Latency-Tolerant Cloud-Edge Collaborative Vision-Language-Action Models via Emergent Representational Specialization

Daojie Peng[1], **Fulong Ma**[1], **Bingtao Wang**[2], **Sheng Wang**[3], **Jun Ma**[1*]

[1]HKUST(GZ), [2]Shandong University, [3]RoboScience
dpeng108@connect.hkust-gz.edu.cn, fmaaf@connect.hkust-gz.edu.cn,
wangbt@mail.sdu.edu.cn, swangei@connect.ust.hk, jun.ma@ust.hk

Abstract

Deploying billion-parameter Vision-Language-Action (VLA) policies on mobile robots creates a systems conflict: semantic reasoning benefits from cloud GPUs, whereas closed-loop control must respond locally despite network delay and jitter. Existing hierarchical and asynchronous policies improve throughput, but their slow-path representations can still arrive stale or require explicit scheduling and delay cues. We introduce CloudEdgeVLA, a cloud-edge policy that treats temporal misalignment as a representation-learning problem. A cloud VLA encodes delayed observations into slowly varying task features, while a lightweight edge head combines the latest available cloud feature with current local vision. During training, current and randomly delayed frames are paired with the same current action target in fresh and stale paths. This objective encourages the cloud representation to preserve task-level information while the edge path supplies state-sensitive corrections. Across four LIBERO suites, CloudEdgeVLA retains 63.8–78.0% success with a 40-step uniform-delay window, whereas VLASH reaches at most 6.4% and the evaluated single-path baselines at most 3.0%. By removing blocking synchronization from the control loop, the design offers a practical route to scalable VLA deployment in which cloud models can grow while edge computation remains lightweight and responsive.

Introduction

Vision-Language-Action (VLA) models transfer semantic knowledge from large vision-language backbones to robot control. RT-2, OpenVLA, π_0, and Octo demonstrate increasingly broad language following and manipulation capabilities (Zitkovich et al. 2023; Kim et al. 2025; Black et al. 2024; Ghosh et al. 2024). Their scale, however, creates a deployment bottleneck: the compute needed for a multi-billion-parameter backbone is difficult to place on a power- and weight-constrained robot, while manipulation still requires a responsive closed loop.

Cloud robotics can move expensive perception and planning to remote accelerators (Kehoe et al. 2015; Wan et al. 2016). This split makes larger policies deployable, but it also exposes the controller to communication delay, jitter, and loss (Chinchali et al. 2019). A feature returned by the cloud describes the scene when its input image was captured, not necessarily the scene in which the action will execute. Blocking until a new feature arrives lowers the control rate; executing with the latest feature avoids blocking but introduces temporal mismatch.

Multi-rate policies provide part of the solution. MResT combines low-frequency global features with high-frequency local sensing (Saxena, Sharma, and Kroemer 2023); DP-VLA and HiRT separate slow semantic reasoning from fast visuomotor control (Han, Kim, and Jang 2024; Zhang et al. 2025); and SmolVLA decouples action generation from execution through an asynchronous inference stack (Shukor et al. 2025). More recent correction and semantic-action decoupling methods explicitly address inference-time staleness (Sendai et al. 2025; Yan et al. 2026). These systems establish the value of fast-slow execution, but they primarily target inference scheduling, action-chunk correction, or explicit temporal conditioning. The cloud-edge setting additionally requires the representation passed across the network to remain useful when its age is variable and not known in advance.

CloudEdgeVLA addresses this requirement by separating information according to temporal role, following the System 1/System 2 analogy (Kahneman 2011). The cloud backbone provides slowly varying task context (what to do), and the edge head uses current local vision for state-sensitive control (how to do it now). The edge always consumes the most recently received cloud feature and never waits for a particular update. To make that feature useful across ages, paired-frame dual-path training applies the same current action supervision to cloud features computed from both current and randomly delayed frames. This creates pressure for the cloud path to retain information shared across the pair, while current local vision resolves state details needed at execution time.

Contributions.

1. We formulate asynchronous cloud-edge VLA control around a non-blocking interface: a lightweight Vision-Augmented Action Head fuses the latest cloud feature with current edge vision without requiring clock or frequency alignment.
2. We introduce paired-frame dual-path training, which jointly supervises fresh and delayed cloud features against

*Corresponding author.

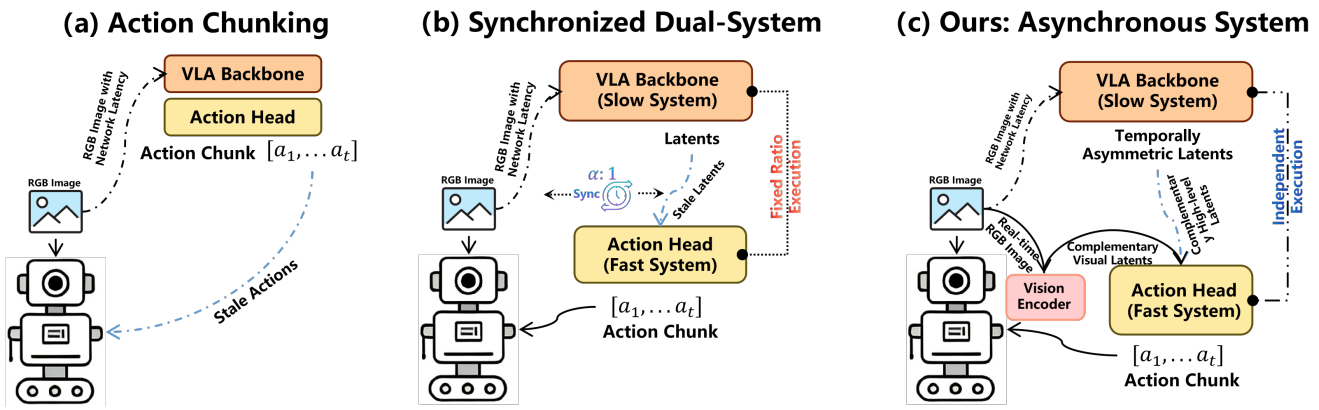

Figure 1: **Conceptual comparison of dual-system paradigms.** (a) Action-chunk re-planning assumes environment stasis during open-loop execution, compounding staleness. (b) Synchronized fast-slow pipelines require fixed frequency ratio α and bounded latency—conditions rarely met in networked deployments. (c) Our asynchronous paradigm: the cloud produces time-invariant planning features; the edge grounds them with real-time vision without any temporal alignment. The edge never blocks on a cloud response and uses the latest available feature.

the current action and thereby encourages temporal specialization without an explicit invariance loss.

3. We evaluate synchronous control and uniformly sampled observation delays on LIBERO (Liu et al. 2023), decompose delay sensitivity across the backbone and action head, and include a small real-robot success-rate sanity check.

Related Work

Vision-Language-Action Models

RT-2 represents robot actions as language tokens, while OpenVLA provides an open 7B-parameter policy trained on diverse robot data (Zitkovich et al. 2023; Kim et al. 2025). Continuous and generative action heads improve control quality and throughput: OpenVLA-OFT combines parallel action-chunk decoding with continuous regression, and π_0 uses flow matching (Kim, Finn, and Liang 2025; Black et al. 2024). Octo and UniVLA explore generalist policies based on transformer and unified multimodal tokenization, respectively (Ghosh et al. 2024; Wang et al. 2025). These works establish strong policy backbones; our focus is the systems and learning problem created when backbone features cross a delayed network boundary.

Dual-System Robot Architectures

Action chunking reduces effective planning horizon but can weaken feedback during chunk execution (Zhao et al. 2023; Chi et al. 2023). Multi-rate alternatives pair a slow semantic module with a fast local policy: MResT uses different sensing rates, DP-VLA and HiRT couple slow VLM reasoning with faster control, and SmolVLA generates chunks asynchronously (Saxena, Sharma, and Kroemer 2023; Han, Kim, and Jang 2024; Zhang et al. 2025; Shukor et al. 2025). Fast-in-Slow embeds fast execution within a slow VLM (Chen et al. 2025). Among asynchronous methods, VLASH rolls robot state forward to execution time and evaluates delay on all four LIBERO suites, while A2C2 applies current-observation corrections to stale chunks (Tang et al. 2025; Sendai et al. 2025). Semantic-action decoupling instead uses history and time-misalignment training to interpret stale semantics (Yan et al. 2026). CloudEdgeVLA differs by sending a learned representation rather than actions across the slow-fast boundary and training it with paired current/delayed frames without delay metadata at inference.

