CVPR #29

Decoupling Action Position from Inference Frequency in Stale-Observation VLA Deployment

Anonymous CVPR submission

Paper ID 29

Abstract

Vision-Language-Action (VLA) models face observation latency during deployment, where the standard response is to shrink the action chunk size. We show that this conflates two distinct factors: how often the model re-queries, and which position inside the predicted chunk is executed. We treat these as independent parameters in OpenVLA-OFT and find that, under matched compute budgets, selecting a staleness-appropriate position outperforms the chunk-size-constrained position by 32–96 percentage points on LIBERO-Spatial (∼132K simulated episodes). The effect is task-dependent. Late positions degrade performance on LIBERO-Spatial but remain reliable on LIBERO-Object. Within the setting we study, execution position is a separate design dimension that the chunk-size interface does not expose.

1. Introduction

Vision-Language-Action (VLA) models [1–5] have become a standard approach to general-purpose robot control. Following ACT [7] and Diffusion Policy [8], they predict a chunk of K actions from a single observation and execute them open-loop before the next inference call. Chunking amortizes the cost of running a large model and is now a standard component of the VLA pipeline.

Real deployments, however, do not run at zero latency. A 7B-parameter VLA takes hundreds of milliseconds per inference call [11, 15], and network delays and asynchronous control loops add more on top. By the time a chunk reaches the robot, the observation it was conditioned on is already stale. A common response is to shrink the chunk size c, under the intuition that smaller chunks re-query more often and are therefore more robust.

This intuition implicitly conflates two distinct factors. The chunk size c controls both how many actions are executed before the next inference and which positions inside the predicted chunk are used. Shrinking c changes both at once, and the standard interface in OpenVLA-OFT does not separate these two factors. When a small chunk works better under latency, it is unclear whether the gain comes from re-querying more often or from picking different positions in the chunk.

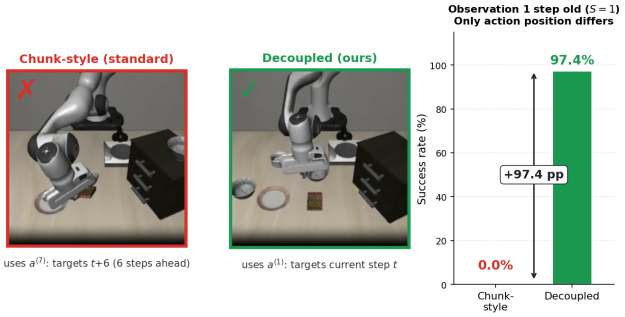

Figure 1. **Decoupling action position under staleness.** At S=1 and e=2, standard chunking executes misaligned positions $(a^{(6)}, a^{(7)})$. Selecting temporally aligned positions $(a^{(1)}, a^{(2)})$ at identical compute raises success from 0.8% to 97.2%. No retraining required.

We decouple these factors by defining an action offset (the starting position within the chunk) and an execution count (the number of actions executed per inference) as independent parameters. The standard protocol is a constrained special case that always starts from a fixed position and executes the entire selected suffix. With offset and execution count decoupled, we can fix the compute budget and vary the starting position independently.

We evaluate this design on OpenVLA-OFT with two LIBERO task suites under controlled observation staleness (Figure 1 shows a representative case). At matched compute, choosing the offset based on the current staleness beats the standard protocol by 32 to 97 percentage points in success rate on LIBERO-Spatial, and the gap follows directly from the temporal training target of each chunk position. We further find that chunk-internal reliability is task-dependent: late positions degrade on LIBERO-Spatial but not on LIBERO-Object. Within the setting we study, these

CVPR #29

results suggest that the position used inside a chunk should be considered an explicit design choice, not a side effect of chunk size.

2. Related Work

Chunking under latency. ACT [7] introduced action chunking for non-Markovian behavior, and Diffusion Policy [8] and recent VLAs [2–6] use the same scheme. These methods do not model observation staleness; inference latency is noted as an engineering issue but is not studied. We treat staleness S as a controlled variable.

