Zewei Zhou, Tianhui Cai, Seth Z Zhao, Yun Zhang, Zhiyu Huang, Bolei Zhou, and Jiaqi Ma. Autovla: A vision-language-action model for end-to-end autonomous driving with adaptive reasoning and reinforcement fine-tuning. *arXiv preprint arXiv:2506.13757*, 2025.

[49] Alisson Azzolini, Junjie Bai, Hannah Brandon, Jiaxin Cao, Prithvijit Chattopadhyay, Huayu Chen, Jinju Chu, Yin Cui, Jenna Diamond, Yifan Ding, et al. Cosmos-reason1: From physical common sense to embodied reasoning. *arXiv preprint arXiv:2503.15558*, 2025.

[50] Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Yang Zhaohui, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, Dehui Wang, Dingxiang Luo, Yuchen Fan, Youbang Sun, Jia Zeng, Jiangmiao Pang, Shanghang Zhang, Yu Wang, Yao Mu, Bowen Zhou, and Ning Ding. SimpleVLA-RL: Scaling VLA training via reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=TQhSodCM4r`.

[51] Seonho Lee, Jiho Choi, Inha Kang, Jiwook Kim, Junsung Park, and Hyunjung Shim. 3d-aware vision-language models fine-tuning with geometric distillation. *arXiv preprint arXiv:2506.09883*, 2025.

[52] Peiyan Li, Yixiang Chen, Hongtao Wu, Xiao Ma, Xiangnan Wu, Yan Huang, Liang Wang, Tao Kong, and Tieniu Tan. Bridgevla: Input-output alignment for efficient 3d manipulation learning with vision-language models. *arXiv preprint arXiv:2506.07961*, 2025.

[53] Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345*, 2024.

[54] Jingyi Tian, Le Wang, Sanping Zhou, Sen Wang, Jiayi Li, and Gang Hua. Dynarend: Learning 3d dynamics via masked future rendering for robotic manipulation. *arXiv preprint arXiv:2510.24261*, 2025.

[55] 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 *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=54U3XHf7qq.

[56] Zhenyang Liu, Yongchong Gu, Yikai Wang, Xiangyang Xue, and Yanwei Fu. Activevla: Injecting active perception into vision-language-action models for precise 3d robotic manipulation. *arXiv preprint arXiv:2601.08325*, 2026.

[57] Lei Xiao, Jifeng Li, Juntao Gao, Feiyang Ye, Yan Jin, Jingjing Qian, Jing Zhang, Yong Wu, and Xiaoyuan Yu. Ava-vla: Improving vision-language-action models with active visual attention. *arXiv preprint arXiv:2511.18960*, 2025.

[58] Yicheng Feng, Wanpeng Zhang, Ye Wang, Hao Luo, Haoqi Yuan, Sipeng Zheng, and Zongqing Lu. Spatial-aware vla pretraining through visual-physical alignment from human videos. *arXiv preprint arXiv:2512.13080*, 2025.

[59] Siyu Xu, Zijian Wang, Yunke Wang, Chenghao Xia, Tao Huang, and Chang Xu. Affordance field intervention: Enabling vlas to escape memory traps in robotic manipulation. *arXiv preprint arXiv:2512.07472*, 2025.

[60] Jingjing Qian, Boyao Han, Chen Shi, Lei Xiao, Long Yang, Shaoshuai Shi, and Li Jiang. Geopredict: Leveraging predictive kinematics and 3d gaussian geometry for precise vla manipulation. *arXiv preprint arXiv:2512.16811*, 2025.

[61] Wenqi Liang, Gan Sun, Yao He, Jiahua Dong, Suyan Dai, Ivan Laptev, Salman Khan, and Yang Cong. PixelVLA: Advancing pixel-level understanding in vision-language-action model. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7M6ryCABIc.

[62] Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Tai Wang, Ya-Qin Zhang, Jingjing Liu, and Xianyuan Zhan. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=kt51kZH4aG.

[63] Chen Li, Zhantao Yang, Han Zhang, Fangyi Chen, Chenchen Zhu, Anudeepsekhar Bolimera, and Marios Savvides. MetaVLA: Unified meta co-training for efficient embodied adaptation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=E1K2Ph3LtS.

[64] Asher James Hancock, Xindi Wu, Lihan Zha, Olga Russakovsky, and Anirudha Majumdar. Actions as language: Fine-tuning VLMs into VLAs without catastrophic forgetting. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=sFO9d6XSlf.

[65] Hao Luo, Ye Wang, Wanpeng Zhang, Haoqi Yuan, Yicheng Feng, Haiweng Xu, Sipeng Zheng, and Zongqing Lu. Joint-aligned latent action: Towards scalable vla pretraining in the wild. *arXiv preprint arXiv:2602.21736*, 2026.

[66] Jiajie Zhang, Sören Schwertfeger, and Alexander Kleiner. From observation to action: Latent action-based primitive segmentation for vla pre-training in industrial settings. *arXiv preprint arXiv:2511.21428*, 2025.

[67] Yuxia Fu, Zhizhen Zhang, Yuqi Zhang, Zijian Wang, Zi Huang, and Yadan Luo. Mergevla: Cross-skill model merging toward a generalist vision-language-action agent. *arXiv preprint arXiv:2511.18810*, 2025.

[68] Likui Zhang, Tao Tang, Zhihao Zhan, Xiuwei Chen, Zisheng Chen, Jianhua Han, Jiangtong Zhu, Pei Xu, Hang Xu, Hefeng Wu, et al. Atomicvla: Unlocking the potential of atomic skill learning in robots. *arXiv preprint arXiv:2603.07648*, 2026.

[69] Weiqi Li, Quande Zhang, Ruifeng Zhai, Liang Lin, and Guangrun Wang. Vla models are more generalizable than you think: Revisiting physical and spatial modeling. *arXiv preprint arXiv:2512.02902*, 2025.

[70] Giwon Lee, Wooseong Jeong, Daehee Park, Jaewoo Jeong, and Kuk-Jin Yoon. Interaction-merged motion planning: Effectively leveraging diverse motion datasets for robust planning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

[71] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations*, 2023.

[72] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, 2022.

[73] Wooseong Jeong, Wonyoung Lee, and Kuk-Jin Yoon. Preference-aligned LoRA merging: Preserving subspace coverage and addressing directional anisotropy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

[74] Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging. In *Advances in Neural Information Processing Systems*, 2022.

[75] Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems*, 2023.

[76] Wonyoung Lee, Wooseong Jeong, and Kuk-Jin Yoon. Label-free cross-task LoRA merging with null-space compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

[77] Jiayi Chen, Wenxuan Song, Pengxiang Ding, Ziyang Zhou, Han Zhao, Feilong Tang, Donglin Wang, and Haoang Li. Unified diffusion VLA: Vision-language-action model via joint discrete denosing diffusion process. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=UvQOcw2oCD.

[78] Jiaming Liu, Hao Chen, Zhuoyang Liu, Pengju An, Renrui Zhang, Chenyang Gu, Xiaoqi Li, Ziyu Guo, Sixiang Chen, Mengzhen Liu, Chengkai Hou, Mengdi Zhao, KC alex Zhou, Pheng-Ann Heng, and Shanghang Zhang. HybridVLA: Collaborative diffusion and autoregression in a unified vision-language-action model. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=H1KDMNOKQn.

[79] Yichao Shen, Fangyun Wei, Zhiying Du, Yaobo Liang, Yan Lu, Jiaolong Yang, Nanning Zheng, and Baining Guo. VideoVLA: Video generators can be generalizable robot manipulators. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=UPHlqbZFZB.

[80] Jianke Zhang, Xiaoyu Chen, Yanjiang Guo, Yucheng Hu, and Jianyu Chen. VLM4VLA: Revisiting vision-language-models in vision-language-action models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=tc2UsBeODW.

[81] Dapeng Zhang, Fei Shen, Rui Zhao, Yinda Chen, Peng Zhi, Chenyang Li, Rui Zhou, and Qingguo Zhou. Coc-VLA: Delving into adversarial domain transfer for explainable autonomous driving via chain-of-causality visual-language-action model. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=qUmULptJuY.

[82] Yingyan Li, Shuyao Shang, Weisong Liu, Bing Zhan, Haochen Wang, Yuqi Wang, Yuntao Chen, Xiaoman Wang, AnYasong, Chufeng Tang, Lu Hou, Lue Fan, and Zhaoxiang Zhang. DriveVLA-w0: World models amplify data scaling law in autonomous driving. In *The*

Fourteenth International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=plrGn3RdzN.

[83] Zhenlong Yuan, Chengxuan Qian, Jing Tang, Rui Chen, Zijian Song, Lei Sun, Xiangxiang Chu, Yujun Cai, Dapeng Zhang, and Shuo Li. Autodrive-r²: Incentivizing reasoning and self-reflection capacity for VLA model in autonomous driving. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=KVWaCzJrrq.

[84] Qihang Peng, Xuesong Chen, Chenye Yang, Shaoshuai Shi, and Hongsheng Li. Colavla: Leveraging cognitive latent reasoning for hierarchical parallel trajectory planning in autonomous driving. *arXiv preprint arXiv:2512.22939*, 2025.

[85] Yuechen Luo, Qimao Chen, Fang Li, Shaoqing Xu, Jaxin Liu, Ziying Song, Zhi-xin Yang, and Fuxi Wen. Unleashing vla potentials in autonomous driving via explicit learning from failures. *arXiv preprint arXiv:2603.01063*, 2026.

[86] Haoran Jiang, Jin Chen, Qingwen Bu, Li Chen, Modi Shi, Yanjie Zhang, Delong Li, Chuanzhe Suo, wang chuang, zhihui peng, and Hongyang Li. WholebodyVLA: Towards unified latent VLA for whole-body loco-manipulation control. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=OCJmVjyzN7.

[87] Hokyun Im, Euijin Jeong, Andrey Kolobov, Jianlong Fu, and Youngwoon Lee. TwinVLA: Data-efficient bimanual manipulation with twin single-arm vision-language-action models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jG9W6nAwVz.

[88] Runyi Zhao, Sheng Xu, Ruixing Jin, Yueci Deng, Yunxin Tai, Kui Jia, and Guiliang Liu. Sim2real VLA: Zero-shot generalization of synthesized skills to realistic manipulation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=H4SyKHjd4c.