Latency Robustness in Control

Delayed observations have also been studied independently of VLAs. Concurrent control lets a robot act while policy computation proceeds (Xiao et al. 2020); delayed-observation RL uses history augmentation or delay-resolved state estimates (Wang et al. 2024); and world-model methods predict a current latent state from delayed inputs (Karamzade et al. 2024). CloudEdgeVLA does not reconstruct the current state. It reserves a direct, current visual path at the edge and trains the cloud feature to supply complementary task context. This design connects delayed-control learning with the established cloud-robotics goal of offloading expensive computation without making the local loop depend on network response time (Kehoe et al. 2015; Chinchali et al. 2019).

Method

Problem Formulation

We formalize the asynchronous cloud-edge VLA deployment problem as follows. At each environment step t:

- The **edge** captures the current observation o_t and sends it to the cloud.
- Due to network latency, the cloud receives o_{t-k} (delayed by k steps) and produces planning features $h_{t-k} = f_\theta(o_{t-k}, \ell)$, where ℓ is the language instruction and f_θ is the VLA backbone.
- The **edge** must produce the action a_t using the stale planning features h_{t-k} and its own real-time observation o_t.

The goal is to learn an action head g_ϕ such that $\hat{a}_t = g_\phi(h_{t-k}, v_\psi(o_t))$ achieves high task success despite the temporal mismatch between h_{t-k} and o_t, where v_ψ is a local vision encoder.

System Architecture

Our framework consists of three components operating under a clear temporal asymmetry (see Figure 2):

Cloud-Side VLA Backbone f_θ (System 2, Latency-Insensitive). A large-scale vision-language model that processes visual observations and language instructions to produce high-level planning representations. Given an observation o and instruction ℓ, the backbone produces hidden-state representations:

$$h = f_\theta(o, \ell) \in \mathbb{R}^{L \times D} \tag{1}$$

where L is the number of action-token positions (corresponding to an action chunk of length T with A action dimensions, so $L = T \times A$) and D is the hidden dimension. The backbone is parameterized by LoRA-adapted weights (Hu et al. 2022) on top of a pretrained vision-language model and runs exclusively on the cloud server. Critically, the backbone is designed to learn features that encode *what* to do (task goals, manipulation strategy, object semantics) rather than *exactly when* to do it, as motivated in *Emergent Representational Specialization*.

Edge-Side Vision Encoder v_ψ (System 1 Perception, Latency-Sensitive). A lightweight vision encoder (e.g., SigLIP-Base (Zhai et al. 2023)) that runs locally on the robot and extracts real-time visual features from the current observation:

$$z_t = v_\psi(o_t) \in \mathbb{R}^{D_v} \tag{2}$$

The vision encoder is frozen during training to leverage pretrained visual representations. Crucially, z_t is always computed from the *current* observation o_t, making it a **delay-free signal** that captures the instantaneous state of the environment.

Edge-Side Action Head g_ϕ (System 1 Control, Latency-Sensitive). A learnable module that fuses the (potentially stale) cloud planning features with the real-time edge vision features to predict continuous actions:

$$\hat{a}_t = g_\phi(h_{t-k}, z_t) \in \mathbb{R}^{T \times A} \tag{3}$$

The action head first mean-pools the planning features over the action-dimension axis to obtain per-timestep planning embeddings, projects the vision features into the same latent space, concatenates them, and passes the result through a residual MLP to predict the action chunk. The action head acts as a **real-time grounding layer**: it translates the cloud's high-level (but potentially stale) directives into precise motor commands using the edge's current visual context.

Paired-Frame Dual-Path Training

The key challenge is that the action head must learn to produce correct actions from *both* fresh and stale planning features. We achieve this through a paired-frame training strategy that leverages the temporal structure of demonstration trajectories.

Paired Frame Extraction. During training, each sample provides a window of W consecutive observations from the same episode: $\mathcal{W}_t = \{o_{t-W+1}, \ldots, o_t\}$. From this window, we construct:

- **Current frame**: o_t (the latest observation)
- **Delayed frame**: o_{t-d}, where $d \sim \text{Uniform}(1, W-1)$

Dual Forward Pass. Both frames are processed through the cloud backbone:

$$h^{\text{fresh}} = f_\theta(o_t, \ell), \quad h^{\text{stale}} = f_\theta(o_{t-d}, \ell) \tag{4}$$

Note that h^{stale} is *not detached* from the computation graph; both receive gradients.

Vision-Augmented Action Prediction. Both sets of planning features are fused with the *same* real-time vision features $z_t = v_\psi(o_t)$:

$$\hat{a}^{\text{fresh}} = g_\phi(h^{\text{fresh}}, z_t), \quad \hat{a}^{\text{stale}} = g_\phi(h^{\text{stale}}, z_t) \tag{5}$$

Dual-Path Loss. Both predictions are trained toward the same ground-truth action a_t:

$$\mathcal{L} = (1-\lambda) \underbrace{\|\hat{a}^{\text{fresh}} - a_t\|_1}_{\mathcal{L}_{\text{fresh}}} + \lambda \underbrace{\|\hat{a}^{\text{stale}} - a_t\|_1}_{\mathcal{L}_{\text{stale}}} \tag{6}$$

$\mathcal{L}_{\text{fresh}}$ trains the action head for synchronous operation and provides direct action supervision to the backbone. $\mathcal{L}_{\text{stale}}$ trains the action head to *compensate* for stale planning features using real-time vision: when h^{stale} is misaligned with the current state, the action head must rely more heavily on z_t. λ balances the two losses; we use curriculum learning to gradually increase λ from 0 to λ_{max} over the first part of training steps n_{warmup}, allowing the backbone to first learn strong representations before being pressured to be delay-invariant.

Deployment Protocol

At test time, the system operates fully asynchronously:

1. Edge captures observation o_t.
2. Edge sends o_t to cloud (asynchronous, non-blocking).
3. When cloud returns h (possibly from a previous o_{t-k}): $h_{\text{received}} \leftarrow h$.
4. Edge computes $z_t = v_\psi(o_t)$ (real-time, local).
5. Edge computes $\hat{a}_t = g_\phi(h_{\text{received}}, z_t)$.
6. Edge executes $\hat{a}_t$.

The robot **never blocks** waiting for the cloud: it always uses the most recently received planning features combined with current real-time vision.

Emergent Representational Specialization

The stale path receives an action target from time t but a cloud input from $t-d$:

$$\nabla_\theta \mathcal{L}_{\text{stale}} = \nabla_\theta \|g_\phi(f_\theta(o_{t-d}, \ell), v_\psi(o_t)) - a_t\|_1. \tag{7}$$

Across random d, features tied only to the instantaneous state of o_{t-d} are unreliable predictors of a_t, whereas task identity, goal, and coarse progress are more stable. The objective