Mitigations at the chunk level. Another line of work attacks latency through scheduling. RTC [11] generates the next chunk asynchronously while the current one runs. BID [12] trades off consistency and reactivity at chunk boundaries. LNOB [14] corrects a precomputed chunk against the latest observation. SmolVLA [15] targets per-chunk cost. In all of these the chunk is the unit of analysis. We ask a different question: given staleness S, which position inside the chunk should be executed?

Per-position reliability. ACT's temporal ensembling [7] averages predictions from overlapping chunks at the same target timestep. This assumes that all positions in a chunk are equally reliable estimates of the same timestep. Later chunked policies inherit this assumption and tune chunk length as a single scalar. We test it by varying the offset k at fixed S and measuring reliability along two axes at once: the position inside the predicted chunk, and the timestep being targeted.

3. Method

3.1. Preliminaries

We consider a Vision-Language-Action (VLA) model that operates in a discrete-time control loop. At each control step t, the model receives a visual observation o_t and a proprioceptive state p_t, and predicts a chunk of K future actions:

$$\mathbf{a}_t = (a_t^{(0)}, a_t^{(1)}, \ldots, a_t^{(K-1)})$$

where $a_t^{(j)} \in \mathbb{R}^7$ is supervised as the action for environment step $t+j$, and we set $K = 8$ following OpenVLA-OFT [3].

In practice, visual observations are subject to latency from image acquisition and processing. We formalize this as observation staleness $S \geq 0$: at control step t, the model receives image o_{t-S} while proprioception p_t remains current. This reflects the common deployment scenario where proprioceptive sensing operates at sub-millisecond latency while camera capture and transmission introduce 100–500 ms of delay [11]. At LIBERO's 20 Hz control frequency, each unit of S corresponds to 50 ms.

3.2. Chunk-Based Protocol

The standard deployment interface exposes a single hyperparameter, the execution chunk size $c \in \{1, \ldots, K\}$. In OpenVLA-OFT [3], c dictates the execution of the chunk suffix $a_t^{(K-c)}, \ldots, a_t^{(K-1)}$ before re-querying. This fixes the *action offset* $k = K - c$ (the starting index within the chunk) and the *execution count* $e = c$ (the number of steps per inference call). This convention contrasts with ACT [7] and Diffusion Policy [8], which execute the chunk prefix ($k = 0$, $e = c$). Under OFT, reducing c shifts execution toward later indices that target the farther future. In ACT, k remains fixed at 0. The common observation that "smaller chunks are more robust to staleness" [11] conflates increased re-querying frequency with shifts in action positioning. The standard interface in OpenVLA-OFT does not allow these effects to be isolated.

Protocol	k	e	Notes
ACT-style (prefix)	0	c	Discards $a^{(c)}, \ldots, a^{(K-1)}$
OFT-style (suffix)	$K-c$	c	Discards $a^{(0)}, \ldots, a^{(K-c-1)}$
Decoupled (ours)	free	free	Any (k, e) with $k + e \leq K$

Table 1. **Three deployment protocols.** The first two are special cases of the third. Both standard protocols couple k and e through a single c; the decoupled protocol treats them independently.

3.3. Decoupled Protocol

To disentangle these effects we treat k and e as independent parameters. At each re-query step t, the model predicts $\mathbf{a}_t$ based on (o_{t-S}, p_t), after which the agent executes $a_t^{(k)}, a_t^{(k+1)}, \ldots, a_t^{(k+e-1)}$ and advances $t \leftarrow t + e$. Setting $k = K - c$ and $e = c$ recovers the standard chunk-size protocol in OpenVLA-OFT, while other (k, e) pairs access slices of the predicted chunk that the standard interface does not expose. Table 1 summarizes how the standard protocols sit inside this larger space.

A key property is *budget parity*: for a fixed e, both the standard and decoupled protocols invoke the model every e steps. The inference frequency and total per-episode compute are therefore identical, so any performance difference observed at matched e can be attributed to the choice of action offset k rather than to additional computation.

Under staleness S, the action $a_t^{(k)}$ is trained to act at absolute time $(t-S) + k$, which coincides with the current time t when $k = S$. This is the basis for choosing k to match the expected staleness.

