LATENCY-TOLERANT CLOUD-EDGE COLLABORATIVE VISION-LANGUAGE-ACTION MODELS VIA EMERGENT REPRESENTATIONAL SPECIALIZATION

Anonymous authors
Paper under double-blind review

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, driving emergent specialization. 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.

1 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., 2025b; 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 et al., 2023); DP-VLA and HiRT separate slow semantic reasoning from fast visuomotor control (Han et al., 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.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

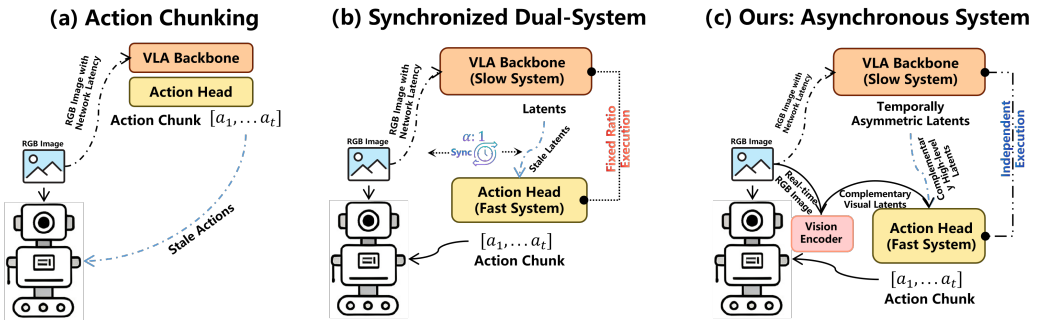


Figure 1: **Conceptual comparison of dual-system paradigms.** (a) Action-chunk re-planning assumes environment stasis during open-loop execution, compounding staleness. (b) Synchronized fast-slow pipelines require fixed frequency ratio α and bounded latency–conditions rarely met in networked deployments. (c) Our asynchronous paradigm: the cloud produces delay-resilient 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.

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. As illustrated in Figure 1, our edge never stalls for cloud updates; it always consumes the latest arrived feature, even when that feature corresponds to an observation captured many steps ago.

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 propose paired-frame dual-path training: by supervising both fresh and staled cloud features against the identical current-step action target, our objective induces emergent representational specialization without hand-crafted invariance regularization. The cloud learns delay-robust task context, and the edge compensates for instantaneous local state changes.
3. We conduct systematic experiments on LIBERO benchmarks under controlled uniform network delay, including ablations disentangling the respective robustness contributions of cloud backbone representations and edge-side correction. We further validate physical feasibility via a small-scale real-robot Franka pick-and-place pilot study.

2 Related Work

2.1 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., 2025b). 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 et al., 2025a; 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.

2.2 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 et al., 2023; Han et al., 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.

2.3 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).

3 Method

3.1 Problem Formulation

We formalize the asynchronous cloud-edge VLA deployment problem from the edge execution perspective. At environment step t:

- The **edge** captures real-time local observation o_t.
- The latest cloud representation available at the edge at step t is computed from an observation collected k steps earlier: $h_{t-k} = f_\theta(o_{t-k}, \ell)$, where ℓ denotes the language instruction and f_θ is the cloud-hosted VLA backbone.
- The **edge** generates action a_t using this stale cloud representation h_{t-k} together with its real-time local observation o_t.

Here k denotes the total staleness: the sum of uplink transmission, cloud computation, downlink transmission and queueing delays. We jointly optimize the cloud backbone parameters θ and edge action head parameters ϕ, such that the predicted action $\hat{a}_t = g_\phi\big(h_{t-k}, v_\psi(o_t)\big)$ maximizes task success under the temporal mismatch between the stale cloud representation h_{t-k} and the current local observation o_t. v_ψ denotes the lightweight local vision encoder on edge hardware.