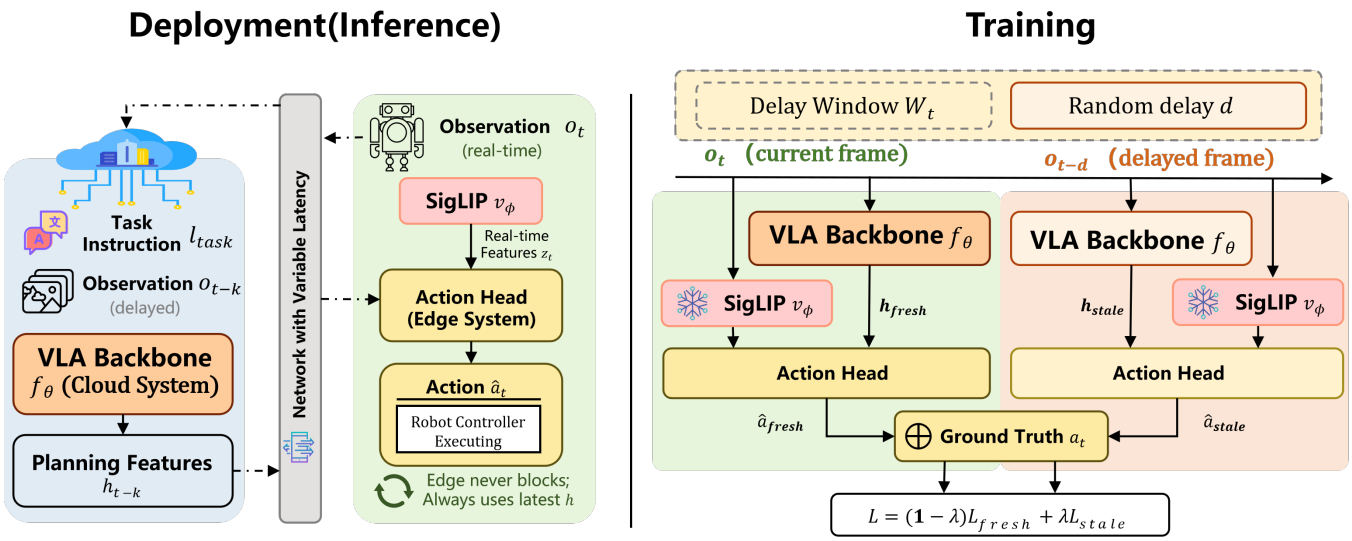

Figure 2: **System architecture and training pipeline. Left:** At deployment time, the cloud VLA backbone produces planning features from a delayed observation, while the edge vision encoder extracts real-time features from the current observation. The action head fuses both to produce actions without any blocking or temporal alignment. **Right:** During training, paired frames from the same episode (current and randomly delayed) are processed through the backbone, producing fresh and stale planning features. Both are fused with the same real-time vision features and supervised toward the same ground-truth action, inducing representational specialization.

therefore *encourages*, but does not mathematically guarantee, invariance to temporal displacement, consistent with the broader connection between nuisance variation and invariant representations (Achille and Soatto 2018). The current edge feature z_t remains available to encode state-sensitive information.

This design targets complementary robustness at both sides of the cloud–edge interface. Paired-frame training encourages cloud features to be less sensitive to observation age, while the edge-aware action head attenuates residual representation mismatch before it reaches the action output. We validate both effects in the experiments that follow.

Experiments

Implementation

We build on **OpenVLA-OFT** (Kim, Finn, and Liang 2025), which adapts a 7B OpenVLA backbone using LoRA, parallel action-chunk decoding, and continuous L1 regression. Our modifications include: (1) a new Vision-Augmented Action Head that replaces the original L1RegressionActionHead; (2) a paired-frame dataset pipeline extending the RLDS loader to extract historical frames from the same episode; (3) a dual-path forward pass with dual L1 loss; and (4) a delayed evaluation protocol that simulates network latency.

Benchmark

We evaluate on the **LIBERO** manipulation benchmark (Liu et al. 2023), which consists of 4 task suites (Spatial, Object, Goal, and Long) of 10 tasks each, with 50 demonstration episodes per task. LIBERO provides a standardized testbed for evaluating language-conditioned manipulation in simulation.

Baselines

We compare single-path OpenVLA (Kim et al. 2025), OpenVLA-OFT (Kim, Finn, and Liang 2025), and Uni-VLA (Wang et al. 2025); future-state-aware asynchronous VLASH (Tang et al. 2025); and CloudEdgeVLA. We feed delayed frames to the single-path policies without changing their inference paths. VLASH is evaluated on the same uniform-delay-window grid, while CloudEdgeVLA receives delayed cloud frames and current edge frames.

Evaluation Conditions

We evaluate under:

- **No delay** ($d_{\max}$=0), the synchronous reference.
- **Uniform delay** ($k \sim \text{Uniform}\{1, \ldots, d_{\max}\}$), with $d_{\max} \in \{5, 10, 15, 20, 25, 30, 40\}$ steps.

We report task success rate (%) over 50 trials per task. For CloudEdgeVLA, each setting is evaluated with three random seeds (7, 8, and 9), and we report the mean and standard deviation across seeds.

Ablation Studies

We isolate the contribution of current edge vision, encoder capacity, and the two loss paths. Table 2 reports aggregate success under uniform delay; all variants use the same backbone, demonstrations, and optimization budget.

The decisive architectural factor is whether the action head receives a current observation. Adding the frozen SigLIP-Base encoder used by our default model raises success from

Table 1: **LIBERO task success (%) under uniformly sampled observation delay.** At $d_{\max}=40$, CloudEdgeVLA retains 63.8–78.0% success across the four suites, whereas VLASH reaches at most 6.4% and the single-path baselines at most 3.0%. CloudEdgeVLA entries report mean ± standard deviation over seeds 7, 8, and 9; published baseline values are point estimates.

	$d_{\max}=0$				$d_{\max}=10$				$d_{\max}=20$				$d_{\max}=40$			
Method	Spat.	Obj.	Goal	Long	Spat.	Obj.	Goal	Long	Spat.	Obj.	Goal	Long	Spat.	Obj.	Goal	Long
OpenVLA (Kim et al. 2025)	84.6	71.2	77.0	56.2	6.8	1.2	2.2	5.2	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
OpenVLA-OFT (Kim, Finn, and Liang 2025)	**98.4**	98.6	**97.2**	93.4	10.6	3.6	26.2	8.4	0.0	0.0	4.0	1.4	0.0	0.0	0.0	0.0
UniVLA (Wang et al. 2025)	96.0	96.6	94.6	93.2	27.4	34.8	48.2	35.0	0.4	2.6	21.8	2.2	0.0	0.0	3.0	0.5
VLASH (Tang et al. 2025)	97.3	**99.6**	96.7	**93.5**	60.0	62.8	56.8	45.6	5.0	8.2	28.2	9.2	0.0	0.4	6.4	0.0
Ours	97.9±0.23	97.8±0.20	96.5±0.31	91.7±0.42	**93.6±0.53**	**94.4±0.53**	**92.9±0.83**	**83.2±0.53**	**89.6±1.20**	**92.1±0.42**	**90.9±0.81**	**78.1±0.42**	**76.4±0.92**	**75.6±0.80**	**78.0±1.20**	**63.8±1.51**

Table 2: **Ablation: Vision encoder and loss components.** Aggregate success under uniform delay $d_{\max}=10$.

Variant	**Success (%)**
No vision encoder (stale h only)	31.6
SigLIP-Base (frozen, default)	95.1
SigLIP-SO400M (frozen)	95.8
$\mathcal{L}_{\text{fresh}}$ only	54.8
$\mathcal{L}_{\text{stale}}$ only	89.2
$\mathcal{L}_{\text{fresh}} + \mathcal{L}_{\text{stale}}$ (Ours)	95.1

Table 3: **Real-robot task success (%).** Cells report static/-dynamic success over 10 trials per variant. CE-VLA denotes CloudEdgeVLA. RTT is the emulated delay added to the native serving pipeline.

	Added network RTT (ms)		
Method	**0**	**400**	**1000**
VLASH (Tang et al. 2025)	100/90	60/30	0/0
CE-VLA	**100/90**	**90/90**	**80/70**

31.6% to 95.1%, a 63.5-point gain over stale cloud features alone. Replacing it with the substantially larger SigLIP-SO400M reaches 95.8%, only 0.7 points higher. Thus, most of the benefit comes from real-time visual grounding rather than encoder scale, supporting the lightweight Base model for edge deployment. The loss ablation shows a complementary pattern under this default configuration. Stale-only supervision reaches 89.2%, 34.4 points above fresh-only training, confirming that explicit exposure to delayed features drives most of the robustness. Joint supervision restores the full model to 95.1%, another 5.9-point gain over stale-only training; the fresh path therefore provides complementary grounding rather than replacing delay exposure.