3.4. Experimental Design

We evaluate on LIBERO [9], a simulated tabletop manipulation benchmark with a 7-dimensional action space (3D position, 3D rotation, 1D gripper). We use the LIBERO-Spatial and LIBERO-Object suites, each containing 10

CVPR #29

tasks. For every condition we run 50 trials per task (500 episodes), with seed = 7 and center-crop preprocessing. An episode succeeds if the goal predicate holds for at least 10 consecutive steps.

Our experiments cover three regimes. The chunk-based sweep varies $c \in \{1, \dots, 8\}$ and $S \in \{0, \dots, 10\}$ on Spatial under the standard OFT protocol (∼40K episodes). This gives the confounded landscape where k and e change together. The position-isolation sweep fixes $e = 1$ and varies $k \in \{0, \dots, 7\}$ and $S \in \{0, \dots, 8\}$ on both suites (∼36K episodes on Spatial, ∼36K on Object). The model requeries every step, so this isolates the effect of position. The matched-budget comparison evaluates the standard protocol ($k = K - e$) against the decoupled protocol ($k = \tilde{k}(S)$) at $e \in \{2, 3, 4\}$ (∼20K episodes), holding inference frequency and compute fixed. In total we run approximately 132K simulated episodes.

At $S = 0$ and $c = 8$ (no staleness, full-chunk execution), our LIBERO-Spatial success rate matches the 97.6% reported in OpenVLA-OFT [3], confirming that our reproduction aligns with the published baseline.

4. Results

4.1. Chunk size confounds two effects

We first examine what the standard chunk-size interface reveals about staleness. Figure 2 shows the chunk-based protocol on LIBERO-Spatial. Smaller chunks perform better at high staleness, while larger chunks perform better at low staleness, consistent with the common claim that small chunks are more robust to observation delay.

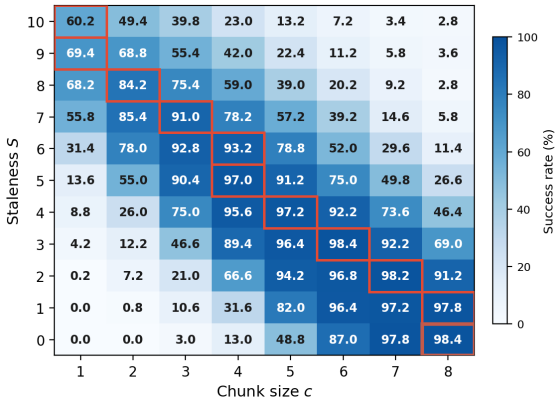

Figure 2. **Chunk-based protocol on LIBERO-Spatial** (∼40K episodes). Success rate vs. chunk size c and staleness S.

Reducing c conflates two distinct mechanisms. It increases the re-querying frequency and shifts which action positions are executed. With $c = 1$ the executed action is $a^{(7)}$, whereas $c = 8$ executes $a^{(0)}$ through $a^{(7)}$. Figure 2 alone cannot disentangle which mechanism drives the observed trend.

4.2. Position alone explains most of the gap

We ask how much of the chunk-size trend is driven by action position rather than by re-query frequency. To isolate the position effect, we apply the decoupled protocol with $e = 1$, which forces the model to re-query at every step and execute a single action at position k. Any variation across k at a fixed S is then attributable to position.

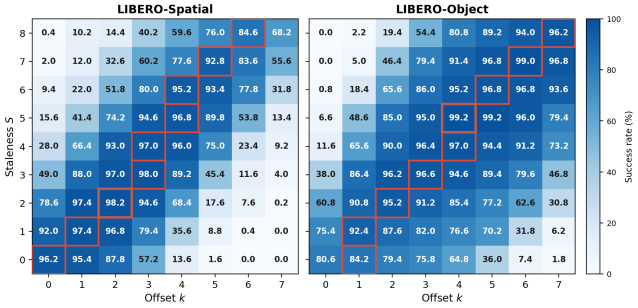

Figure 3. **Decoupled protocol with** $e = 1$ (∼36K episodes per suite). A diagonal band along $k = S$ emerges on both suites. Late positions diverge: Spatial (left) drops at $k \geq 6$, Object (right) stays near-optimal through $k = 7$.