3.2 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

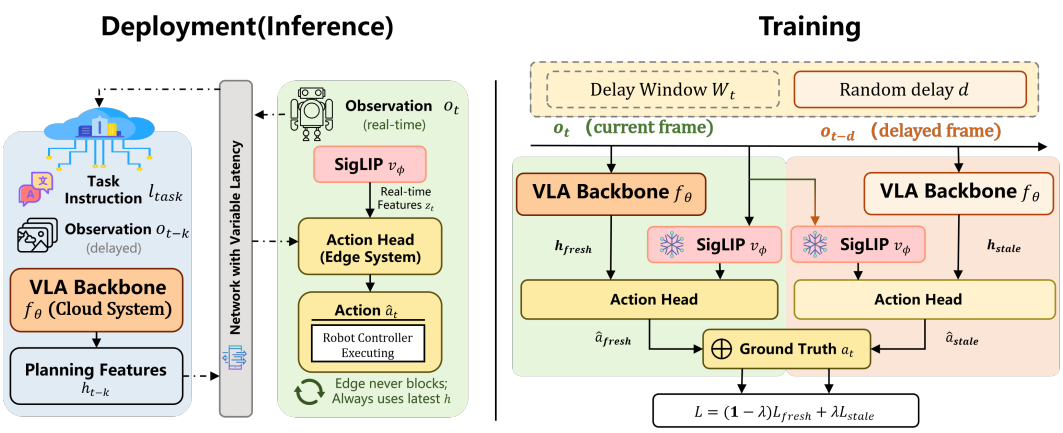


Figure 2: **System architecture and training pipeline. Left:** At deployment time, the cloud VLA backbone produces planning features from a delayed observation, while the edge vision encoder extracts real-time features from the current observation. The action head fuses both to produce actions without 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.

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 *fresh 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) received cloud planning embeddings h_{t-k} with the real-time edge vision features z_t to predict continuous actions:

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

The edge head projects the cloud planning latents and the local visual features into the latent space 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.

3.3 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: (1) *Current frame* o_t (the latest observation) and (2) *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.

3.4 Deployment Protocol

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

3.5 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 therefore *encourages* invariance to temporal displacement, consistent with the broader connection between nuisance variation and invariant representations (Achille & 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. At inference time, the model does not require explicit delay magnitude as input. The edge simply uses whichever cloud feature is most recently available, regardless of how many steps old it is. This is a key distinction from methods that condition policies on measured latency.

4 Experiments

4.1 Implementation

We build on OpenVLA-OFT (Kim et al., 2025a), 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. Notably, our paired-frame training is performed exclusively with a maximum delay window $d_{\max} = 20$, yet we test generalization to larger staleness values up to $d_{\max} = 40$ at evaluation time.

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 three seeds.

	$d_{\max}$=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	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	**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	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	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**

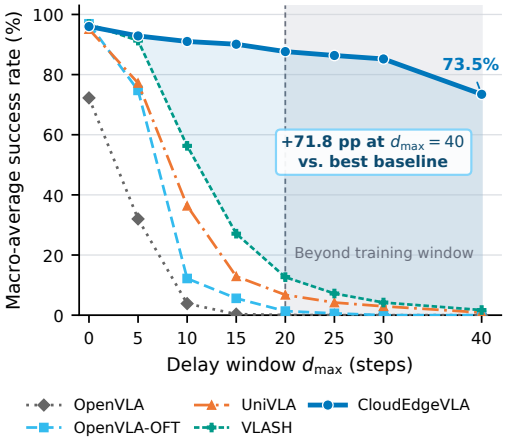


Figure 3: **Closed-loop uniform-delay sweep on LIBERO.** Each point averages Spatial, Object, Goal, and Long. The shaded range $d_{\max}$>20 lies beyond training. At $d_{\max}$=40, CloudEdgeVLA averages 73.5% success, 71.8 points above VLASH.

4.2 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. We compare single-path OpenVLA (Kim et al., 2025b), OpenVLA-OFT (Kim et al., 2025a), and UniVLA (Wang et al., 2025); future-state-aware asynchronous VLASH (Tang et al., 2025); and CloudEdgeVLA. We feed delayed inputs to the policies for evaluation without changing their inference paths.

Evaluation Conditions. We evaluate under: (1) *No delay ($d_{\max}$=0)*, the synchronous reference. (2) *Uniform delay ($k \sim \mathrm{Uniform}\{1, \ldots, d_{\max}\}$)*, 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, and we report the mean and standard deviation across seeds.