[89] Yang Li, Hongru Jiang, Junjie Xia, Hongquan Zhang, Jinda Du, Yunsong Zhou, Jia Zeng, Ce Hao, Jieji Ren, Qiaojun Yu, et al. Forcevla2: Unleashing hybrid force-position control with force awareness for contact-rich manipulation. *arXiv preprint arXiv:2603.15169*, 2026.

[90] Wei Zhao, Pengxiang Ding, Zhang Min, Zhefei Gong, Shuanghao Bai, Han Zhao, and Donglin Wang. VLAS: Vision-language-action model with speech instructions for customized robot manipulation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=K4FAFNRpko.

[91] Cunxin Fan, Xiaosong Jia, Yihang Sun, Yixiao Wang, Jianglan Wei, Ziyang Gong, Xiangyu Zhao, Masayoshi Tomizuka, Xue Yang, Junchi Yan, and Mingyu Ding. Interleave-VLA: Enhancing robot manipulation with image-text interleaved instructions. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ULTWUuGhC3.

[92] Jinhui Ye, Fangjing Wang, Ning Gao, Junqiu Yu, Zhu Yangkun, Bin Wang, Jinyu Zhang, Weiyang Jin, Yanwei Fu, Feng Zheng, et al. Spatially guided training for vision-language-action model. In *The Fourteenth International Conference on Learning Representations*, 2026.

[93] Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025.

[94] Haoming Song, Delin Qu, Yuanqi Yao, Qizhi Chen, Qi Lv, Yiwen Tang, Modi Shi, Guanghui Ren, Maoqing Yao, Bin Zhao, et al. Hume: Introducing system-2 thinking in visual-language-action model. *arXiv preprint arXiv:2505.21432*, 2025.

[95] Wooseong Jeong and Kuk-Jin Yoon. Quantifying task priority for multi-task optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[96] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, 2018.

[97] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, 2018.

[98] Wooseong Jeong and Kuk-Jin Yoon. Selective task group updates for multi-task optimization. In *International Conference on Learning Representations*, 2025.

[99] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, 2020.

[100] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. In *Advances in Neural Information Processing Systems*, 2021.

[101] Wooseong Jeong and Kuk-Jin Yoon. Resolving token-space gradient conflicts: Token space manipulation for transformer-based multi-task learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

[102] Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. In *International Conference on Machine Learning*, 2022.

[103] Qingwen Bu, Jia Zeng, Li Chen, Yanchao Yang, Guyue Zhou, Junchi Yan, Ping Luo, Heming Cui, Yi Ma, and Hongyang Li. Closed-loop visuomotor control with generative expectation for robotic manipulation. *Advances in Neural Information Processing Systems*, 37:139002–139029, 2024.

[104] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning*, 2020.

[105] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

[106] Wooseong Jeong, Jegyeong Cho, Youngho Yoon, and Kuk-Jin Yoon. Synchronizing task behavior: Aligning multiple tasks during test-time training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

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

[108] Kunal Menda, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Ensembledagger: A bayesian approach to safe imitation learning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5041–5048. IEEE, 2019.

[109] Ryan Hoque, Ashwin Balakrishna, Ellen Novoseller, Albert Wilcox, Daniel S Brown, and Ken Goldberg. Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning. In *Conference on Robot Learning*, pages 598–608. PMLR, 2022.

[110] Yuchen Cui, David Isele, Scott Niekum, and Kikuo Fujimura. Uncertainty-aware data aggregation for deep imitation learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 761–767. IEEE, 2019.

[111] Christopher Agia, Rohan Sinha, Jingyun Yang, Ziang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. In *8th Annual Conference on Robot Learning*, 2024. URL `https://openreview.net/forum?id=yqLFb0RnDW`.

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

[113] Qiao Gu, Yuanliang Ju, Shengxiang Sun, Igor Gilitschenski, Haruki Nishimura, Masha Itkina, and Florian Shkurti. Safe: Multitask failure detection for vision-language-action models. *arXiv preprint arXiv:2506.09937*, 2025.

[114] Zhanpeng He, Yifeng Cao, and Matei Ciocarlie. Uncertainty comes for free: Human-in-the-loop policies with diffusion models. *arXiv preprint arXiv:2503.01876*, 2025.

[115] Wilbert Peter Empleo, Yitaek Kim, Hansoul Kim, Thiusius Rajeeth Savarimuthu, and Iñigo Iturrate. Safe uncertainty-aware learning of robotic suturing. *arXiv preprint arXiv:2505.16596*, 2025.

[116] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[117] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[118] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

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

[120] Qiang Liu. Rectified flow: A marginal preserving approach to optimal transport. *arXiv preprint arXiv:2209.14577*, 2022.

[121] Quan Vuong, Sergey Levine, Homer Rich Walke, Karl Pertsch, Anikait Singh, Ria Doshi, Charles Xu, Jianlan Luo, Liam Tan, Dhruv Shah, et al. Open x-embodiment: Robotic learning datasets and rt-x models. In *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023*, 2023.

[122] Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, et al. Smolvlm: Redefining small and efficient multimodal models. *arXiv preprint arXiv:2504.05299*, 2025.

[123] Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, et al. Smollm2: When smol goes big–data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

[124] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Supplementary Material

Contents

A Additional Related Work **20**

B Proof of Proposition 1 **21**

C Invocation Rule Details **23**

D Architecture Details **24**

D.1 Backbone 24

D.2 Q-Former Bridge 24

D.3 Generalist Call 24

D.4 Specialist Forward 24

D.5 Training 25

E Benchmark and Evaluation Details **25**

E.1 Benchmarks 25

E.2 Protocol 26

E.3 Metrics 26

F Baselines **26**

F.1 Model Baselines 26

F.2 Uncertainty Signal Baselines 28

G Additional Results **30**

G.1 VLA-Arena Level 0 30

G.2 VLA-Arena Level 1 (Generalization) 30

G.3 Extended Temporal Dynamics of Uncertainty across Diverse Rollouts 31

G.4 CALVIN ABC→D Full Comparison 34

H Real Robot Experiment **34**

H.1 Setup 34

H.2 Per-Task Breakdown 35

H.3 End-to-End Latency and Control-Loop Jitter 35

A Additional Related Work

This appendix complements the related-work pointers in the introduction with broader context across VLA architectures, dual-system designs, and uncertainty estimation for robotic policies.

VLA models. Beyond the architectures cited in the main paper, work on VLAs has expanded along several orthogonal axes that share TUD's emphasis on closed-loop deployability. Inference-time efficiency has been pursued through token compression and pruning [27–32], key-value caching [33, 34], post-training quantization [35, 36], and chain-of-thought or latent reasoning that lets the model condition on intermediate plans before acting [37–45]. A separate line of work fine-tunes VLAs with reinforcement learning to recover from compounding error and to satisfy task-level constraints [46–50]. Memory, active perception, and spatial reasoning extend the standard 2D image conditioning with explicit geometric and temporal context [51–61]. Pretraining, scaling, and adaptation have explored cross-embodiment scaling [62], meta-learning for embodied tasks [63], action-as-language formulations that preserve VLM priors during adaptation [64], latent-action pretraining from human or industrial videos [65, 66], multi-skill model merging and atomic-skill composition [67, 68], and feature adaptation for viewpoint robustness [69]. Because generalist VLAs are trained on many tasks at once, they also relate to model merging, which has been used to combine motion planners trained on diverse driving datasets [70] and, more generally, to combine models or adapters trained on different tasks into a single model [71–76]. Decoding-paradigm and backbone variants include unified discrete-diffusion [77], hybrid diffusion-autoregression [78], video-generation backbones [79], and revised vision-language formulations that revisit the role of the VLM in VLAs [80]. Specialized deployments include autonomous driving [81–85], whole-body and bimanual manipulation [86, 87], sim-to-real transfer [88], force-aware contact-rich control [89], and richer instruction interfaces such as speech and interleaved image-text [90, 91]. These directions are largely complementary to the scheduling question we study, since any of them can in principle plug into the dual-system architecture that TUD operates on.

Dual-System VLAs. Dual-system VLAs [11] separate high-level reasoning from low-level action generation to improve control efficiency. Synchronous dual-system designs run the slow VLM in lockstep with the action expert, treating System 2's output as a per-step conditioning signal [6, 92, 93, 15], which cleanly preserves language grounding but inherits the VLM's per-step latency. Asynchronous variants relax this by running System 2 at a low fixed frequency while System 1 issues actions at a higher rate, and they differ mainly in what System 2 hands to System 1, ranging from conditioning tokens fed forward to a flow-matching specialist (Robodual [10], Fast-in-Slow [14]), to planning tokens for a 3D-aware diffusion policy (OpenHelix [13]), to latent goals refined by an action expert (Hume [94]). All of these still pin the call interval Δ at a single value chosen offline, so they sit in the design space we extend with adaptive scheduling. A single offline Δ is limiting in part because tasks differ in how often they need the generalist (Figure 1a). This parallels multi-task learning, where tasks trained together differ in learning dynamics and gradient behavior, so treating them uniformly is suboptimal and has motivated task-adaptive loss balancing and update rules [95–102]. TUD carries this view to inference time by letting the call frequency adapt to each task and to each phase of a rollout. OneTwoVLA [17] is the closest prior attempt to move beyond a fixed call interval. It learns a decision token that switches between reasoning and acting modes, supervised using manually annotated phase boundaries. This design depends on the heuristic choice of where a phase begins and may therefore limit transfer to unseen scenes. CLOVER [103] also re-invokes its visual planner adaptively, but the trigger is a perceptual distance against a generated subgoal image and requires training a dedicated video-diffusion subgoal generator. Adaptive Action Chunking (AAC) [23] uses action entropy estimated from multiple sampled action chunks to adapt the length of the action chunk executed before replanning. In contrast, TUD measures predictive action-plan dispersion using forwards that the dual-system architecture already performs, and uses this signal to selectively re-invoke the generalist. This directly exposes the generalist call budget as a controllable inference-time trade-off. TUD is also distinct from test-time adaptation, which updates model parameters on unlabeled test inputs to cope with distribution shift [104–106]. TUD keeps all weights fixed and adapts only how often the generalist is invoked during a rollout.