Real-Robot Sanity Check

We conduct a small Franka pick-and-place pilot to check physical feasibility. An RTX 5080 workstation runs the edge vision encoder, action head, and control loop, while the 7B backbone is served from an RTX 4090. The robot must place a toy bear into a box. We test a static setting and a dynamic variant in which the task target is displaced within 10 cm during reaching. A rollout succeeds when the bear is released inside the box. Each method receives 10 trials per variant under added RTT of 0, 400, and 1000 ms. These values are emulated delays added on top of the native system latency: 0 ms means no *additional* RTT, not zero camera-to-action latency. Camera acquisition, preprocessing, transport, request scheduling, and model inference remain present in every profile.

Table 3 is a feasibility check rather than a hardware benchmark. With no added RTT, CE-VLA and VLASH both reach 100% in the static task, while CE-VLA is 10 points higher in the dynamic task (100% versus 90%). At 400 ms, CE-VLA reaches 100/90% versus 70/30% for VLASH, gaps of 30 and 60 points. At 1000 ms, CE-VLA retains 80/70%, whereas VLASH falls to 0/0%. The larger separation under added delay is consistent with the closed-loop simulation results. With only 10 trials per cell, these differences are descriptive evidence of physical feasibility rather than a statistically powered hardware benchmark.

Figure 3 complements these success rates with representative simulation and real-robot rollouts.

Analysis

Delay Robustness

Figures 4 and 5 summarize the macro-average across the four LIBERO suites, while Table 1 reports the suite-level results. At $d_{\max}=10$, OpenVLA reaches at most 6.8%, and OpenVLA-OFT reaches at most 26.2%. UniVLA degrades more gradually on Goal and Long, retaining 48.2% and 35.0%, respectively, but reaches at most 3.0% at $d_{\max}=40$. VLASH is stronger at intermediate delay windows, with 45.6–62.8% success at $d_{\max}=10$, yet falls to 0.0–6.4% at $d_{\max}=40$. Across the four-suite mean curve, CloudEdgeVLA attains a normalized delay AUC of 90.8%, compared with 32.4% for VLASH, and retains 76.5% of its synchronous success at $d_{\max}=40$ compared with 1.8% for VLASH. Thus synchronous accuracy and moderate-delay gains do not by themselves imply tolerance to severe observation staleness.

CloudEdgeVLA retains 76.4, 75.6, 78.0, and 63.8% on Spatial, Object, Goal, and Long at $d_{\max}=40$, corresponding to drops of 21.5, 22.2, 18.5, and 27.9 percentage points from $d_{\max}=0$. The degradation is therefore suite-dependent: Long shows the largest loss, followed by Object and Spatial, while Goal is the most stable over the tested range. Even in the worst case, Long, CloudEdgeVLA remains 63.3 points above the strongest baseline at $d_{\max}=40$. These results support robustness over the tested range; they do not imply tolerance to unbounded delay or cloud disconnection.

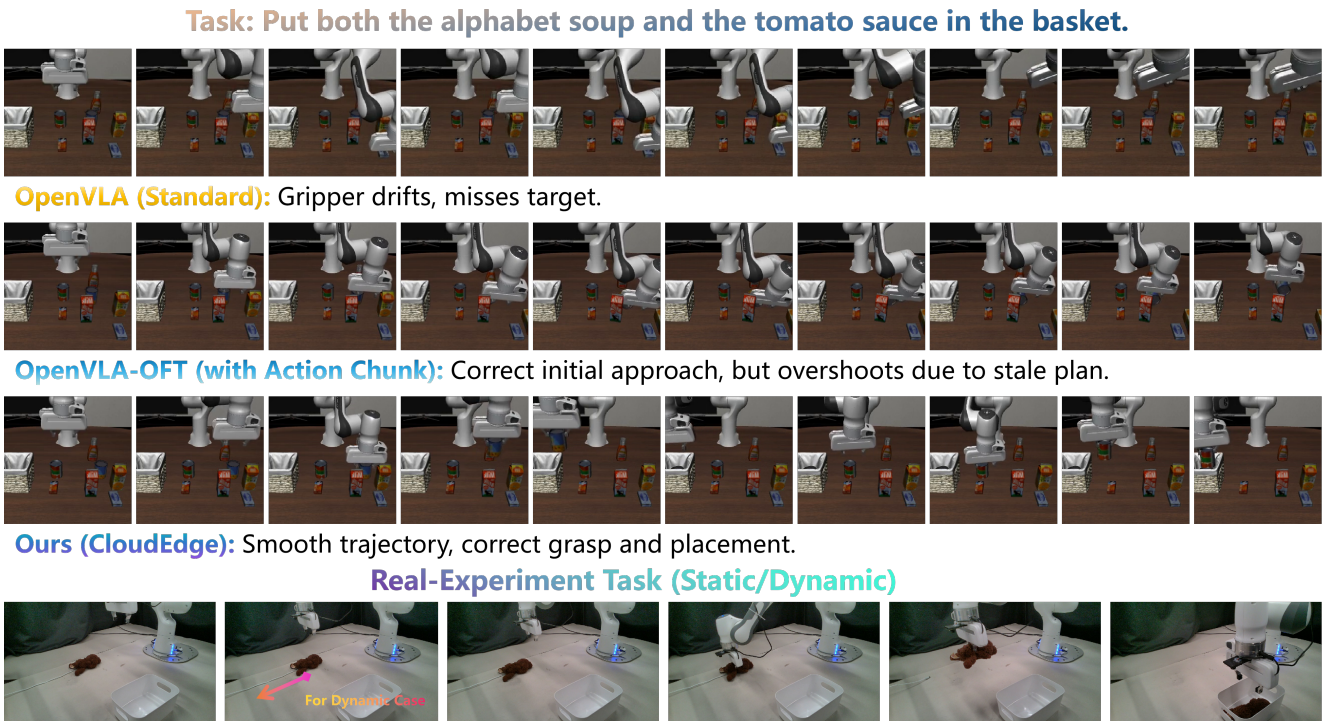

Figure 3: **Qualitative rollouts in simulation and on the real robot.** The top three sequences compare OpenVLA, OpenVLA-OFT, and CloudEdgeVLA on the same LIBERO task of placing two specified objects in the basket. The bottom sequence shows CloudEdgeVLA executing the real-robot task of placing a toy bear into a box; the arrows mark the externally displaced target in the dynamic setting.

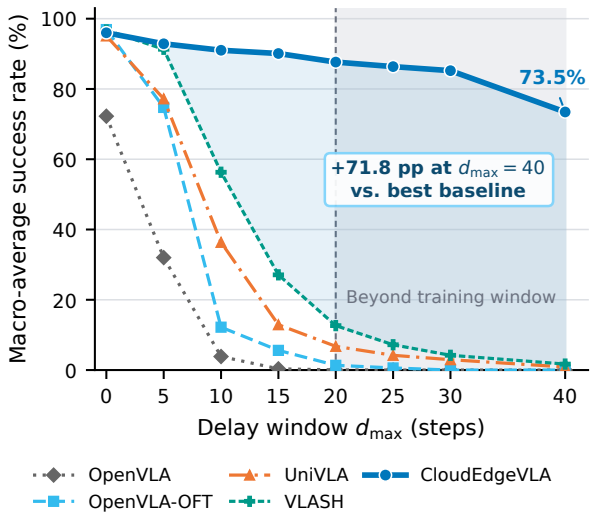

Figure 4: **Closed-loop uniform-delay sweep on LIBERO.** Each point averages Spatial, Object, Goal, and Long. The shaded range $d_{\max}>20$ lies beyond training. At $d_{\max}=40$, CloudEdgeVLA averages 73.5% success, 71.8 points above VLASH.