4.3 Delay Robustness

Figures 3 and 4 summarize macro-averaged outcomes across the four LIBERO suites, while Table 1 presents detailed per-suite performance. At $d_{\max}$=10, OpenVLA achieves at most 6.8% success, and OpenVLA-OFT tops out at 26.2%. Although UniVLA degrades comparatively slowly on the Goal and Long suites, reaching 48.2% and 35.0% respectively, its performance collapses to no more than 3.0% under the heavy $d_{\max}$=40 delay setting. VLASH delivers competitive performance under moderate delay, obtaining 45.6–62.8% success at $d_{\max}$=10, yet its success rate plummets to merely 0.0–6.4% when delay increases to $d_{\max}$=40. By contrast, CloudEdgeVLA maintains 76.5% of its zero-delay synchronous success rate at $d_{\max}$=40, substantially outperforming all other baselines.

Under $d_{\max} = 40$ uniform delay, CloudEdgeVLA achieves success rates of 76.4% (Spatial), 75.6% (Object), 78.0% (Goal), and 63.8% (Long). Relative to zero-delay performance, the success drop is

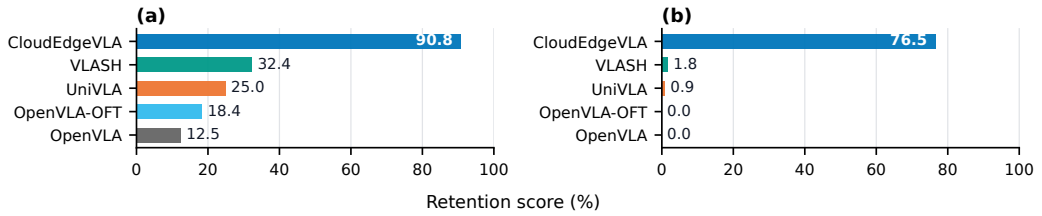


Figure 4: **Closed-loop delay-retention summaries.** Metrics are computed from each method's four-suite macro-average curve. (a) Normalized area under the uniform-delay-window curve. (b) Synchronous success retained at $d_{\max}$=40. CloudEdgeVLA achieves 90.8% and 76.5%, respectively; VLASH reaches 32.4% and 1.8%.

Table 2: **Ablation and real-robot results.**

(a) Ablation: Vision encoder and loss components on LIBERO-Object under uniform delay $d_{\max}$=10.

Variant	Success Rate (%)
No vision encoder	80.6
SigLIP-Base	94.2
SigLIP-SO400M	94.8
$\mathcal{L}_{\text{fresh}}$ only	84.6
$\mathcal{L}_{\text{stale}}$ only	92.4
$\mathcal{L}_{\text{fresh}} + \mathcal{L}_{\text{stale}}$	94.2

(b) Real-robot task success (%). Cells report static/dynamic success over 10 trials per variant. Round-trip time (RTT) is the emulated delay added to the native serving pipeline.

	Added network RTT (ms)		
Method	**0**	**400**	**1000**
VLASH	100/90	60/30	0/0
CloudEdgeVLA	**100/90**	**90/90**	**80/70**

21.5, 22.2, 18.5, 27.9 percentage points respectively. 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. Notably, our model is only trained with a maximum training delay of $d_{\max} = 20$, yet it retains strong task performance under the more challenging test-time condition $d_{\max} = 40$. Even in the worst case on the Long suite, CloudEdgeVLA remains 63.3 points above the strongest baseline at $d_{\max} = 40$. This demonstrates that our paired-frame training objective yields meaningful generalization to larger staleness magnitudes not directly seen during training.

4.4 Ablation Studies

We isolate the contribution of current edge vision, encoder capacity, and the two loss paths on the LIBERO-Object suite. Table 2a reports success under uniform delay with $d_{\max}$=10; 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 80.6% to 94.2%, a 13.6-point gain over stale cloud features alone. Replacing it with the substantially larger SigLIP-SO400M reaches 94.8%, only 0.6 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 92.4%, 7.8 points above fresh-only training, confirming that explicit exposure to delayed features drives most of the robustness. Joint supervision restores the full model to 94.2%, another 1.8-point gain over stale-only training; the fresh path therefore provides complementary grounding rather than replacing delay exposure.