Uncertainty estimation for robotic policies. Uncertainty estimation in robotic policies has been studied at both training and inference time. At training time, iterative imitation learning frameworks based on DAgger [107] use uncertainty to decide when to query an expert. Concrete realizations include ensemble variance over independently trained policies [108, 109], MC-dropout over a single

policy [110], and diffusion-loss comparison against a training-distribution reference [19]. At inference time, uncertainty drives failure detection [111–113] and human-in-the-loop intervention [114, 115]. These methods share a common limitation: they require auxiliary training pipelines, multiple policy models, or repeated stochastic forwards, all of which are at odds with the throughput requirements that motivate dual-system VLAs. Our work targets a signal that avoids all three by computing dispersion from forwards the architecture already runs, and Section 4.3 empirically shows it separates success from failure on VLA-Arena more reliably than three representative prior signals from this family.

B Proof of Proposition 1

We briefly introduce the basic concepts used in the proof to keep the paper self-contained.

Lipschitz continuity. A map $f : \mathcal{X} \to \mathcal{Y}$ between normed spaces is L-Lipschitz if $\|f(x) - f(x')\| \leq L\|x - x'\|$ for all $x, x' \in \mathcal{X}$. We use the Euclidean norm $\|\cdot\|_2$ on observations and action chunks throughout. The assumption is mild, and any neural network with bounded weights is Lipschitz with some finite constant.

We restate the proposition for convenience.

Proposition 1 (Action-deviation bound under Lipschitz specialist). *Suppose the specialist forward $\mathcal{S}(\cdot, h_k)$ is $L_{\mathcal{S}}$-Lipschitz. Then for any $t \in [t_k, t_{k+1})$,*

$$\left\|\tilde{a}_{t_k}[j(t)] - \tilde{a}_t[j(t)]\right\|_2 \;\leq\; \alpha_E\, u_t \;+\; \beta\, L_{\mathcal{S}}\, \rho_{t,k},$$

with $\alpha_E = 2d_a\sqrt{E-1}$, $\beta \in \{0, 1\}$ depending on whether $t - t_k \leq E$, and $\rho_{t,k} = \max_{t' \in [t_k, t]} \|O_{t'} - O_{t_k}\|$.

Proof of Proposition 1. Because every chunk in the buffer is anchored at the same generalist call t_k, slot index $j(t)$ refers to the same upcoming step across all entries. For each buffer entry $i \in \{0, \ldots, E-1\}$, define

$$y_i \;=\; \tilde{a}_{t-i}[j(t), :] \;\in\; \mathbb{R}^{d_a},$$

let $\bar{y} = (1/E)\sum_i y_i$ and let s_d denote the per-coordinate sample standard deviation of $\{y_{i,d}\}_{i=0}^{E-1}$. By Eq. (5) of the main paper,

$$u_t \;=\; \frac{1}{d_a}\sum_d s_d \;=\; \frac{\|s\|_1}{d_a}.$$

We first record a standard deviation-from-sample-mean inequality. For each coordinate d, the identity $\sum_{i'=0}^{E-1}(y_{i',d} - \bar{y}_d)^2 = (E-1)s_d^2$ implies $|y_{i,d} - \bar{y}_d|^2 \leq (E-1)s_d^2$ for every i. Summing across d and taking square roots,

$$\|y_i - \bar{y}\|_2 \;\leq\; \sqrt{E-1}\cdot\|s\|_2 \;\leq\; \sqrt{E-1}\cdot\|s\|_1 \;=\; d_a\sqrt{E-1}\cdot u_t, \tag{7}$$

where the second inequality uses $\|s\|_2 \leq \|s\|_1$ (valid since each $s_d \geq 0$). Define

$$y_t \;:=\; \tilde{a}_t[j(t), :], \qquad y_{t_k} \;:=\; \tilde{a}_{t_k}[j(t), :].$$

Because the buffer is updated by inserting $\tilde{a}_t$ at index 0 at every step, $y_t = y_0$ is always present in the buffer.

Case 1: $t - t_k \leq E$. The anchor entry $\tilde{a}_{t_k}$ has not yet been evicted, so $y_{t_k} = y_{t-t_k}$ is also a buffer entry. Applying Eq. (7) to both y_t and y_{t_k} and using the triangle inequality,

$$\|y_{t_k} - y_t\|_2 \;\leq\; \|y_{t_k} - \bar{y}\|_2 + \|y_t - \bar{y}\|_2 \;\leq\; 2d_a\sqrt{E-1}\cdot u_t.$$

This proves the bound with $\alpha_E = 2d_a\sqrt{E-1}$ and $\beta = 0$.

Case 2: $t - t_k > E$. The anchor entry has been evicted and the oldest still-in-buffer entry is $y_{E-1} = \tilde{a}_{t-(E-1)}[j(t), :]$. Both $\tilde{a}_{t_k}$ and $\tilde{a}_{t-(E-1)}$ are computed via the same cached context h_k, so by the Lipschitz hypothesis,

$$\|\tilde{a}_{t_k}[j(t), :] - \tilde{a}_{t-(E-1)}[j(t), :]\|_2 \;\leq\; \|\tilde{a}_{t_k} - \tilde{a}_{t-(E-1)}\|_2 \;\leq\; L_{\mathcal{S}}\cdot\|O_{t_k} - O_{t-(E-1)}\|_2 \;\leq\; L_{\mathcal{S}}\rho_{t,k},$$

where the last inequality uses the definition $\rho_{t,k} = \max_{t' \in [t_k, t]} \|O_{t'} - O_{t_k}\|$ and that $t - (E-1) \in [t_k, t]$. Combining with Eq. (7) applied to y_{E-1} and y_t, and using triangle inequality:

$$\begin{aligned} \|y_{t_k} - y_t\|_2 &\leq \|y_{t_k} - y_{E-1}\|_2 + \|y_{E-1} - \bar{y}\|_2 + \|y_t - \bar{y}\|_2 \\ &\leq L_{\mathcal{S}} \rho_{t,k} + 2d_a\sqrt{E-1} \cdot u_t. \end{aligned}$$

This proves the bound with $\alpha_E = 2d_a\sqrt{E-1}$ and $\beta = 1$. □

The bound is loose by a factor of $\sqrt{d_a}$ in the worst case where the per-dimension dispersion is concentrated in a single coordinate. Under the more typical regime where per-dimension stds are comparable in magnitude, $\|s\|_2 \approx \sqrt{d_a}\,(\|s\|_1/d_a)$, and the constant tightens to $\alpha_E \approx 2\sqrt{(E-1)d_a}$.

Empirical bound check and interpretation. Proposition 1 bounds the deviation between the cached specialist action and its current re-prediction at the executed action slot, with two complementary terms. The observable dispersion u_t captures disagreement among recent action re-predictions, while $L_{\mathcal{S}}\rho_{t,k}$ accounts for changes in the observation after the cached anchor leaves the prediction buffer. The coefficient $\alpha_E = 2d_a\sqrt{E-1}$ scales the dispersion term with the buffer size, and β activates the observation-drift term only when the anchor $\tilde{a}_{t_k}$ is no longer contained in the buffer. We use $d_a = 7$, corresponding to three translation dimensions, three rotation dimensions, and one gripper dimension.

Increasing E averages the dispersion estimate over a longer prediction history, which can improve temporal stability, but simultaneously increases α_E and therefore does not necessarily tighten the bound. For the buffer sizes considered in our experiments,

Table 6: Effect of the prediction-buffer size E on the dispersion coefficient $\alpha_E = 2d_a\sqrt{E-1}$ with $d_a = 7$.

E	α_E	**Interpretation**
3	19.80	Tighter bound, but noisier dispersion estimate
5	28.00	Default balance
7	34.29	Smoother dispersion estimate, but looser bound

We further evaluate how conservative the bound is on held-out test trajectories. For this diagnostic, we fix $L_{\mathcal{S}}$ to its calibration $p99$ value of 71.65 and compute the bound-to-deviation ratio

$$r_t = \frac{\alpha_E u_t + \beta L_{\mathcal{S}} \rho_{t,k}}{\|\tilde{a}_{t_k}[j(t)] - \tilde{a}_t[j(t)]\|_2}. \tag{8}$$

A ratio of 1 corresponds to a tight bound, whereas values larger than 1 indicate an increasingly conservative margin. Table 7 reports the lower tail of this ratio for several buffer sizes.

Table 7: Empirical bound-to-deviation ratio r_t on held-out trajectories. Larger values indicate a more conservative bound.

E	**Minimum**	**p1**	**p5**	**p10**
3	1.472	1.654	2.140	2.425
5	2.009	2.210	2.834	3.220
7	2.209	2.742	3.278	3.672
9	2.324	3.054	3.624	4.036
11	2.557	3.224	3.801	4.188

For $E \geq 5$, even the minimum ratio exceeds 2, while the p1 ratio exceeds 3 for $E \geq 9$, indicating a substantial conservative margin in the lower tail. We do not use this analysis to select an optimal buffer size. Instead, the bound provides design guidance for limiting both the prediction buffer and prolonged cached-plan reuse. We therefore use the moderate default $E = 5$ together with a finite $\Delta_{\max}$, while the action-chunk horizon $C = 50$ follows the base π_0 configuration.

C Invocation Rule Details

Implementation details. Algorithm 1 summarizes the complete invocation procedure. Unless otherwise stated, we use a prediction-buffer size of $E = 5$, a side-channel measurement period of $\Delta_s = 2$, a decision period of $\Delta = 3$, and a maximum interval without a generalist call of $\Delta_{\max} = 20$. The action-chunk horizon is fixed to $C = 50$ following the base π_0 configuration and is not independently tuned.

Algorithm 1 Invocation rule for adaptive generalist calling.