Backbone–Head Delay Mechanism

We test the learned interface rather than infer specialization from task success alone. We compare the original OpenVLA-OFT checkpoint with the aligned 120k CloudEdgeVLA checkpoint on 80 shared demonstration timesteps from the ten LIBERO-Spatial tasks. For each current timestep, we replace the image used by the cloud backbone with a frame delayed by d steps while retaining current proprioception. We measure backbone drift and end-to-end action drift as

$$D_h(d) = \mathbb{E}\left[1 - \cos(h_t, h_{t-d})\right], \quad D_a(d) = \mathbb{E}\left[\left\|\hat{a}_t^{(0)} - \hat{a}_t^{(d)}\right\|\right], \tag{8}$$

and define the staleness transfer gain

$$\kappa(d) = \frac{D_a(d)}{D_h(d) + \epsilon}. \tag{9}$$

Lower D_h indicates a more temporally stable backbone; lower κ indicates that the action head transfers less residual representation drift into actions. The supplement separately plots D_a and normalized MAE to demonstrated action chunks, ruling out the trivial explanation that an insensitive predictor merely changes less.

Figures 6 and 7 localize robustness to both sides of the interface: at d=20, backbone drift falls from 0.391 to 0.160 and κ from 1.082 to 0.295. Their combination reduces action drift from 0.423 to 0.047 and demonstration MAE from 0.421

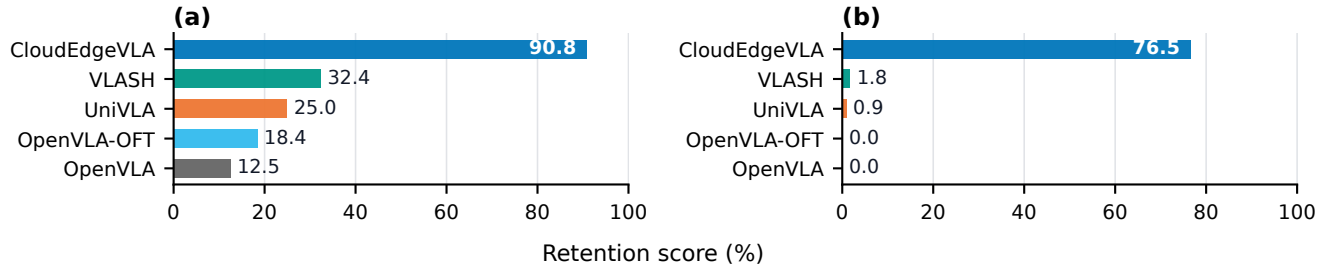

Figure 5: **Closed-loop delay-retention summaries.** Metrics are computed from each method's four-suite macro-average curve. (a) Normalized area under the uniform-delay-window curve. (b) Synchronous success retained at $d_{\max}$=40. CloudEdgeVLA achieves 90.8% and 76.5%, respectively; VLASH reaches 32.4% and 1.8%.

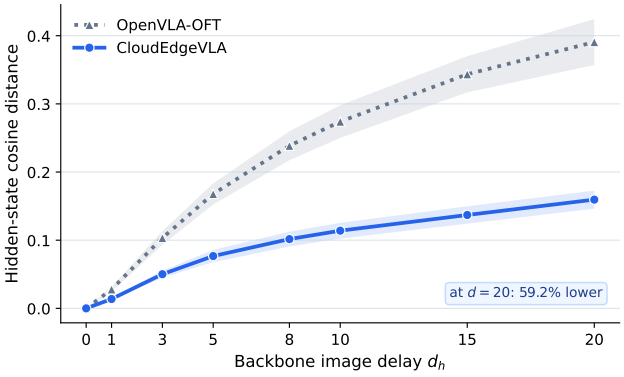

Figure 6: **Backbone representation staleness on LIBERO-Spatial.** Over 80 shared states from ten tasks, CloudEdgeVLA reduces backbone drift D_h by 59.2% at d=20; bands are task-level 95% confidence intervals.

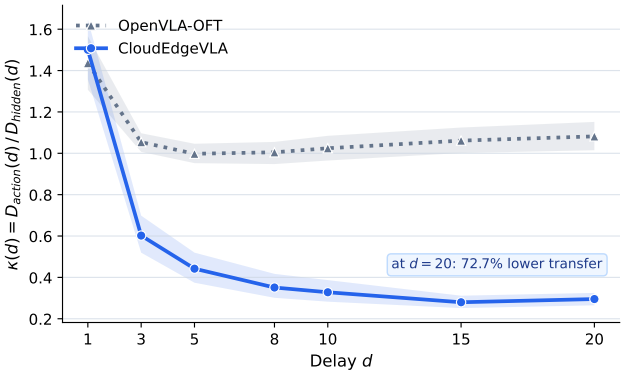

Figure 7: **Head staleness-transfer gain on LIBERO-Spatial.** CloudEdgeVLA reduces κ, the action drift transferred per unit backbone drift, by 72.7% at d=20.

to 0.048. The supplement visualizes these end-to-end curves, their small fresh-accuracy cost, per-task consistency, action-chunk effects, and counterfactual edge age. The latter audit attributes this checkpoint's gain mainly to backbone stability and head attenuation, not a large directly measured correction from current edge vision.

Discussion and Limitations

CloudEdgeVLA changes the slow-fast interface from a time-indexed action plan to a reusable task feature grounded by current local vision. This does not make old information current; it limits which decisions depend on that information. Compared with action-chunk replay, the edge closes the visual feedback loop at every step. Compared with reconstructing the present through a learned world model (Karamzade et al. 2024), it directly observes the current image but provides no prediction for unobserved state. These choices favor a small, auditable edge path and allow the cloud backbone to scale independently.

The evidence has four boundaries. First, robustness is established only for the tested delay distributions; an extended disconnection can invalidate even task-level context. Second, paired-frame training uses two backbone passes. Third, a frozen RGB encoder may miss depth, force, or contact cues. Fourth, the real-robot result is a small success-rate pilot on one platform with a workstation-class edge device, not evidence of embedded efficiency or broad hardware generalization. Timestamped features and a local fallback policy remain necessary for disconnection handling.

Conclusion

CloudEdgeVLA enables a cloud VLA and edge controller to run without blocking synchronization. Paired-frame training makes stale cloud features and current local vision jointly predictive of the current action. On LIBERO, this design preserves high success with a 40-step uniform-delay window while substantially outperforming both the evaluated single-path VLAs and VLASH under long staleness; a small real-robot pilot provides a physical feasibility check. Together, the results frame latency tolerance as a learned interface property rather than only an inference-scheduling problem.

References

Achille, A.; and Soatto, S. 2018. Emergence of Invariance and Disentanglement in Deep Representations. *Journal of Machine Learning Research*, 19(50): 1–34.

Black, K.; Brown, N.; Driess, D.; Esmail, A.; Equi, M.; Finn, C.; Fusai, N.; Groom, L.; Hausman, K.; Ichter, B.; et al. 2024. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*.

Chen, H.; Liu, J.; Gu, C.; Liu, Z.; Zhang, R.; Li, X.; He, X.; Guo, Y.; Fu, C.-W.; Zhang, S.; and Heng, P.-A. 2025. Fast-in-Slow: A Dual-System Foundation Model Unifying Fast Manipulation within Slow Reasoning. *arXiv preprint arXiv:2506.01953*.

Chi, C.; Feng, S.; Du, Y.; Xu, Z.; Cousineau, E.; Burchfiel, B. C. M.; and Song, S. 2023. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Proceedings of Robotics: Science and Systems*.

Chinchali, S.; Sharma, A.; Harrison, J.; Elhafsi, A.; Kang, D.; Pergament, E.; Cidon, E.; Katti, S.; and Pavone, M. 2019. Network Offloading Policies for Cloud Robotics: A Learning-Based Approach. In *Proceedings of Robotics: Science and Systems*.

Ghosh, D.; Walke, H. R.; Pertsch, K.; Black, K.; Mees, O.; Dasari, S.; Hejna, J.; Kreiman, T.; Xu, C.; Luo, J.; et al. 2024. Octo: An Open-Source Generalist Robot Policy. In *Proceedings of Robotics: Science and Systems*.

Han, B.; Kim, J.; and Jang, J. 2024. A Dual Process VLA: Efficient Robotic Manipulation Leveraging VLM. *arXiv preprint arXiv:2410.15549*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Kahneman, D. 2011. *Thinking, Fast and Slow*. Farrar, Straus and Giroux.