Figure 3 reports the resulting 9×8 grid. At $S = 4$, success varies from 28.0% at $k = 0$ to 97.0% at $k = 3$. At $S = 6$, the corresponding values are below 1% and 95.2%. Despite identical compute and observations, the success rate varies drastically based solely on the chosen position, and the high-success region forms a diagonal band peaking near $k = S$.

This pattern is consistent with the temporal training target derived in Section 3. Since $a_t^{(k)}$ is trained to act at time $t + k$, an observation that is S steps stale aligns with the current environment time when $k = S$. The Spatial panel of Figure 3 shows that OpenVLA-OFT retains this chunk-internal temporal coordinate under staleness, and the Object panel shows the same diagonal pattern.

4.3. Matched-budget comparison

We ask how much of the position effect translates to practical gains under realistic compute budgets. From Figure 3, the diagonal degrades beyond $k = 4$ on LIBERO-Spatial (analyzed in Section 4.5), so we define a reference rule $\tilde{k}(S) = \min(S, 4)$. This rule is read off Figure 3 after the fact and does not depend on test-time tuning.

We compare the chunk-based protocol, which forces $k = 8 - e$, against the decoupled protocol with $k = \tilde{k}(S)$. Both configurations operate at the same execution count e, the same inference frequency, and the same compute budget. Any difference is attributable to action position.

CVPR #29

(e, S)	Chunk k=8−e	Decoupled k=$\tilde{k}$	Gap
$(2, 1)$	0.8%	**97.2%**	+96.4
$(3, 0)$	3.0%	**97.2%**	+94.2
$(3, 1)$	10.6%	**98.0%**	+87.4
$(4, 1)$	31.6%	**98.8%**	+67.2
$(3, 3)$	46.6%	**98.2%**	+51.6
$(4, 2)$	66.6%	**98.6%**	+32.0
$(2, 7)$	85.4%	**90.8%**	+5.4
$(3, 5)$	90.4%	**97.4%**	+7.0

Table 2. **Matched-budget comparison on LIBERO-Spatial.** Rows share the same e and compute budget. Decoupled rows use $\tilde{k}(S) = \min(S, 4)$. The gap is large when the chunk-forced k is far from $\tilde{k}$ (top) and small when close (bottom).

(S, k)	e=2	e=3	e=4	Range
Staleness-aligned ($k \approx S$)				
$(0, 0)$	96.8%	97.2%	97.6%	0.8 pp
$(1, 1)$	97.2%	98.0%	98.8%	1.6 pp
$(3, 3)$	97.8%	98.2%	96.6%	1.6 pp
$(4, 3)$	97.6%	98.2%	98.0%	0.6 pp
$(5, 4)$	96.6%	97.4%	96.8%	0.8 pp
$(6, 4)$	93.2%	93.2%	93.2%	0.0 pp
Misaligned ($k = 0$)				
$(2, 0)$	83.2%	85.4%	88.0%	4.8 pp
$(4, 0)$	33.0%	37.0%	38.8%	5.8 pp
$(6, 0)$	8.2%	9.2%	10.4%	2.2 pp

Table 3. **Effect of execution count e at fixed (S, k) on LIBERO-Spatial.** At staleness-aligned (S, k) (top), e shifts success by at most 2 pp. At misaligned $k = 0$ (bottom), the effect stays under 6 pp, far below the position effect of 32–96 pp (Table 2).

The observed gaps follow from the temporal semantics. Consider $(e = 2, S = 1)$, in which the chunk-based protocol executes $a^{(6)}$ and $a^{(7)}$. At observation time $t - 1$, $a^{(6)}$ targets time $t + 5$, which is five steps ahead of the current environment state. Executing an action intended for five steps in the future as if it were the current action is misaligned, and the resulting near-zero success rate is the expected outcome. The chunk-size interface offers no mechanism to substitute $a^{(1)}$ or $a^{(2)}$, even though these positions are temporally appropriate.