Require: call interval Δ, threshold θ_p, max steps without a call $\Delta_{\max}$, buffer size E
$t_k \leftarrow 0$, $h_0 \leftarrow \mathcal{G}(I, O_0)$ (Eq. 2), compute $\tilde{a}_0$ via Eq. 3, reset buffer $\mathcal{B}$
for each step $t = 1, 2, \ldots$ **do**
  execute the action drawn from $\tilde{a}_{t_k}$ at slot $j(t)$
  if $t \bmod \Delta_s = 0$, compute $\tilde{a}_t$ via Eq. 3 and update $\mathcal{B}_t$
  update u_t via Eq. 5 from the available buffer
  if $t > 0$ **and** $t \bmod \Delta = 0$ **(decision point) then**
    $\bar{u} \leftarrow$ mean of $\{u_\tau : t_k < \tau \leq t\}$
    if $\bar{u} < \theta_p$ **and** $t - t_k < \Delta_{\max}$ **then**
      skip: keep h_k and $\tilde{a}_{t_k}$
    else
      call: $h_k \leftarrow \mathcal{G}(I, O_t)$, $t_k \leftarrow t$, recompute $\tilde{a}_{t_k}$, reset $\mathcal{B}$

Online threshold adaptation. For each task suite, we initialize a FIFO buffer $\mathcal{H}$ at the first trial and retain it across subsequent trials. The buffer stores up to $N = 128$ observed uncertainty values u_t, discarding the oldest entries when full. At each decision point, θ_p is set to the empirical p-th percentile of the values currently in $\mathcal{H}$. Adaptive invocation is enabled only after at least 16 measurements have been collected. During this warm-up period, the generalist is always invoked. This provides suite-level online adaptation without offline threshold calibration.

Hyperparameter sensitivity. We evaluate the sensitivity of TUD to E, Δ_s, Δ, and $\Delta_{\max}$ by varying each parameter around the default configuration.

Table 8: Hyperparameter sensitivity of the invocation rule. Bold indicates the default setting. Calls denote generalist invocations per episode.

Factor	Setting	SR ↑	Calls ↓
Buffer E	3	0.832	27.3
	5	**0.850**	**21.6**
	7	0.850	18.5
Measurement Δ_s	1	0.848	28.0
	2	**0.850**	**21.6**
	4	0.852	51.2
Decision Δ	2	0.840	29.4
	3	**0.850**	**21.6**
	5	0.793	15.7
Maximum interval $\Delta_{\max}$	12	0.858	24.0
	20	**0.850**	**21.6**
	32	0.830	20.1

Across moderate variations around the default configuration, SR remains relatively stable while the number of generalist calls varies more noticeably. Increasing E reduces calls without degrading SR, whereas overly sparse uncertainty measurements ($\Delta_s = 4$) substantially reduce the call-saving benefit. The decision period Δ and $\Delta_{\max}$ show clearer efficiency–accuracy trade-offs: overly sparse decisions ($\Delta = 5$) or prolonged cache reuse ($\Delta_{\max} = 32$) reduce calls but degrade SR. Overall, the default configuration provides a balanced operating point without requiring narrow hyperparameter tuning.

D Architecture Details

This appendix supplements Section 3.1 with the concrete components of our backbone, the bridge module, and the inference and training pipelines.

D.1 Backbone

We adopt a π_0 [6] flow-matching Vision-Language-Action policy as the backbone. The vision-language model is PaliGemma. Inside the architecture, the observation O_t of Section 2.1 is split into a visual stream and a language plus proprioception stream. The visual stream is encoded by a SigLIP image tower of width 2048 that produces 256 patch tokens per camera image. Language and proprioception are tokenized in the standard PaliGemma format. The action expert has hidden width 1024 and decodes action chunks via flow matching, with chunk horizon $C = 50$ time slots and internal action dimension 32, of which the first $d_a = 7$ dimensions are executed (3D end-effector position, 3D axis-angle rotation, 1D gripper command).

D.2 Q-Former Bridge

The bridge takes two SigLIP feature maps (a current and a prefix one) and returns a small set of conditioning tokens prepended to the action expert's input. Its components are:

- 8 learnable query tokens, initialized at $\mathcal{N}(0, 0.02^2)$.
- Two linear projections that lift the SigLIP features of the two streams to expert width (1024).
- A stack of 2 Q-Former layers. Each layer applies, in order, self-attention on the queries, cross-attention from the queries to the concatenation of the two projected streams (with a learned 2-vector type embedding distinguishing the two slots), and a 2-layer GELU MLP of inner width 2048.
- All attention modules use 8 heads. A final linear projection produces the output tokens.

The concat-cross-attention design lets the queries directly compare the two streams within a single attention map, halving parameter count relative to two separate cross-attentions.

D.3 Generalist Call

At a decision point t_k, given the multimodal observation O_{t_k}, the generalist proceeds as follows.

1. Encode O_{t_k} with SigLIP and run the bridge in self-context mode (the same SigLIP feature is supplied to both bridge inputs).
2. Embed PaliGemma prefix tokens (proprioception, language, image) and run the LLM forward to obtain the cached key-value tensors and prefix mask.
3. Run the flow-matching sampler for 10 Euler steps to denoise an action chunk $\tilde{a}_{t_k} \in \mathbb{R}^{C\times 32}$.

The LLM prefix cache and the SigLIP feature of O_{t_k} together constitute h_k in the notation of Section 3.1. The cost is a full PaliGemma image and text encode plus an LLM prefix forward plus 10 flow-matching steps.

D.4 Specialist Forward

For any $t \in [t_k, t_{k+1})$, given h_k and the latest observation O_t, the specialist proceeds as follows.

1. Encode O_t with SigLIP. The cached LLM key-value tensors inside h_k are reused without re-encoding.
2. Run the bridge in cross-context mode so that the queries attend to both the current and the cached SigLIP features within a single attention pass.
3. Run a single Euler step of the flow-matching sampler conditioned on the new bridge tokens to produce $\tilde{a}_t \in \mathbb{R}^{C\times 32}$.

The cost is a SigLIP encode of the new image plus the bridge plus a single flow-matching step. The expensive PaliGemma LLM prefix together with the language and proprioception encodings is amortized by the cached state in h_k.

D.5 Training

We initialize from the publicly released π_0 pre-training checkpoint. The Q-Former bridge is always trainable, and the rest of the model uses the π_0 default fine-tuning setup. The training objective is the standard π_0 flow-matching loss, augmented with a second observation (the "delta image") used as the bridge's prefix input. When a sample lacks a delta image (around 10% of batches), the bridge falls back to self-context (current = prefix) so the loss is well-defined. We train for $30{,}000$ steps with batch size 32 on $4\times$ NVIDIA H200 GPUs. The optimizer is AdamW [116] with learning rate 2.5×10^{-5}, weight decay 10^{-10}, $\beta_1 = 0.9$, $\beta_2 = 0.95$, and gradient clipping at 1.0. The learning-rate schedule is a $1{,}000$-step linear warmup followed by cosine decay to zero.

E Benchmark and Evaluation Details

E.1 Benchmarks

To rigorously evaluate the long-horizon manipulation capabilities, zero-shot generalization, and robustness of the proposed models, we utilize two comprehensive benchmarking frameworks: VLA-Arena [18] and CALVIN [1].

VLA-Arena [18] is a large-scale, open-source framework formulated to systematically benchmark the capability frontiers and failure modes of Vision-Language-Action (VLA) models. The framework comprises 170 distinct tasks categorized along four orthogonal dimensions: Safety, Distractor, Extrapolation, and Long Horizon. To objectively measure generalization capabilities rather than mere data memorization, tasks are structured across three escalating difficulty levels (L0, L1, and L2). Model fine-tuning is strictly confined to the base-level dataset (L0), with subsequent evaluations conducted in the progressively complex L1 and L2 environments. Furthermore, independent of the structural task difficulty, VLA-Arena introduces systematic perturbations to evaluate model robustness. By incorporating varying degrees of language perturbations (W0–W4) and visual perturbations (V0–V4), the benchmark enables the quantitative separation of an agent's true structural reasoning from memorization.

CALVIN [1] is a simulation benchmark explicitly designed to evaluate language-conditioned policy learning for long-horizon robot manipulation tasks. The dataset features unconstrained natural language instructions and complex action spaces, supporting flexible sensor suite configurations. It is designed to test an agent's ability to map raw visual observations and diverse language commands to continuous sequences of control actions. The primary evaluation protocol within the CALVIN framework assesses zero-shot generalization, measuring a model's ability to successfully execute long-horizon tasks in completely unseen environments and follow novel language instructions that were absent from the training distribution.

Table 9: Comparison of detailed specifications between CALVIN and VLA-Arena benchmarks.

Specification	**CALVIN** [1]	**VLA-Arena** [18]
Input Modalities	RGB, Depth, Proprioception, Language	RGB, Language
Visual Observation	Static RGB ($200 \times 200 \times 3$) Depth Static (200×200) Gripper RGB ($84 \times 84 \times 3$) Depth Gripper (84×84) Tactile Image ($120 \times 160 \times 2$)	Multi-view RGB Cameras (Typically resized to standard inputs, e.g., 224×224 or 256×256)
Action Space	**7-DoF Continuous Control** - Relative EE translation (3) - Relative EE rotation (3) - Gripper state (1, discrete)	**7-DoF Continuous Control** - Absolute EE pos & ori (6) - Gripper state (1, discrete)
Unique Features	**Proprioceptive State:** 15-dim vector (EE pos/ori, joints, gripper) **Control Freq:** 30 Hz	**Perturbation Inputs:** Visual (V0–V4), Language (W0–W4) **Task Diversity:** 170 tasks (L0, L1, L2)

E.2 Protocol

VLA-Arena. The proposed model is evaluated across 4 distinct domains (total 170 tasks), following a structured protocol to ensure robustness and reproducibility.

- **Experimental Design**: Each task is assessed across 10 trials. Success rates for each suite represent the mean performance over its inclusive tasks. In total, every model is subjected to 1,700 trials, covering 170 unique task configurations.
- **Initial State Selection**: To ensure environmental variability, we maintain a pool of 50 predefined initial states for each task. Each of the 10 trials utilizes a distinct starting configuration from this pool.
- **Reproducibility**: The selection of initial states is performed deterministically using a fixed global seed, ensuring a fair and consistent comparison across all evaluated models.

CALVIN. We follow the ABC→D setting, in which the policy is trained on environments A, B, and C and evaluated on the unseen environment D. We evaluate 100 instruction chains of five consecutive sub-tasks, sampled with a fixed seed so that every method is evaluated on the same chains. Each sub-task is allotted at most 360 environment steps, and a chain ends at its first failed sub-task.

E.3 Metrics

VLA-Arena. To quantitatively evaluate the performance and safety of VLA models, we employ two primary metrics: **Success Rate (SR)** and **Cumulative Cost (CC)**.