Karamzade, A.; Kim, K.; Kalsi, M.; and Fox, R. 2024. Reinforcement Learning from Delayed Observations via World Models. In *Reinforcement Learning Conference*.

Kehoe, B.; Patil, S.; Abbeel, P.; and Goldberg, K. 2015. A Survey of Research on Cloud Robotics and Automation. *IEEE Transactions on Automation Science and Engineering*, 12(2): 398–409.

Kim, M. J.; Finn, C.; and Liang, P. 2025. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. *arXiv preprint arXiv:2502.19645*.

Kim, M. J.; Pertsch, K.; Karamcheti, S.; Xiao, T.; Balakrishna, A.; Nair, S.; Rafailov, R.; Foster, E. P.; Sanketi, P. R.; Vuong, Q.; et al. 2025. OpenVLA: An Open-Source Vision-Language-Action Model. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, 2679–2713.

Liu, B.; Zhu, Y.; Gao, C.; Feng, Y.; Liu, Q.; Zhu, Y.; and Stone, P. 2023. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. In *Advances in Neural Information Processing Systems*, volume 36.

Saxena, S.; Sharma, M.; and Kroemer, O. 2023. Multi-Resolution Sensing for Real-Time Control with Vision-Language Models. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, 2210–2228.

Sendai, K.; Alvarez, M.; Matsushima, T.; Matsuo, Y.; and Iwasawa, Y. 2025. Leave No Observation Behind: Real-Time Correction for VLA Action Chunks. *arXiv preprint arXiv:2509.23224*.

Shukor, M.; Aubakirova, D.; Capuano, F.; Kooijmans, P.; Palma, S.; Zouitine, A.; Aractingi, M.; Pascal, C.; Russi, M.; Marafioti, A.; et al. 2025. SmolVLA: A Vision-Language-Action Model for Affordable and Efficient Robotics. *arXiv preprint arXiv:2506.01844*.

Tang, J.; Sun, Y.; Zhao, Y.; Yang, S.; Lin, Y.; Zhang, Z.; Hou, J.; Lu, Y.; Liu, Z.; and Han, S. 2025. VLASH: Real-Time VLAs via Future-State-Aware Asynchronous Inference. *arXiv preprint arXiv:2512.01031*.

Wan, J.; Tang, S.; Yan, H.; Li, D.; Wang, S.; and Vasilakos, A. V. 2016. Cloud Robotics: Current Status and Open Issues. *IEEE Access*, 4: 2797–2807.

Wang, W.; Han, D.; Luo, X.; and Li, D. 2024. Addressing Signal Delay in Deep Reinforcement Learning. In *International Conference on Learning Representations*.

Wang, Y.; Li, X.; Wang, W.; Zhang, J.; Li, Y.; Chen, Y.; Wang, X.; and Zhang, Z. 2025. Unified Vision-Language-Action Model. *arXiv preprint arXiv:2506.19850*.

Xiao, T.; Jang, E.; Kalashnikov, D.; Levine, S.; Ibarz, J.; Hausman, K.; and Herzog, A. 2020. Thinking While Moving: Deep Reinforcement Learning with Concurrent Control. In *International Conference on Learning Representations*.

Yan, S.; Wang, G.; Liu, Q.; Meng, W.; Yang, J.; Yao, C.; Feng, F.; Ma, X.; Zhao, Y.; and Han, Y. 2026. Acting While Understanding: Asynchronous Semantic-Action Decoupling for Real-Time Vision-Language-Action Models. *arXiv preprint arXiv:2606.15285*.

Zhai, X.; Mustafa, B.; Kolesnikov, A.; and Beyer, L. 2023. Sigmoid Loss for Language Image Pre-Training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 11975–11986.

Zhang, J.; Guo, Y.; Chen, X.; Wang, Y.-J.; Hu, Y.; Shi, C.; and Chen, J. 2025. HiRT: Enhancing Robotic Control with Hierarchical Robot Transformers. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, 933–946.

Zhao, T. Z.; Kumar, V.; Levine, S.; and Finn, C. 2023. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proceedings of Robotics: Science and Systems*.

Zitkovich, B.; Yu, T.; Xu, S.; Xu, P.; Xiao, T.; Xia, F.; Wu, J.; Wohlhart, P.; Welker, S.; Wahid, A.; et al. 2023. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, 2165–2183.

Table 4: **Fixed-offset offline diagnostics on LIBERO-Spatial.** OFT denotes original OpenVLA-OFT and CE denotes aligned CloudEdgeVLA at 120k steps. All entries are means over 80 shared demonstration states. The transfer gain is undefined at d=0 because both drift terms are zero.

	D_h		κ		D_a		E_{demo}	
d	OFT	CE	OFT	CE	OFT	CE	OFT	CE
0	0.000	0.000	–	–	0.000	0.000	0.020	0.023
1	0.028	0.014	1.436	1.500	0.040	0.021	0.041	0.028
3	0.103	0.050	1.054	0.602	0.109	0.030	0.107	0.035
5	0.168	0.077	0.998	0.442	0.167	0.034	0.165	0.040
8	0.239	0.102	1.005	0.351	0.240	0.036	0.238	0.041
10	0.274	0.114	1.024	0.328	0.281	0.037	0.279	0.042
15	0.343	0.137	1.061	0.280	0.365	0.038	0.363	0.041
20	0.391	0.160	1.082	0.295	0.423	0.047	0.421	0.048

A Offline Mechanism-Diagnostic Protocol

This supplement expands the backbone–head analysis in the main paper using the aligned CloudEdgeVLA checkpoint at 120k training steps and the original OpenVLA-OFT checkpoint. We use one demonstration episode from each of the ten LIBERO-Spatial tasks and select eight valid current timesteps per task, giving 80 shared evaluation states. For this offline diagnostic, we replace the cloud image by a frame at each deterministic offset

$$d \in \{0, 1, 3, 5, 8, 10, 15, 20\}$$

environment steps. These fixed offsets resolve how feature and action drift develop within the W=21 paired-frame training window; they are mechanism probes, not additional closed-loop delay conditions. The closed-loop evaluation in the main paper instead samples observation age as $k \sim \text{Uniform}\{1, \ldots, d_{\max}\}$. Proprioception remains current for both models, and CloudEdgeVLA additionally receives the current image through its edge encoder. Predictions are normalized eight-step action chunks with seven action dimensions.

Let h_t denote the action-token hidden states produced from the current image and h_{t-d} those produced from the delayed image. We report

$$D_h(d) = \mathbb{E}\left[1 - \cos(h_t, h_{t-d})\right], \tag{10}$$

$$D_a(d) = \mathbb{E}\left[\left|\hat{a}_t^{(0)} - \hat{a}_t^{(d)}\right|\right], \tag{11}$$

$$\kappa(d) = \frac{D_a(d)}{D_h(d) + \epsilon}, \tag{12}$$

$$E_{\text{demo}}(d) = \mathbb{E}\left[\left|\hat{a}_t^{(d)} - a_t\right|\right]. \tag{13}$$

D_h measures backbone representation staleness, κ measures how strongly the head transfers residual hidden-state drift into the action output, and E_{demo} prevents a nearly constant predictor from appearing robust merely because its output changes little. Confidence intervals in the figures are computed across the ten task means rather than across temporally correlated frames.

Table 4 gives the numerical counterpart to the main paper's mechanism figure. At d=20, CloudEdgeVLA reduces D_h, κ, D_a, and E_{demo} by 59.2%, 72.7%, 88.9%, and 88.5%, respectively.

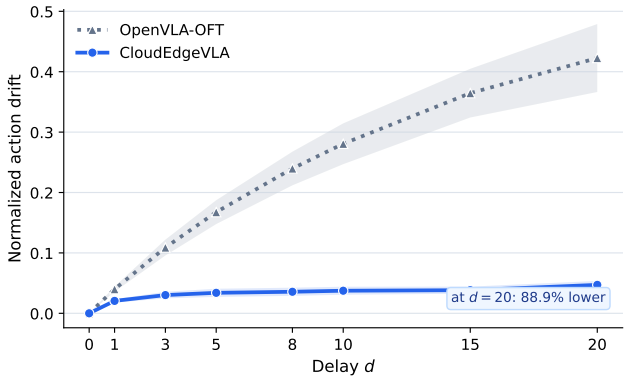

Figure 8: **Action drift under delayed vision.** Bands are task-level 95% confidence intervals. At d=20, CloudEdgeVLA reduces drift from its fresh prediction by 88.9%.