4.5 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

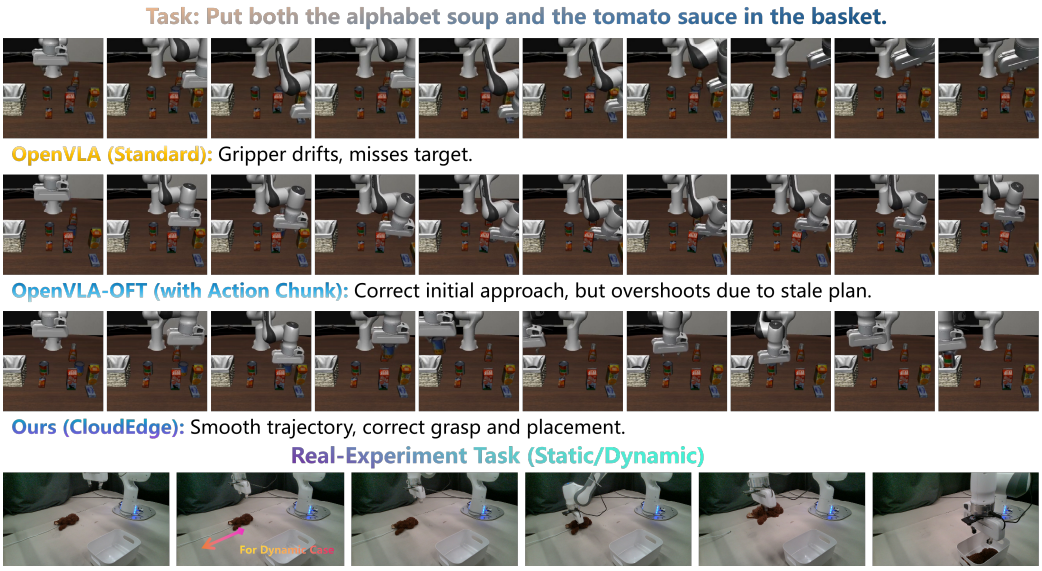


Figure 5: **Qualitative rollouts in simulation and on the real robot.** The top three sequences compare OpenVLA, OpenVLA-OFT, and CloudEdgeVLA on the same LIBERO task of placing two specified objects in the basket. The bottom sequence shows CloudEdgeVLA executing the real-robot task of placing a toy bear into a box; the arrows mark the externally displaced target in the dynamic setting.

means no *additional* RTT, not zero camera-to-action latency. Camera acquisition, preprocessing, transport, request scheduling, and model inference remain present in every profile.

The results are summarized in Table 2b. With zero added RTT, CE-VLA and VLASH achieve 100%/90% success on static and dynamic tasks. At 400 ms emulated RTT, CE-VLA maintains 90%/90% success, outperforming VLASH (60%/30%) by 30 and 60 percentage points. At 1000 ms RTT, CE-VLA retains 80%/70% success, while VLASH degrades completely to 0%/0%. This widening performance gap under elevated latency further validates that our cloud-edge design preserves manipulation performance even under severe network round-trip delays. Figure 5 complements these success rates with representative simulation and real-robot rollouts.

5 Representation Analysis: Backbone-Head Delay Mechanism

To verify the effectiveness of our delay-robust feature learning and uncover the underlying robustness mechanism, we perform controlled representation analysis comparing the original OpenVLA-OFT checkpoint against our CloudEdgeVLA checkpoint. We evaluate over 80 shared demonstration timesteps sampled across the ten LIBERO-Spatial tasks. For each timestep, we feed the cloud backbone a frame delayed by d steps. We quantify backbone representation drift and end-to-end action drift via

$$\begin{aligned} D_{hidden}(d) &= \mathbb{E}\left[1 - \cos(h_t, h_{t-d})\right], \\ D_{action}(d) &= \mathbb{E}\left[\left|\hat{a}_t^{(0)} - \hat{a}_t^{(d)}\right|\right], \end{aligned} \tag{8}$$

and further define the staleness transfer gain

$$\kappa(d) = \frac{D_{action}(d)}{D_{hidden}(d)}. \tag{9}$$

A lower D_{hidden} indicates the backbone yields temporally stable features under delayed observations, while a lower κ shows the edge head constrains residual backbone drift from propagating into action outputs. The supplementary material plots D_{action} and normalized MAE versus demonstrated action chunks, ruling out the trivial hypothesis that gains stem from an overly insensitive policy emitting near-constant actions.