The same reasoning applies throughout the table. Whenever $S \ll 8 - e$, the chunk-forced k is far from the staleness-appropriate k, and the gap tracks the degree of mismatch. The bottom two rows of Table 2 confirm this. At $(e = 2, S = 7)$ and $(e = 3, S = 5)$, where the chunk-forced k already happens to be near optimal, the gap shrinks to 5.4 and 7.0 pp. Because the compute budget is held fixed throughout, these gaps arise from the choice of action position.

4.4. Execution count plays a minor role

A natural question is whether the gains in Section 4.3 arise partly from the choice of e rather than k. To check this, we sweep $e \in \{2, \ldots, 8\}$ at fixed (S, k) pairs on LIBERO-Spatial.

Within the explored range $e \in \{2, \ldots, 8\}$, varying e has little effect on success rate. At staleness-aligned conditions ($k \approx S$), the spread across e stays within 2 pp. At misaligned $k = 0$, e has a larger effect (up to ∼7 pp), but this is an order of magnitude smaller than the position effect of 32–96 pp (Table 2). In our setting, position is the dominant axis and e acts as a compute knob. We discuss the limits of this conclusion in Section 5.

4.5. Cross-suite analysis: late positions degrade on Spatial but not Object

The decoupled protocol reveals a second, distinct effect beyond temporal alignment. Consider the diagonal conditions where $k = S$, each of which targets the current timestep. If action quality depended on temporal alignment, these conditions would yield similar success rates regardless of k.

Figure 3 shows that this is not the case. On LIBERO-Spatial (left panel), the diagonal remains high through $k = 4$ (95.8% at (S=4, k=4)) and then drops: 89.8% at $k = 5$, 77.8% at $k = 6$, and 55.6% at $k = 7$. Since all conditions share the same target time, the drop reflects prediction-quality degradation at late chunk positions, not temporal misalignment.

On LIBERO-Object (right panel), no such degradation appears within the tested range. The diagonal stays above 95% through $k = 7$, and at (S=7, k=7) success is 96.8%. Table 4 summarizes the contrast: at $S = 8$, Object peaks at $k = 7$ (96.2%) while Spatial's $k = 7$ drops to 68.2%. Both suites use the same base model architecture, so the discrepancy arises from the fine-tuning distribution rather than the model itself.

Data-level correlate. Why do late positions degrade on Spatial but not on Object? We examine the fine-tuning demonstrations using state-conditioned variance. Chunks are grouped by starting proprioceptive state, and within-bin action variance is computed per chunk position. LIBERO-Spatial shows about 2× higher variance than LIBERO-Object at all positions (ratio 1.82–2.07), consistent with L1 regression collapsing multimodal futures toward their mean where futures are more diverse. This is a suite-level correlation, not a causal explanation: per-task variance does not predict per-task failure at $k = 7$ ($r \approx 0$). The Object

baseline at S=0 (80.6% vs. 96.2%) reflects overall task difficulty rather than staleness sensitivity.

S	Spatial peak (k^*)	Object peak (k^*)	Δ
0	96.2% (k=0)	80.6% (k=0)	−15.6
2	98.2% (k=2)	95.2% (k=2)	−3.0
4	97.0% (k=3)	97.0% (k=4)	0.0
6	95.2% (k=4)	96.8% (k=5)	+1.6
8	84.6% (k=6)	96.2% (k=7)	+11.6

Table 4. **Peak performance per staleness level.** On Object, k^* tracks S through $k = 7$. On Spatial, k^* saturates at 4 as late positions degrade. The Δ column (Object minus Spatial) flips sign beyond $k = 4$.

5. Discussion and Conclusion

Practical recipe. Our results point to a simple deployment guideline for OpenVLA-OFT. Measure the system's vision latency L, convert it to staleness steps $S = \lfloor L \cdot f_c \rfloor$ at control frequency f_c (at LIBERO's 20 Hz, one step of S corresponds to 50 ms), and set $k = \min(S, k_{\max})$, where $k_{\max}$ is validated empirically for each task distribution (4 for Spatial and 7 for Object in our experiments). The execution count e can be chosen to match the available compute. No retraining is required.