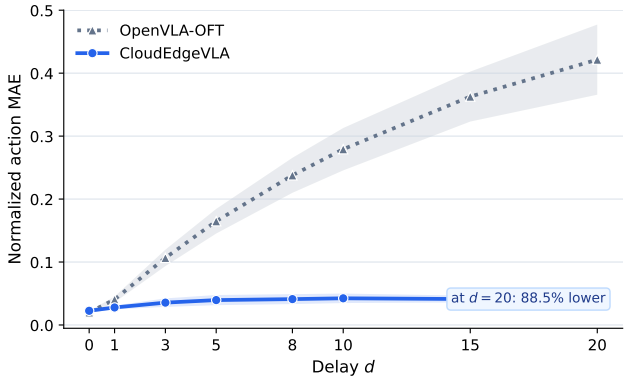

Figure 9: **Normalized demonstration MAE under delayed vision.** CloudEdgeVLA incurs a small fresh-accuracy cost but becomes better by d=1 and reduces MAE by 88.5% at d=20.

B Offline Action-Output Robustness

Figures 8 and 9 expose the tradeoff hidden by a drift-only metric. At d=0, demonstration MAE is 0.023 for CloudEdgeVLA and 0.020 for OpenVLA-OFT; CloudEdgeVLA becomes better by d=1, and the gap then widens over the tested range. Thus delay training sacrifices a small amount of fresh accuracy for substantially greater robustness rather than simply producing an insensitive action head.

C Per-Task Feature-to-Action Geometry

Figures 10–12 show that CloudEdgeVLA's lower aggregate error is not driven by a small subset of tasks. At d=20, it has lower demonstration action MAE on all ten tasks. The absolute task-level reduction ranges from 0.278 to 0.566 normalized MAE. The feature-to-action plot also reveals a qualitative difference in propagation: OpenVLA-OFT action drift grows approximately with its hidden-state distance, whereas

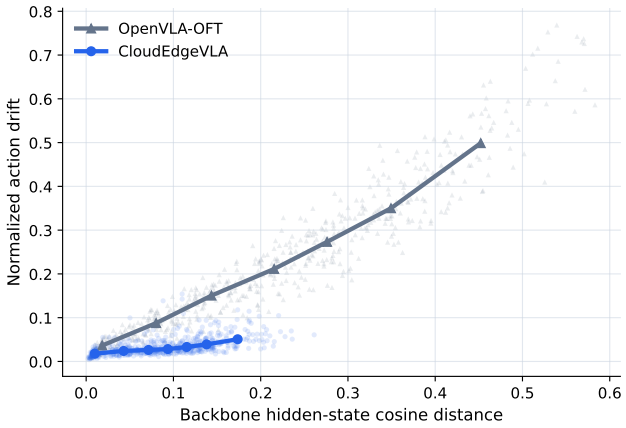

Figure 10: **Feature-to-action delay geometry.** Action drift grows with hidden-state distance for OpenVLA-OFT but remains comparatively flat for CloudEdgeVLA.

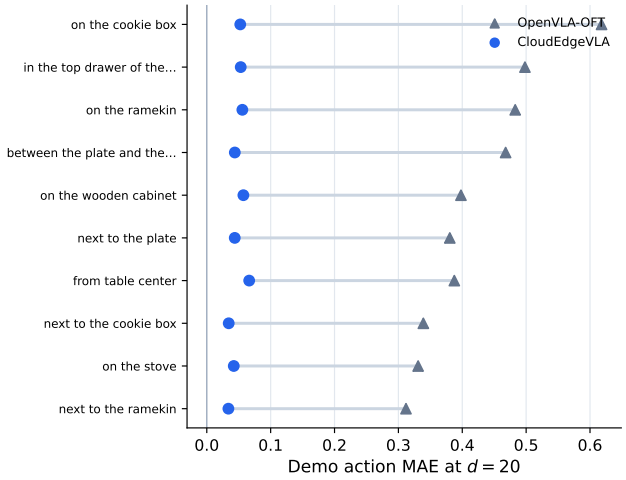

Figure 11: **Per-task delayed action error at** d=20**.** CloudEdgeVLA has lower demonstration MAE on every LIBERO-Spatial task.

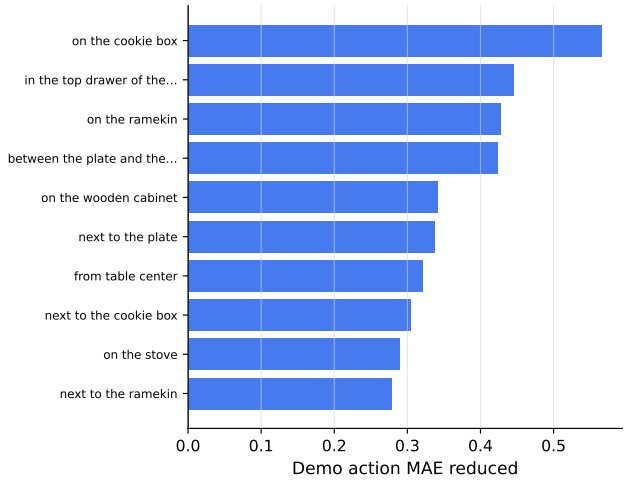

Figure 12: **Per-task robustness gain at** d=20**.** Positive bars denote lower demonstration MAE for CloudEdgeVLA.

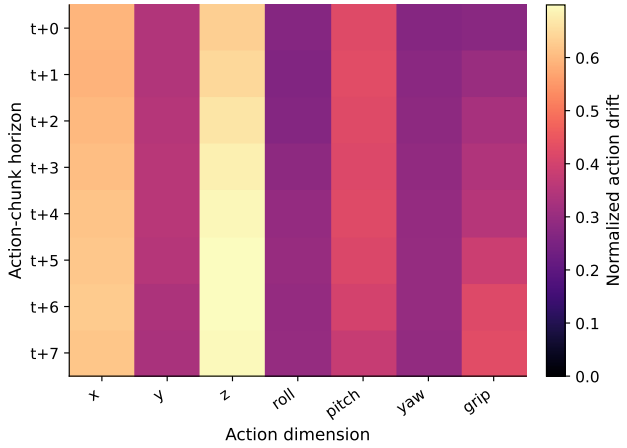

Figure 13: **OpenVLA-OFT action-chunk drift at** d=20**.** Cells show mean absolute drift from the fresh prediction.

CloudEdgeVLA's action drift remains comparatively flat as its backbone representation ages.

D Action-Chunk Decomposition

The reduction is distributed across the complete action chunk rather than concentrated at the first control step. Comparing Figures 13 and 14, CloudEdgeVLA has less drift throughout the horizon. Figure 15 confirms suppression in all 56 horizon–dimension cells. The mean suppression is 0.376, with cell values ranging from 0.214 to 0.651. Translation in x and z shows the largest average reductions, but rotation and gripper dimensions also improve throughout the horizon.

E Counterfactual Backbone and Edge Age

For CloudEdgeVLA alone, we independently vary the age of the image supplied to the cloud backbone, d_h, and the age of the image supplied to the edge vision encoder, d_z. This produces a two-dimensional counterfactual surface rather than assuming that the two paths always share the same observation age.

The nearly horizontal surface in Figure 16 is an important negative result. At d_h=20, changing the edge input from equally stale to current reduces mean action drift from 0.047104 to 0.047097, an edge-rescue fraction of only 0.03%. The correction vector has mean cosine alignment 0.030 with the action change needed to recover the fresh prediction. Consequently, this 120k checkpoint does not provide strong evidence that current edge vision directly repairs stale cloud features. Its measured advantage is instead dominated by a more stable backbone and a head that attenuates residual hidden-state drift. This distinction constrains the mechanism claim without weakening the observed end-to-end delay robustness.