- **Success Rate (SR)**: This metric represents the logical task completion capability of the model. It is defined as the ratio of successful episodes to the total number of attempted episodes N. An episode is considered successful if the robot fulfills the linguistic instruction within the maximum time steps T.

$$\text{SR} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(\text{success}_i)$$

  where $\mathbb{I}(\cdot)$ is the indicator function.
- **Cumulative Cost (CC)**: This metric quantifies the safety violations encountered during task execution. In safety-constrained environments, a cost c_t is incurred at each time step t whenever a safety constraint (e.g., collision with obstacles, entering hazard zones) is violated. The CC for an episode is the sum of these costs:

$$\text{CC} = \sum_{t=1}^{T} c_t$$

  A lower CC indicates a safer policy, with 0 representing a perfectly safe trajectory.

CALVIN. We report the average length of consecutive tasks completed (0-5) and the success rates for task chains of lengths 1 through 5. These metrics reflect the model's ability to maintain consistency over an extended horizon.

F Baselines

F.1 Model Baselines

To comprehensively evaluate our method, we compare it with multiple state-of-the-art VLA baselines across two distinct benchmarks.

Baselines for VLA-Arena.

- **OpenVLA** [5]: OpenVLA (7B) formulates robotic control as a next-token prediction problem by discretizing or mapping continuous robot actions into the vocabulary space of a large language model. It is built on top of the Prismatic-7B vision-language model [4], which integrates a fused

visual encoder composed of DINOv2 [117] and SigLIP [25] with Llama 2 [118] as the language backbone. The model takes as input a single RGB image and a natural language instruction, and predicts the corresponding continuous robot action.

- **OpenVLA-OFT** [7]: OpenVLA-OFT is an enhanced variant of OpenVLA [5] that improves both computational efficiency and policy performance through an Optimized Fine-Tuning (OFT) recipe. Specifically, it introduces techniques such as parallel decoding and action chunking to make action prediction more efficient while preserving the overall vision-language-action framework. It retains the same base vision-language model architecture and the same input formulation as OpenVLA, taking a single RGB image and a language instruction as input to predict robot actions.
- π_0 [6]: π_0 is built upon the PaliGemma vision-language model [3] and trains an action expert using conditional flow matching [119, 120], enabling the modeling of complex continuous actions. To enable scalability across diverse robot embodiments, cross-embodiment pre-training is performed using a mixture of a dataset directly collected from seven different robot configurations and 68 different tasks, together with the OXE dataset [121], which includes data from 22 robot embodiments. Subsequently, downstream task-specific post-training leads to strong performance even on complex tasks.
- π_0**-FAST** [26]: FAST [26] proposes transforming action sequences into the frequency domain using the discrete cosine transform (DCT) and then tokenizing the transformed sequence. By reducing the high temporal redundancy between adjacent tokens, this approach enables autoregressive learning to work effectively even for high-frequency tasks. Building upon FAST-based action tokenization, π_0-FAST incorporates it into π_0 [6], achieving performance comparable to diffusion-based VLAs while dramatically reducing training time.
- **UniVLA** [8]: UniVLA learns diverse vision-language-action policies across cross-embodiment settings within a single model, not by performing autoregressive learning in the low-level action space, but instead in a latent action space. For task-centric latent action learning, a two-stage training strategy is adopted that separates action latents containing task-irrelevant dynamics from task-centric latent actions. Autoregressive learning is then performed in the task-centric latent action space, enabling generalizable learning across diverse embodiments and video data. For downstream task adaptation, a specific action decoder is used to generate actions conditioned on visual embeddings and task-centric latent action tokens, followed by post-training. Performance is further improved by incorporating historical latent actions as inputs, and the framework is built on the Prismatic-7B vision-language model [4].
- **SmolVLA** [9]: SmolVLA is a framework designed for efficient VLA systems and uses SmolVLM-2 [122], a lightweight VLM composed of a SigLIP [25] vision encoder and a SmolLM2 [123] language decoder. To improve inference efficiency, it applies layer skipping in the VLM and reduces the number of visual tokens to 64 per frame. In addition, its action expert adopts an alternating arrangement of self-attention and cross-attention layers, achieving both high success rates and fast inference, and is trained using conditional flow matching [119, 120, 124].
- **OneTwoVLA** [17]: OneTwoVLA is a unified VLA framework that integrates reasoning and acting within a single model and adaptively switches between the two modes during task execution. It introduces two decision tokens, [BOR] and [BOA], to trigger textual reasoning or action generation, respectively. To train this mode-switching behavior, we follow OneTwoVLA by segmenting robot demonstrations into reasoning and acting intervals and constructing embodied reasoning annotations for critical stages of task execution.

Dual-system baselines for CALVIN.

- **Robodual** [10]: Robodual is a dual-system VLA that pairs a large generalist VLM (System 2) with a lightweight diffusion-based specialist (System 1). The generalist consumes the instruction and a third-person image to produce latent conditioning tokens that are fed to the specialist, which decodes high-frequency action chunks. The two systems run asynchronously with a fixed call interval of $\Delta = 8$ specialist steps per generalist invocation, recovering throughput while retaining language-grounded reasoning.
- **OpenHelix** [13]: OpenHelix follows the dual-system VLA design and provides an open-source reference implementation together with an empirical study of the design space. The slow MLLM produces planning tokens that condition a fast 3D-aware diffusion policy. Inference uses an asynchronous schedule with a default ratio of $\Delta = 10$ policy steps per MLLM invocation.

F.2 Uncertainty Signal Baselines

This subsection gives the formal definitions of the uncertainty signals compared in Section 4.3, together with the adaptations required to apply them in our setting. These methods originate from different settings: GenUQ, MC-AleaUQ, and BayesDiff estimate uncertainty in generative or diffusion models, Diff-DAgger uses diffusion model training loss for uncertainty-aware imitation learning, and AAC-ent. estimates action uncertainty for adaptive action chunking in VLAs. Their original formulations therefore make different assumptions about the model architecture, sampling procedure, and output representation, and cannot be applied verbatim to our π_0-based flow-matching specialist. We adapt each method to operate on the specialist $\mathcal{S}(O_t, h_k)$, which produces an action chunk $\tilde{a}_t \in \mathbb{R}^{C \times d_a}$, while preserving the principle that defines each method. We additionally include a lightweight baseline, visual-feature cosine distance (Visual), which compares the current and cached SigLIP features. The notation follows Sections 2.1 and 3.1.

GenUQ [22]. GenUQ fits a Laplace posterior $q(\theta \mid \mathcal{D})$ over a subset of model parameters θ given the training data $\mathcal{D}$, draws M weight samples $\theta_m \sim q$, runs M forward passes with a shared initial noise vector z (the input to the generative sampler), and quantifies uncertainty by the variability across the M outputs. The original work measures variability through the differential entropy of a Gaussian fit in CLIP feature space,

$$u_t^{\text{GenUQ-orig}} \;=\; \mathcal{V}\Big(\mathcal{N}\big(\, c_\phi(\tilde{a}_t^{(m)}) \,\big|\, \bar{c},\; \mathrm{Diag}(\sigma^2_{\text{emp}}) + \sigma^2 I\big)\Big),$$

where c_ϕ is a CLIP-style image encoder. CLIP, however, is defined on natural images and has no counterpart on action vectors, so we cannot evaluate $c_\phi(\tilde{a}_t^{(m)})$ directly. We replace the CLIP feature extractor with raw action-space dispersion, which preserves the role of $\mathcal{V}(\cdot)$ as the variability statistic. We localize the Laplace fit to π_0's final action projection layer, calibrated once on the training distribution using π_0's flow-matching loss. At test time we fix the initial noise z, draw $M = 4$ weight samples, and sample M action chunks with the full π_0 sampler, one per weight sample, to obtain $\{\tilde{a}_t^{(m)}\}_{m=1}^{M}$. The signal is the per-dimension sample standard deviation averaged over the $C \times d_a$ chunk entries,

$$u_t^{\text{GenUQ}} \;=\; \frac{1}{C\, d_a} \sum_{j=1}^{C} \sum_{d=1}^{d_a} \mathrm{std}_{m=1,\dots,M}\big(\tilde{a}_t^{(m)}[j,d]\big).$$

MC-AleaUQ [21]. MC-AleaUQ is an intra-step Monte-Carlo aleatoric estimate. At a fixed denoising step $t^\star$ along the sampler trajectory, the model's predicted clean output $\hat{x}_0$ is recovered, re-noised M times with independent Gaussian samples, and the per-coordinate variance of the resulting one-step predictions is reported. The original work targets pixel-space diffusion with a discrete-timestep DDPM sampler, while π_0 uses continuous-time flow matching, in which a learned velocity field v_θ transports an initial noise to the clean action chunk along a sampler trajectory parameterized by flow-matching time $\tau \in [0, 1]$ (distinct from environment time t). Let a_τ denote the noised action chunk on this trajectory and $v_\tau = v_\theta(a_\tau,\, h_k,\, \tau)$ the predicted velocity, where h_k is the cached context of Section 3.1. We translate the original procedure using the flow identity $\hat{x}_0 = a_\tau - \tau \cdot v_\tau$ at sampler step $t^\star = 5$, then re-sample

$$x_\tau^{(m)} \;=\; \tau\, \epsilon_m + (1 - \tau)\, \hat{x}_0, \qquad \epsilon_m \sim \mathcal{N}(0, I), \quad m = 1, \dots, M,$$

with $M = 10$, where each $x_\tau^{(m)}$ is a re-noised version of $\hat{x}_0$ at flow-matching time τ. We evaluate the velocity field at each re-noised sample, $v^{(m)} = v_\theta(x_\tau^{(m)},\, h_k,\, \tau)$, and report the per-coordinate sample variance averaged over the $C \times d_a$ chunk entries,

$$u_t^{\text{MC-AleaUQ}} \;=\; \frac{1}{C\, d_a} \sum_{j=1}^{C} \sum_{d=1}^{d_a} \mathrm{Var}_{m=1,\dots,M}\big(v^{(m)}[j,d]\big).$$

The clean output $\hat{x}_0$ acts as the anchor across the M samples, so the signal isolates variance that comes from re-noising rather than from observation perturbation.