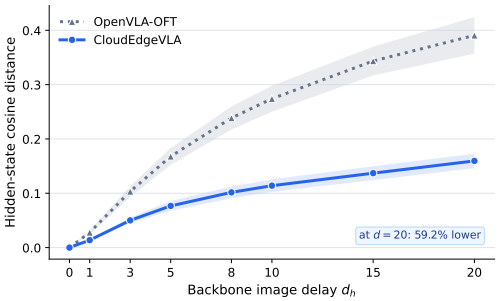


Figure 6: **Backbone representation staleness on LIBERO-Spatial.** Over 80 shared states from ten tasks, CloudEdgeVLA reduces backbone drift D_{hidden} by 59.2% at d=20; bands are task-level 95% confidence intervals.

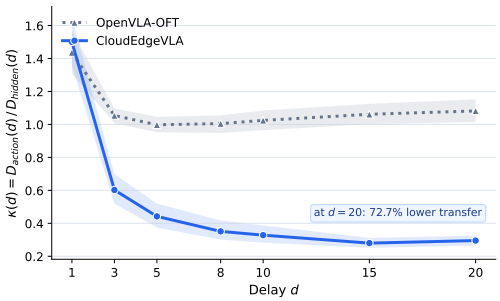


Figure 7: **Head staleness-transfer gain on LIBERO-Spatial.** CloudEdgeVLA reduces κ, the action drift transferred per unit backbone drift, by 72.7% at d=20.

Figures 6 and 7 decompose robustness sources across the cloud-edge interface. At d=20, CloudEdgeVLA reduces backbone drift D_{hidden} from 0.391 to 0.160 and lowers staleness-transfer gain κ from 1.082 to 0.295. Together these two factors suppress end-to-end action drift D_{action} from 0.423 to 0.047, with demonstration MAE falling from 0.421 to 0.048. The findings validate two complementary properties of our learned interface: the cloud backbone learns delay-robust task representations, and the lightweight edge head mitigates leftover drift within received cloud features. The supplement further visualizes these curves, the small accuracy penalty on fresh inputs, per-task consistency, and action-chunk effects.

6 Discussion and Limitations

CloudEdgeVLA re-designs the slow-fast cloud-edge interface. Rather than sending complete temporally aligned action plans from cloud to edge, we transmit delay-resilient task-level representations. The edge then grounds these representations using real-time local visual observations. The state-sensitive decision-making resides on local edge hardware. Unlike action-chunk open-loop execution, our setup maintains visual feedback closure on the edge. This yields a compact, auditable edge stack, while cloud-hosted foundation models are free to scale independently without imposing heavier computation requirements on the robot side.

The evidence has four boundaries. First, robustness is established only for the tested delay distributions; an extended disconnection can invalidate even task-level context, which could be mitigated by local fallback policies. Second, paired-frame training uses two backbone passes per training sample, which increases training compute overhead; future work may adopt feature-caching to reduce cost. Third, a frozen RGB encoder may miss depth, force, or contact cues that would benefit dynamic manipulation, which could be addressed by incorporating multi-modal edge perception. Fourth, our current setup transmits full-dimensional backbone planning features across the network, introducing non-trivial downlink bandwidth overhead; compact learned representations, quantization or sparsification represent promising avenues for improvement.

7 Conclusion

CloudEdgeVLA enables a cloud-hosted VLA and lightweight edge controller to operate without blocking synchronization. Our paired-frame training objective makes stale cloud-derived features and current local vision jointly predictive of the current robot action. This design retains high task success under a 40-step uniform-delay window, twice the maximum staleness encountered during training. It demonstrates generalization to staleness magnitudes outside the training distribution and substantially outperforms all evaluated VLA baselines. The real-robot experiments further provides a physical feasibility check. Collectively, our findings frame latency tolerance for cloud-edge robot control as a learned property, rather than merely an inference-scheduling problem.

REFERENCES

Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. *Journal of Machine Learning Research*, 19(50):1–34, 2018. URL https://jmlr.org/papers/v19/17-646.html.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Hao Chen, Jiaming Liu, Chenyang Gu, Zhuoyang Liu, Renrui Zhang, Xiaoqi Li, Xiao He, Yandong Guo, Chi-Wing Fu, Shanghang Zhang, and Pheng-Ann Heng. Fast-in-slow: A dual-system foundation model unifying fast manipulation within slow reasoning. *arXiv preprint arXiv:2506.01953*, 2025. URL https://arxiv.org/abs/2506.01953.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin C. M. Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026.