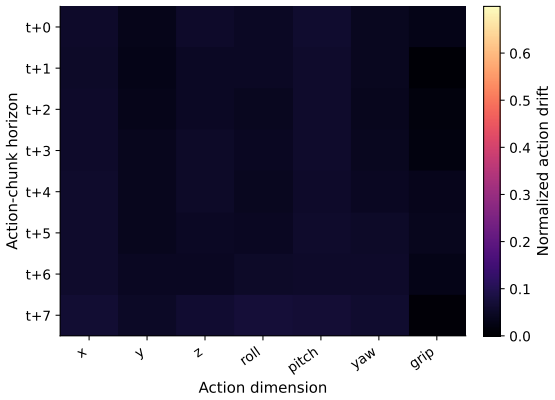

Figure 14: **CloudEdgeVLA action-chunk drift at d=20.** The shared color scale matches Figure 13.

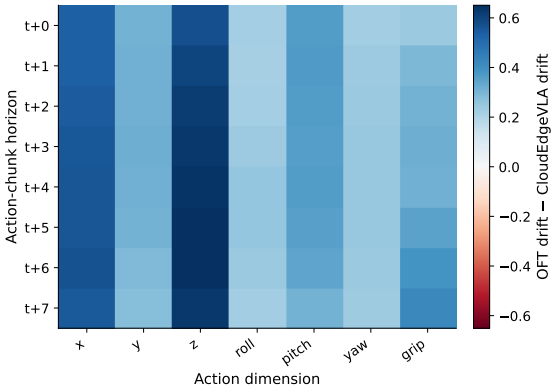

Figure 15: **Action drift suppressed by delay training.** Positive values denote OpenVLA-OFT drift minus CloudEdgeVLA drift.

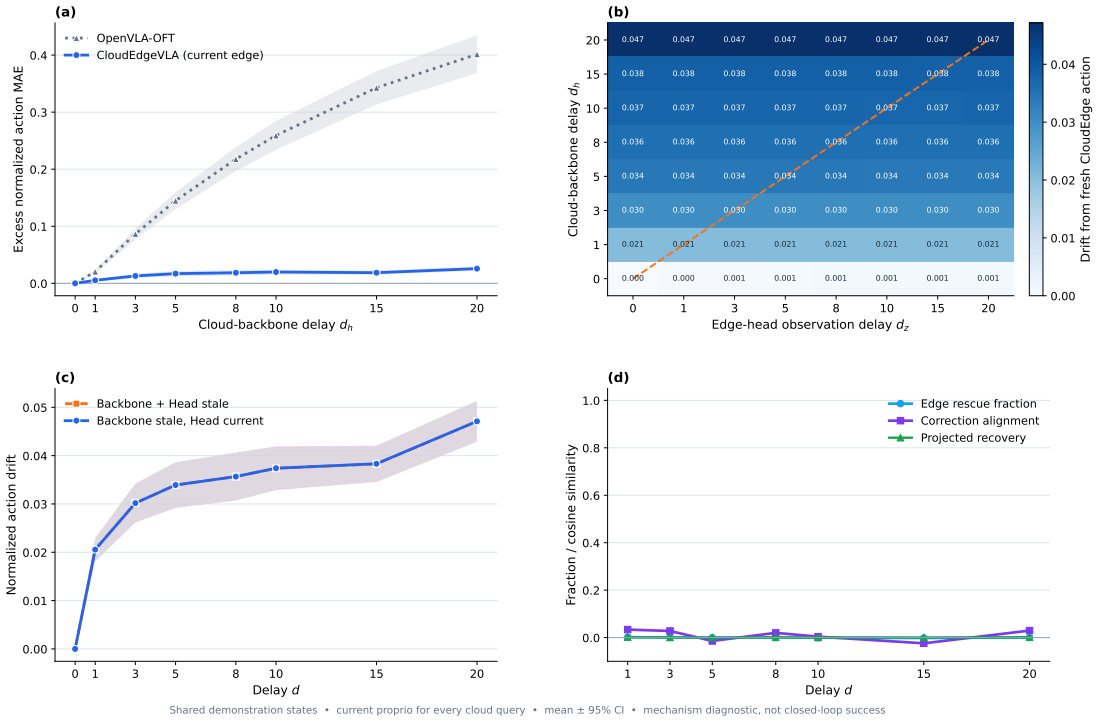

Figure 16: **Counterfactual audit of backbone and edge observation age.** Panel (a) reports delay-induced demonstration error. Panel (b) independently varies backbone delay d_h and edge-observation delay d_z. Panel (c) compares a stale backbone with either stale or current edge features. Panel (d) measures the magnitude and direction of the resulting edge correction. This is a mechanism diagnostic, not a closed-loop success evaluation.

F Qualitative Rollout

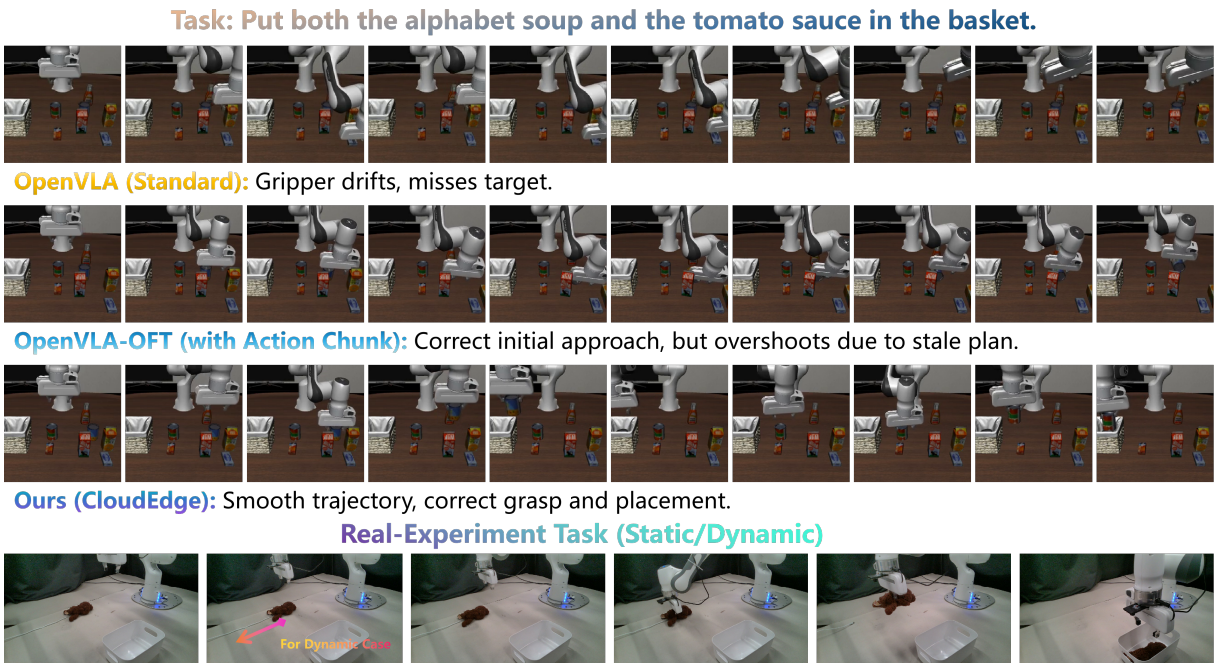

Figure 17: **Qualitative rollout examples.** The top three sequences compare OpenVLA, OpenVLA-OFT, and CloudEdgeVLA on the same LIBERO task. The bottom sequence shows CloudEdgeVLA on the real-robot task; arrows mark the externally displaced target. These examples complement rather than replace the closed-loop success rates.

G Scope and Limitations

These offline diagnostics complement, but do not replace, closed-loop success: one episode and eight sampled timesteps per task provide only ten task-level samples. Because the checkpoints differ in architecture and training objective, the comparison characterizes the complete systems rather than isolating one component. Delay d is measured in environment steps, and its wall-clock duration depends on the control frequency and network pipeline; the reported offsets therefore should not be interpreted as hardware-independent millisecond latency.