Diff-DAgger [19]. Diff-DAgger uses the diffusion training loss itself, evaluated at the policy's deterministic prediction, as the uncertainty score. The original work enumerates the full discrete grid

of training timesteps and noise samples (τ, ϵ), where τ is the flow-matching time of the noised chunk and ϵ the Gaussian noise sample drawn during training. In π_0, however, the flow-matching schedule uses a continuous timestep with $\tau \sim \mathrm{Beta}(1.5, 1)$, so a full enumeration is not defined. We replace the enumeration with $N = 4$ independent draws from the same training distribution,

$$u_t^{\text{Diff-DAgger}} \;=\; \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}_{\text{flow}}\big(\tilde{a}_t,\, h_k;\, \tau_n,\, \epsilon_n\big), \qquad \tau_n \sim \mathrm{Beta}(1.5, 1),\;\; \epsilon_n \sim \mathcal{N}(0, I),$$

where $\tilde{a}_t$ is the specialist chunk prediction (Eq. 3), h_k the cached context, and $\mathcal{L}_{\text{flow}}$ the standard flow-matching MSE between the predicted and target velocity at the noised chunk, evaluated through π_0's own training loss. $N = 4$ matches the effective sample budget of the original discrete enumeration.

BayesDiff [20]. BayesDiff estimates predictive uncertainty in diffusion models by placing a Bayesian approximation on the denoiser and propagating the resulting uncertainty through the reverse generation process. To adapt it to π_0, we fit a diagonal Laplace posterior to the weight and bias of the final action projection layer, while keeping all preceding parameters deterministic. Let Φ denote the hidden features corresponding to the action tokens and let W, b be the posterior mean parameters of the final projection. At each flow-matching step, we compute the predictive mean and variance as

$$v_\mu = \Phi W + b, \qquad v_{\text{var}} = \Phi^{\odot 2}\,\mathrm{Var}[W] + \mathrm{Var}[b],$$

where the variance is computed element-wise under the diagonal posterior approximation. We then propagate both the action mean and variance through the Euler integration steps,

$$x \leftarrow x + \Delta t\, v_\mu, \qquad \mathrm{Var}[x] \leftarrow \mathrm{Var}[x] + \Delta t^2 v_{\text{var}}, \qquad \Delta t = -1/T,$$

starting from $\mathrm{Var}[x] = 0$, where $T = 10$ is the number of Euler steps. Unlike the single-step specialist forward (Appendix D), this requires a full T-step sampling run with the Laplace-equipped projection, that is, T additional velocity evaluations per measurement. The variance update assumes independent uncertainty contributions across Euler steps. The resulting uncertainty score is obtained by averaging the propagated variance over the executed action dimensions at the relevant action slot,

$$u_t^{\text{BayesDiff}} = \frac{1}{d_a}\sum_{d=1}^{d_a} \mathrm{Var}[x]_{j(t),d}.$$

AAC-ent. [23]. Adaptive Action Chunking (AAC) uses action entropy as an inference-time estimate of action uncertainty to adapt the executed action-chunk length. We denote its action-entropy signal by AAC-ent. We retain only AAC's uncertainty-estimation mechanism and repurpose it as a scalar signal for generalist invocation, without adopting its adaptive chunk-size selection. Following AAC, we sample $N = 20$ candidate action chunks and compute uncertainty separately for translation, rotation, and gripper actions. For the continuous translation and rotation components, we compute the Gaussian differential entropy, while discrete gripper uncertainty is measured using categorical entropy. For each chunk position i, the corresponding entropy is

$$E^i_{\text{trans/rot}} = \frac{1}{2}\log\big[(2\pi e)^{d_c}\det(\Sigma_i)\big], \qquad E^i_{\text{grip}} = -\sum_{a\in\mathcal{A}} p_i(a)\log p_i(a), \tag{9}$$

where d_c is the number of continuous action dimensions, Σ_i is their empirical covariance across the N samples, and $\mathcal{A}$ denotes the discrete gripper action space. We use the entropy at the slot $j(t)$,

$$u_t^{\text{AAC}} = E^{j(t)}_{\text{trans}} + E^{j(t)}_{\text{rot}} + E^{j(t)}_{\text{grip}}, \tag{10}$$

where $j(t) = \min(t - t_k, C - 1)$ follows the action-slot definition in Section 3.2.

Visual. Visual measures how much the current observation differs from the visual context cached at the most recent generalist call. Let $F_t = f_{\text{vis}}(O_t) \in \mathbb{R}^{N_v \times d_v}$ denote the SigLIP visual-token features for the current observation, and let F_{t_k} denote the cached visual-token features from the last generalist call. We obtain a pooled visual feature by averaging over the N_v tokens,

$$f_t = \frac{1}{N_v}\sum_{i=1}^{N_v} F_{t,i}, \qquad f_{t_k} = \frac{1}{N_v}\sum_{i=1}^{N_v} F_{t_k,i}.$$

We then define the Visual score as the cosine distance between the two pooled features,

$$u_t^{\text{Visual}} = \max\left(0,\, 1 - \frac{f_t^\top f_{t_k}}{\|f_t\|_2 \|f_{t_k}\|_2 + \epsilon}\right),$$

where $\epsilon = 10^{-8}$ is used for numerical stability. Values near zero indicate little visual change since the last generalist call, while larger values indicate greater change.

Inference cost. Table 10 compares the extra computation that each signal requires per uncertainty measurement.

Table 10: Extra computation per uncertainty measurement in our action-chunk setting.

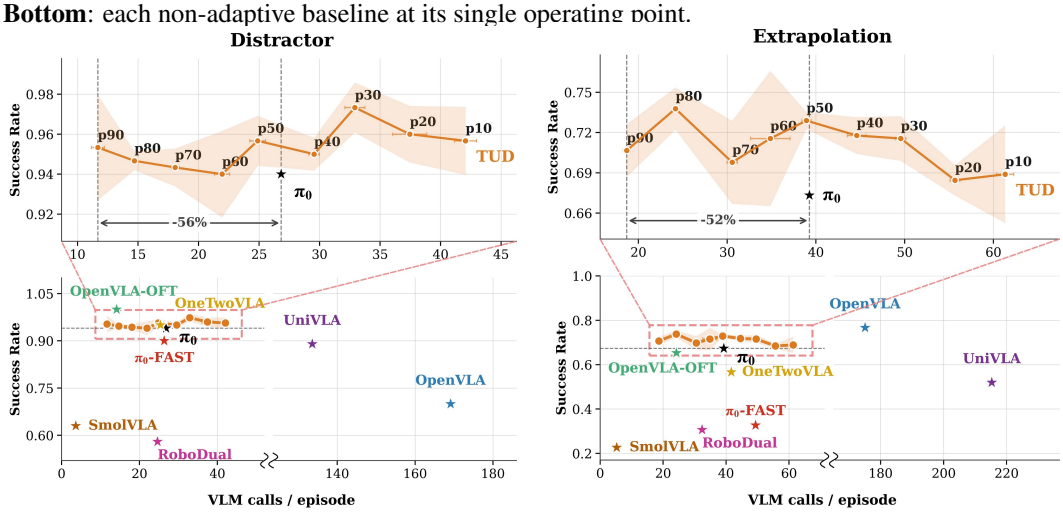
Method	Additional model evaluations	Auxiliary computation
TUD (u_t, ours)	None	std over E buffered chunks
GenUQ	$M = 4$ sampled action chunks	last-layer Laplace weight samples
MC-AleaUQ	$M = 10$ velocity evaluations	re-noising at $t^\star = 5$
Diff-DAgger	$N = 4$ loss evaluations	(τ, ϵ) draws, $\tau \sim \text{Beta}(1.5, 1)$
BayesDiff	$T = 10$ velocity evaluations	variance propagation over T Euler steps
AAC-ent.	$N = 20$ sampled action chunks	action entropy
Visual	None	cosine distance of pooled SigLIP features

G Additional Results

G.1 VLA-Arena Level 0

Similar to the analysis in Figure 3, we conduct extended experiments on the Distractor and Extrapolation settings of VLA-Arena Level 0 in Figure 6. These results examine whether the cost–success trade-off observed in the main paper also holds under visual distractions and extrapolative task conditions, beyond the Safety and Long-Horizon domains. Across both domains, TUD achieves a more favorable cost–success trade-off than the non-adaptive baselines. Compared with π_0, it reduces VLM calls by 56% on Distractor and 52% on Extrapolation while maintaining comparable or higher success rates.

Figure 6: Cost–success trade-off on VLA-Arena level 0. **Left**: Distractor. **Right**: Extrapolation. **Top**: TUD's operating curve as the percentile p varies (each marker = one p, e.g., p70 for $p = 0.70$). **Bottom**: each non-adaptive baseline at its single operating point.



G.2 VLA-Arena Level 1 (Generalization)

Table 11 reports VLA-Arena Level 1 results, where the model is fine-tuned only on the L0 split and evaluated on the more challenging L1 split. Figure 7 further evaluates the cost–success trade-off

under this generalization setting. Across most domains, TUD achieves more favorable performance–efficiency trade-offs than the existing VLA-Arena baselines.

Table 11: Per-task results on VLA-Arena **Level 1** (Generalization). We report Success Rate (SR) and Cumulative Cost (CC). TUD (p70) is the operating point at $p = 0.70$ of the online invocation rule (defined in Section 3.3) and TUD (Best) is the per-task best operating point.