Sandeep Chinchali, Apoorva Sharma, James Harrison, Amine Elhafsi, Daniel Kang, Evgenya Pergament, Eyal Cidon, Sachin Katti, and Marco Pavone. Network offloading policies for cloud robotics: A learning-based approach. In *Proceedings of Robotics: Science and Systems*, 2019. doi: 10.15607/RSS.2019.XV.063.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, et al. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.090.

ByungOk Han, Jaehong Kim, and Jinhyeok Jang. A dual process VLA: Efficient robotic manipulation leveraging VLM. *arXiv preprint arXiv:2410.15549*, 2024. URL https://arxiv.org/abs/2410.15549.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

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

Armin Karamzade, Kyungmin Kim, Montek Kalsi, and Roy Fox. Reinforcement learning from delayed observations via world models. In *Reinforcement Learning Conference*, 2024. URL https://openreview.net/forum?id=ToftWbZfYr.

Ben Kehoe, Sachin Patil, Pieter Abbeel, and Ken Goldberg. A survey of research on cloud robotics and automation. *IEEE Transactions on Automation Science and Engineering*, 12(2):398–409, 2015. doi: 10.1109/TASE.2014.2376492.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025a. URL https://arxiv.org/abs/2502.19645.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, et al. 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*, pp. 2679–2713, 2025b. URL https://proceedings.mlr.press/v270/kim25c.html.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

Saumya Saxena, Mohit Sharma, and Oliver Kroemer. 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*, pp. 2210–2228, 2023. URL https://proceedings.mlr.press/v229/saxena23a.html.

Kohei Sendai, Maxime Alvarez, Tatsuya Matsushima, Yutaka Matsuo, and Yusuke Iwasawa. Leave no observation behind: Real-time correction for VLA action chunks. *arXiv preprint arXiv:2509.23224*, 2025. URL https://arxiv.org/abs/2509.23224.

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. *arXiv preprint arXiv:2506.01844*, 2025. URL https://arxiv.org/abs/2506.01844.

Jiaming Tang, Yufei Sun, Yilong Zhao, Shang Yang, Yujun Lin, Zhuoyang Zhang, James Hou, Yao Lu, Zhijian Liu, and Song Han. VLASH: Real-time VLAs via future-state-aware asynchronous inference. *arXiv preprint arXiv:2512.01031*, 2025. URL https://arxiv.org/abs/2512.01031.

Jiafu Wan, Shenglong Tang, Hehua Yan, Di Li, Shiyong Wang, and Athanasios V. Vasilakos. Cloud robotics: Current status and open issues. *IEEE Access*, 4:2797–2807, 2016. doi: 10.1109/ACCESS.2016.2574979.

Wei Wang, Dongqi Han, Xufang Luo, and Dongsheng Li. Addressing signal delay in deep reinforcement learning. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Z8UfDs4J46.

Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. Unified vision-language-action model. *arXiv preprint arXiv:2506.19850*, 2025. URL https://arxiv.org/abs/2506.19850.

Ted Xiao, Eric Jang, Dmitry Kalashnikov, Sergey Levine, Julian Ibarz, Karol Hausman, and Alexander Herzog. Thinking while moving: Deep reinforcement learning with concurrent control. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJexHkSFPS.

Shenhao Yan, Ge Wang, Qi Liu, Weilin Meng, Jiahao Yang, Chengsi Yao, Fan Feng, Xiaoguang Ma, Yiming Zhao, and Yatong Han. Acting while understanding: Asynchronous semantic-action decoupling for real-time vision-language-action models. *arXiv preprint arXiv:2606.15285*, 2026. URL https://arxiv.org/abs/2606.15285.

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*, pp. 11975–11986, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Zhai_Sigmoid_Loss_for_Language_Image_Pre-Training_ICCV_2023_paper.html.

Jianke Zhang, Yanjiang Guo, Xiaoyu Chen, Yen-Jen Wang, Yucheng Hu, Chengming Shi, and Jianyu Chen. 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*, pp. 933–946, 2025. URL https://proceedings.mlr.press/v270/zhang25b.html.

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

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.