Why the gap is so large. The 32–97 pp gaps in Table 2 appear whenever S is much smaller than $8 - e$. In this regime, the chunk-based protocol executes actions trained for several steps in the future as if they applied to the current timestep, which is the worst case for the chunk-size interface. Moderate latency ($S = 1$–4) combined with moderate compute budget ($e = 2$–4) covers much of what a real VLA deployment looks like, so this regime is not a corner case.

Practical access to staleness. The decoupled protocol requires choosing k as a function of S. In real systems, vision latency can be read from image timestamps, which gives a direct estimate of S. When staleness varies online, k can be set from a conservative upper bound.

Relation to hierarchical approaches. A separate line of work decomposes the policy itself, pairing a slow chunked model with a faster reactive controller [13]. That operates at the architecture level and is orthogonal to the question of which position inside a single chunk to execute. The decoupled protocol could sit inside either layer of such a system.

Limitations. Our evaluation uses a single VLA architecture (OpenVLA-OFT with L1 regression) and two LIBERO task suites in simulation. Whether the same picture extends to diffusion- or flow-based action heads [4, 5, 8] or to real-hardware deployments remains open. The staleness simulation delays only vision while proprioception stays current, whereas real hardware shows correlated delays across modalities. Our execution-count sweep covers $e \in \{2, \ldots, 8\}$ at $K = 8$, and for systems with longer chunks or much larger e, the relative contribution of e may differ. Finally, $\tilde{k}(S) = \min(S, 4)$ is a descriptive rule fit to LIBERO-Spatial at $e = 1$ and serves only as a fixed reference in Table 2, not as a universal policy.

Conclusion. The chunk-size interface in OpenVLA-OFT hides a degree of freedom that matters under latency, which is the choice of which action positions from a predicted chunk are executed. Decoupling this choice from re-query frequency yields 32–97 pp success-rate gains at identical compute on LIBERO-Spatial, and the size of the gap tracks the temporal alignment between prediction and execution. The effect is task-dependent. Late positions stay reliable on LIBERO-Object but degrade on LIBERO-Spatial, so chunk-internal reliability should be characterized for each task distribution rather than assumed uniform. Within the setting we study, the position used inside a chunk is a tunable design variable, not a side effect of chunk size. Whether this picture holds for other VLA architectures and on real hardware is left to future work.

References

[1] Anthony Brohan et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *CoRL*, 2023. 1
[2] Moo Jin Kim et al. OpenVLA: An open-source vision-language-action model. In *CoRL*, 2024. 2
[3] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv:2502.19645*, 2025. 2, 3
[4] Kevin Black et al. π_0: A vision-language-action flow model for general robot control. *arXiv:2410.24164*, 2024. 5
[5] Octo Model Team et al. Octo: An open-source generalist robot policy. In *RSS*, 2024. 1, 5
[6] Karl Pertsch et al. FAST: Efficient action tokenization for vision-language-action models. *arXiv:2501.09747*, 2025. 2
[7] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *RSS*, 2023. 1, 2
[8] Cheng Chi et al. Diffusion policy: Visuomotor policy learning via action diffusion. In *RSS*, 2023. 1, 2, 5
[9] Bo Liu et al. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *NeurIPS D&B*, 2023. 2
[10] Open X-Embodiment Collaboration et al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *ICRA*, 2024.
[11] Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *arXiv:2506.07339*, 2025. 1, 2

[12] Yuejiang Liu, Jubayer Ibn Hamid, Annie Xie, Yoonho Lee, Maximilian Du, and Chelsea Finn. Bidirectional decoding: Improving action chunking via guided test-time sampling. In *ICLR*, 2025. 2

[13] Han Xue et al. Reactive diffusion policy: Slow-fast visual-tactile policy learning for contact-rich manipulation. In *RSS*, 2025. 5

[14] Kohei Sendai, Maxime Alvarez, Tatsuya Matsushima, Yutaka Matsuo, and Yusuke Iwasawa. Leave no observation behind: Real-time correction for VLA action chunks. *arXiv:2509.23224*, 2025. 2

[15] Mustafa Shukor et al. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv:2506.01844*, 2025. 1, 2