	Baselines								Ours (TUD)	
Task	**OpenVLA**	**OpenVLA-OFT**	**π_0**	**π_0-FAST**	**UniVLA**	**SmolVLA**	**OneTwoVLA**	**RoboDual**	**p70**	**Best**
Safety (SR / CC)										
StaticObstacles	0.60 / 8.2	0.20 / 45.4	0.74 / 8.0	0.40 / 56.0	0.42 / 9.7	0.00 / 8.8	0.58 / 9.3	0.40 / 13.7	0.74 / 14.8	0.77 / 19.0
CautiousGrasp	0.40 / 120.2	0.50 / 6.3	0.06 / 16.4	0.06 / 15.6	0.60 / 32.1	0.28 / 30.7	0.26 / 23.8	0.18 / 48.4	0.16 / 11.8	0.28 / 13.2
HazardAvoidance	0.02 / 22.8	0.00 / 22.9	0.00 / 16.8	0.00 / 15.4	0.12 / 18.3	0.00 / 19.5	0.12 / 18.9	0.14 / 18.9	0.06 / 18.9	0.11 / 18.7
StatePreservation	0.06 / 16.0	0.76 / 7.6	0.64 / 6.4	0.56 / 5.6	0.76 / 7.6	0.18 / 1.8	0.66 / 6.6	0.38 / 3.8	0.59 / 5.9	0.67 / 6.7
DynamicObstacles	0.60 / 5.1	0.56 / 3.7	0.64 / 3.3	0.30 / 8.8	0.58 / 16.3	0.24 / 16.6	0.68 / 9.3	0.30 / 9.3	0.71 / 12.0	0.79 / 11.4
Distractor (SR)										
StaticDistractors	0.20	0.00	0.02	0.22	0.12	0.00	0.26	0.00	0.20	0.22
DynamicDistractors	0.58	0.54	0.70	0.29	0.54	0.30	0.64	0.32	0.70	0.75
Extrapolation (SR)										
PrepositionCombinations	0.04	0.18	0.10	0.00	0.02	0.00	0.00	0.08	0.04	0.05
TaskWorkflows	0.20	0.00	0.00	0.00	0.04	0.04	0.00	0.26	0.32	0.36
UnseenObjects	0.60	0.40	0.52	0.00	0.76	0.18	0.38	0.18	0.65	0.66
Long Horizon (SR)										
LongHorizon	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.05	0.05

G.3 Extended Temporal Dynamics of Uncertainty across Diverse Rollouts

To complement the long-horizon rollout shown in Figure 5 of the main paper, we provide four additional rollouts in Figure 8 that cover the remaining VLA-Arena domain families: two extrapolation episodes (one failure, one success), one safety-critical episode and one distractor episode. From top to bottom, the figure shows the preposition-combinations failure, the unseen-objects success, the cautious-grasp success, and the static-distractors success. The plotting convention is identical to the main paper: z-normalized u_t over environment time with $K = 5$ keyframes (highest-u_t peaks plus the rollout endpoints) and the corresponding RGB frames in time order beneath each curve.

Extrapolation. For the preposition-combinations failure *"Pick the tomato in the top layer of the drawer and place it on the bowl between the vase and the teapot"*, the gripper hovers above the drawer without grasping the target and then retracts, and the corresponding u_t trajectory exhibits a sustained spike around the failed grasp attempt.

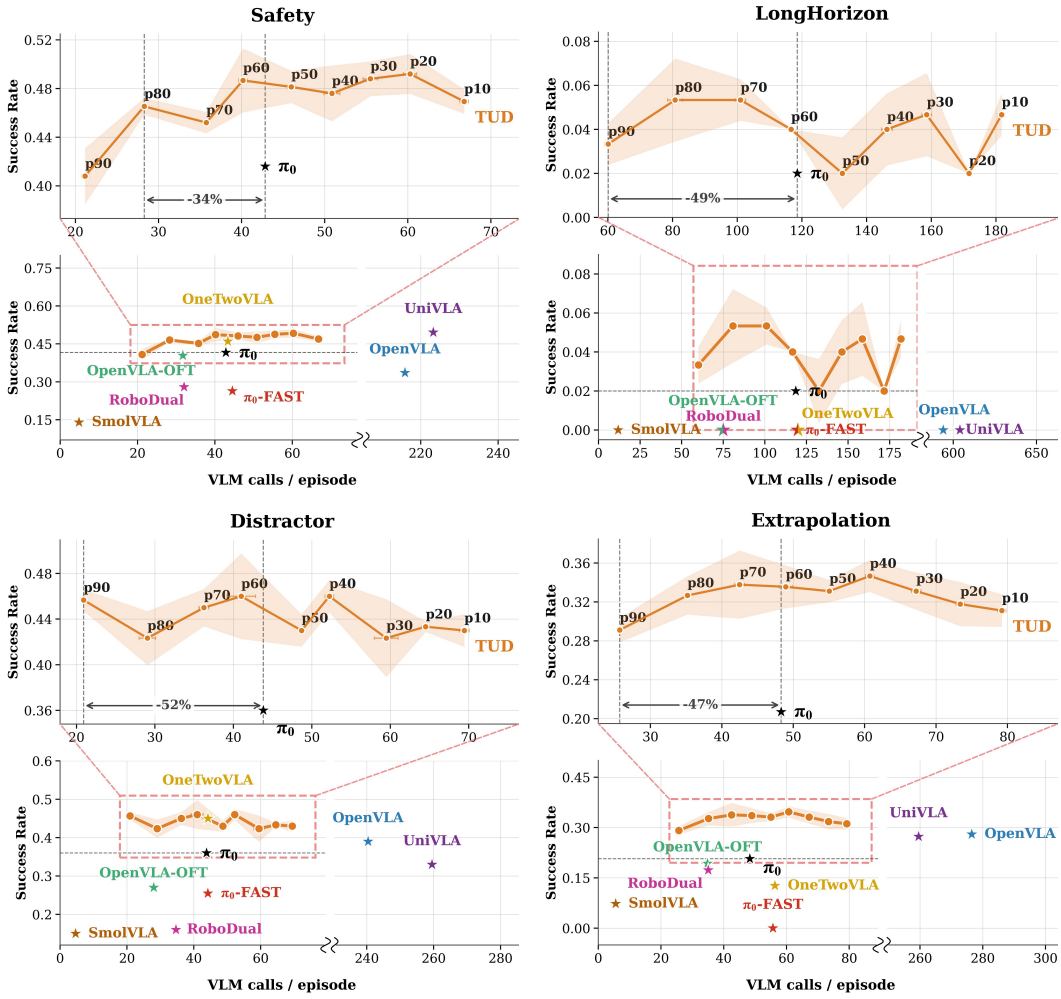
The success of the unseen-objects "*Pick up the onion and place it in the box"* shows the opposite shape: u_t produces brief, localized spikes at first contact and at release, and stays low during the long carry phase.

Safety. For the cautious-grasp success *"Pick up the scissors and place it on the top of the cabinet"*, the gripper aligns above the target before each contact event (once before grasp, once before placement), and u_t spikes at exactly those alignment moments.

Distractor. For the static-distractors success *"Pick the mango on the table and place it on the bowl"*, u_t stays low during simple translations such as approaching the object or transporting it after the grasp, and rises sharply during the grasp itself where fine-grained alignment is required.

Across these rollouts the same pattern recurs: u_t stays low during translation-only motion and rises in moments that require precise alignment or recovery from a failed interaction. Failure cases produce sustained or irregular peaks around the unsuccessful attempt, while successes produce brief, localized spikes at contact events.

Figure 7: Cost–success trade-off on VLA-Arena level 1 across four domains (Safety, Long-Horizon, Distractor, and Extrapolation). **Top**: TUD's operating curve as the percentile p varies (each marker = one p, e.g., p70 for $p = 0.70$). **Bottom**: each non-adaptive baseline at its single operating point.



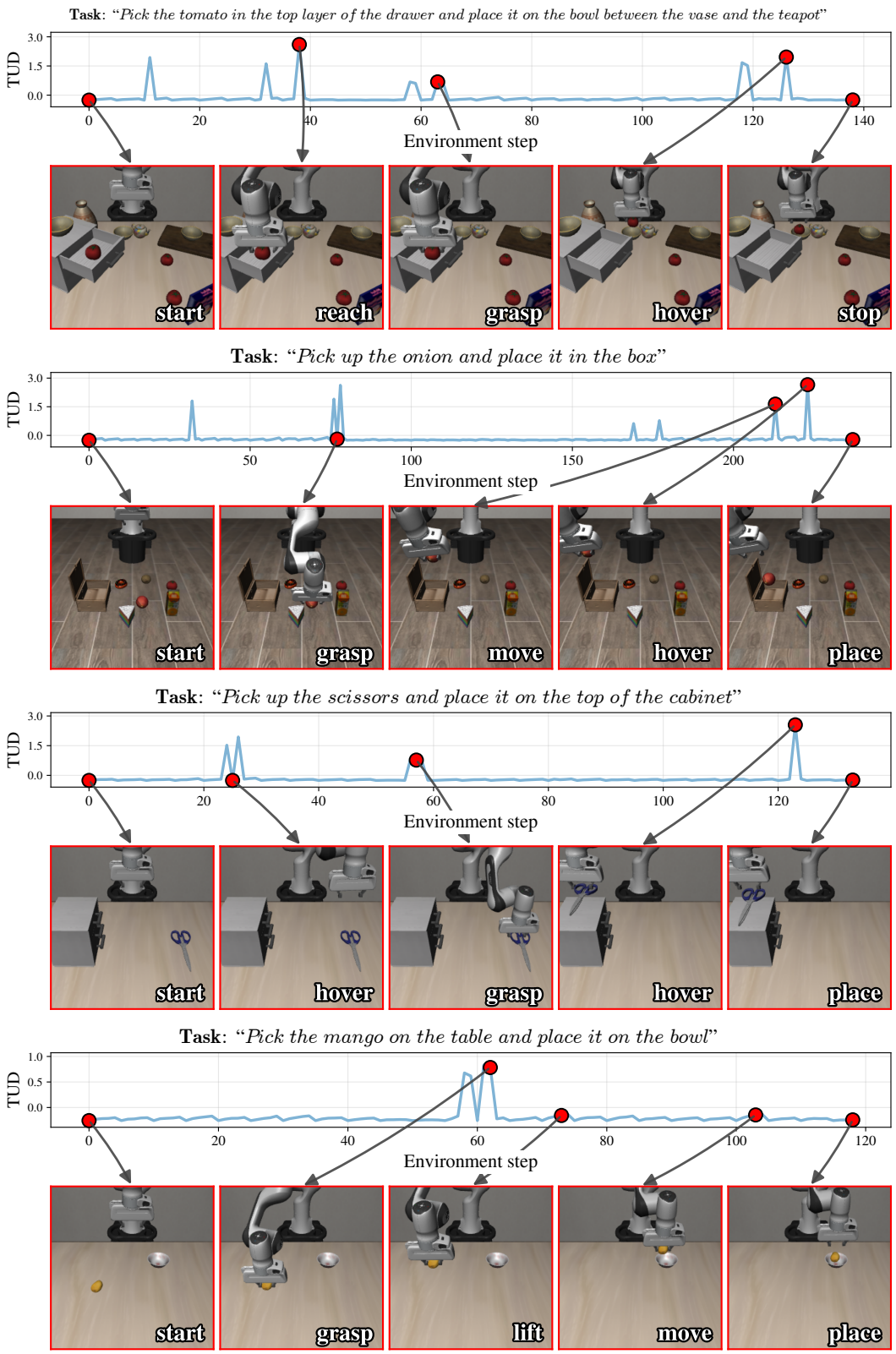


Figure 8: **Additional per-rollout u_t trajectories on VLA-Arena.** From top to bottom: extrapolation (preposition combinations) failure, extrapolation (unseen objects) success, safety (cautious grasp) success, and distractor (static distractors) success. Same plotting convention as Figure 5.

G.4 CALVIN ABC→D Full Comparison

Figure 9 reports the CALVIN ABC→D evaluation results referenced in Section 4.2. The orange curve shows our method (TUD) evaluated across a sweep of the percentile p. Blue and green stars denote two open-source fixed-frequency baselines: OpenHelix and Robodual. TUD reaches a longer average length than both fixed-frequency baselines while using fewer VLM calls per sub-task, suggesting that adapting the generalist call frequency also carries over to long-horizon chains of sub-tasks.

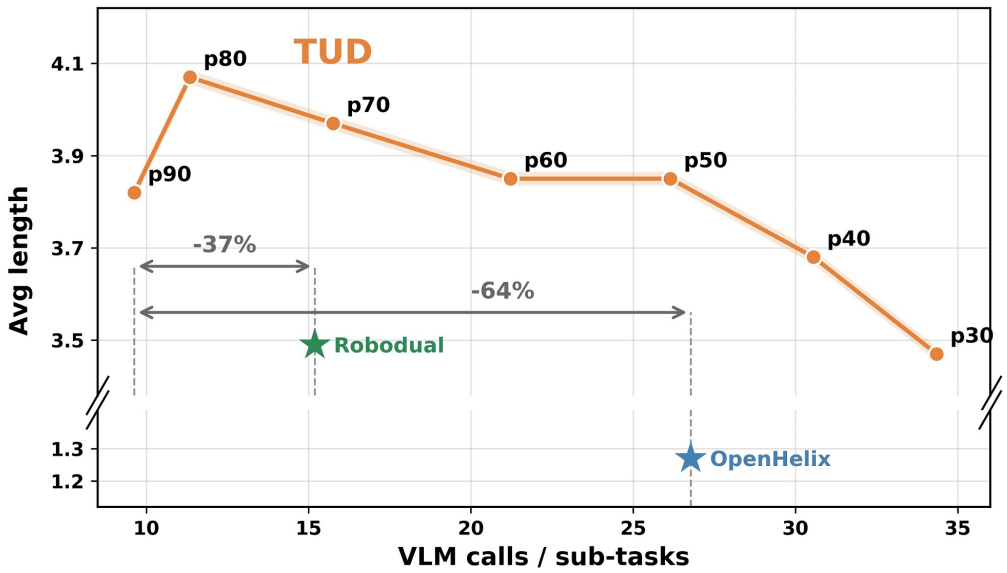


Figure 9: Cost-success trade-off on CALVIN ABC→D evaluation (100 sequences, max 360 steps, each chaining 5 sub-tasks). The x-axis shows mean VLM calls per sub-task; the y-axis shows completed sub-tasks per chain (out of 5).

H Real Robot Experiment

H.1 Setup

As shown in Figure 10, the platform described in Section 4.1 is an SO-101 6-DoF arm robot with two RGB cameras: a third-person view and a gripper view. Models were trained on a single H200 GPU, and inference was performed on a TITAN RTX. We collected data, trained, and evaluated on a total of three task instructions: *"Put the pen into the white bucket"*, *"Close the bottom drawer"*, and *"Stack the small cube on the big cube"*. Figure 11 shows the test environments for each task. For training data, we collected 100 tele-operated demonstrations per task while varying the object positions, and for evaluation, each method was evaluated with 10 rollouts per task across two object positions.

This section expands the real-robot evaluation of Section 4.4. Table 5 reports the average Success Rate, Calls/ep, and Wall-clock over the three tasks, while here we report the per-task breakdown, the end-to-end latency, and the control-loop jitter, all measured on the same rollouts.

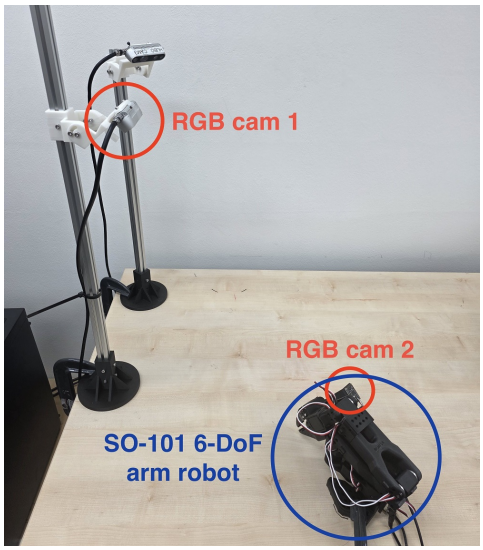


Figure 10: **Real robot setup.** SO-101 6-DoF arm with third-person and gripper-view RGB cameras.

Figure 11: **Test environments.** Evaluation scenes for the three task instructions, with two object positions per task.



(a) "Put the pen into the white bucket"



(b) "Close the bottom drawer"



(c) "Stack the small cube on the big cube"

H.2 Per-Task Breakdown

Table 12 is the per-task performance breakdown, and it shows that the gains in performance and call count are not carried by one easy task. TUD p90 achieved the highest performance on all three task instructions, reaching 1.0 SR on the pen task with 6.4 calls against 42.4 for π_0, and lifting the stack task, on which every baseline was weakest and showed a low SR, to 0.8.

H.3 End-to-End Latency and Control-Loop Jitter

Latency of generalist calls and specialist forwards. On the TITAN RTX, a generalist call takes 1.0–1.2 s, while a specialist forward takes approximately 87 ms. We report image-to-action delay, measured from camera-frame acquisition to transmission of the corresponding

Table 13: **Latency.** mean $\pm$ SD in ms.

Path	Image→action	Cycle
Generalist	1077 ± 109	1107 ± 104
Specialist	94 ± 3	120 ± 3

Table 12: **Real robot comparison per task.** Metrics are the same as in Table 5. SR is the success rate; Calls and Wall are the generalist (VLM prefix) call count and the wall-clock time in seconds, both per episode. Each per-task entry is over 10 rollouts; the *mean* columns average the three tasks and reproduce the values in Table 5.

	pen			**drawer**			**stack**			**mean**		
Method	SR	Calls	Wall (s)	SR	Calls	Wall (s)	SR	Calls	Wall (s)	SR	Calls	Wall (s)
Single-system π_0	0.5	42.4	69.8	0.5	47.8	76.7	0.3	56.8	92.0	0.43	49.0	79.5
Fixed $\Delta = 9$	0.5	46.8	73.7	0.5	38.9	62.6	0.4	59.7	93.0	0.47	48.5	76.4
Fixed $\Delta = 25$	0.4	36.1	79.8	0.6	25.6	56.3	0.3	42.2	92.2	0.43	34.6	76.1
Fixed $\Delta = 50$	0.4	25.8	79.0	0.6	19.5	58.3	0.2	32.5	100.1	0.40	25.9	79.2
TUD p50	0.9	19.6	38.9	0.7	18.7	50.6	0.7	35.3	70.6	0.77	24.5	53.4
TUD p70	0.8	18.6	43.2	0.6	22.4	65.0	0.8	21.3	52.2	0.73	20.8	53.5
TUD p90	**1.0**	**6.4**	**22.6**	**0.7**	**15.2**	**48.1**	**0.8**	**14.7**	**44.9**	**0.83**	**12.1**	**38.5**

robot command, and control-cycle time, measured between consecutive commands. Table 13 reports both measurements separately for the two execution paths.

Control-loop jitter by method. To quantify control-loop jitter under adaptive generalist calling, we aggregate image-to-action latency and control-cycle duration over the same real-robot rollouts for each method. As shown in Table 14, increasing the fixed invocation interval reduces both average latency and jitter, but does so by reusing the cached generalist context for longer periods. TUD achieves a similar reduction adaptively. In particular, TUD p90 attains image-to-action latency and control-cycle variability comparable to fixed $\Delta = 50$, while achieving substantially higher task success. This suggests that adaptive invocation can retain the timing benefits of infrequent generalist calls without committing to a uniformly long reuse interval throughout the rollout.

Table 14: **Control-loop jitter** (ms).

Method	**Image→action** mean / p95	**Cycle** mean ± SD
π_0	70 / 1076	95 ± 256
Fixed $\Delta = 9$	115 / 1180	136 ± 321
Fixed $\Delta = 25$	60 / 96	82 ± 200
Fixed $\Delta = 50$	39 / 94	62 ± 140
TUD p50	59 / 100	81 ± 202
TUD p70	48 / 95	71 ± 175
TUD p90	**38 / 95**	**59 ± 131**

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: Stated in the Abstract and Section 1.

   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: Discussed in Section 5.

   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: [Yes]

Justification: Proposition 1 is stated in Section 3.3 with the full proof in Appendix B.

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: Setup in Section 4.1; architecture, training, and benchmark details in Appendices D and E; real-robot experimental details in H.

   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: [No]

Justification: Our code is not yet publicly available, but we plan to release it. The benchmarks (VLA-Arena, CALVIN) and the π_0 checkpoint are publicly available, and the implementation details necessary to reproduce results are provided in Appendix D.

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: Section 4.1 and Appendix D (Training subsection), with real-robot settings in H.1.

   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: Cost–success curves in Section 4.2 and per-call latency in Table 3 are averaged over multiple seeds and repetitions.

   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: Training compute in Appendix D (Training); per-call latency in Section 4.2; real-robot compute resources and latency measurements in H.3.

   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: The research conforms to the NeurIPS Code of Ethics and evaluates the proposed method on simulated robotic benchmarks and a real robotic arm.

   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: [N/A]

    Justification: The paper studies an inference-time scheduling rule for dual-system VLAs on simulated benchmarks and a real robot, with no direct societal-impact considerations beyond those of the underlying VLA backbone.

    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: The paper does not release any high-risk pretrained model or scraped dataset.

    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: All existing assets (π_0, PaLI-Gemma, SigLIP, VLA-Arena, CALVIN, baseline models) are cited in Section 4.1 and Appendices D–F.

    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: The paper does not release new datasets or models.

    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: The paper does not involve crowdsourcing or human subjects.

    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: The paper does not involve human subjects.

    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: [N/A]

    Justification: LLMs are used only as a standard backbone (PaLI-Gemma in π_0) and not as an original or non-standard methodological component.

    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.