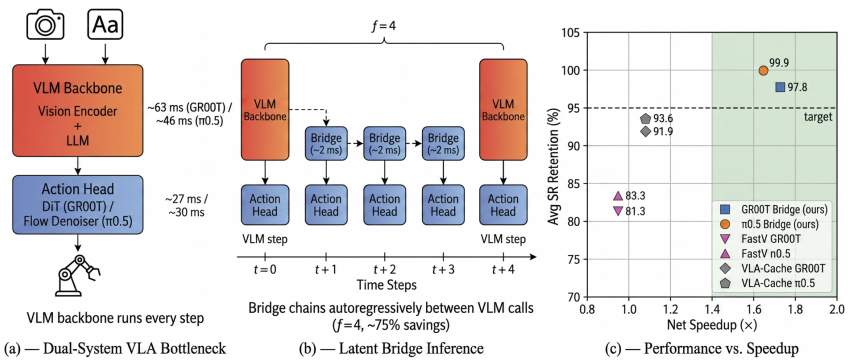
LATENT BRIDGE: FEATURE DELTA PREDICTION FOR EFFICIENT DUAL-SYSTEM VISION-LANGUAGE-ACTION MODEL INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Dual-system Vision-Language-Action (VLA) models achieve state-of-the-art robotic manipulation but are bottlenecked by the VLM backbone, which must execute at every control step while producing temporally redundant features. We propose **Latent Bridge**, a lightweight model that predicts VLM output deltas between timesteps, enabling the action head to operate on predicted outputs while the expensive VLM backbone is called only periodically. We instantiate Latent Bridge on two architecturally distinct VLAs: **GR00T-N1.6** (feature-space bridge) and $\pi_{0.5}$ (KV-cache bridge), demonstrating that the approach generalizes across VLA designs. Our task-agnostic DAgger training pipeline transfers across benchmarks without modification. Across four LIBERO suites, 24 RoboCasa kitchen tasks, and the ALOHA sim transfer-cube task, Latent Bridge achieves **95–100% performance retention** while reducing VLM calls by **50–75%**, yielding **1.55–1.73× net per-episode speedup**.



(a) — Dual-System VLA Bottleneck (b) — Latent Bridge Inference (c) — Performance vs. Speedup

Figure 1: Latent Bridge reduces VLM backbone calls by predicting feature deltas between timesteps at orders-of-magnitude lower latency: 50–75% VLM savings with 95–100% task performance retention across two VLA architectures and diverse benchmarks.

1 INTRODUCTION

Vision-Language-Action (VLA) models have emerged as a promising paradigm for generalist robotic manipulation, leveraging pretrained vision-language models to ground language instructions in visual observations and produce motor commands (Brohan et al., 2023b; Black et al., 2024; Bjorck et al., 2025). Among VLA architectures, *dual-system* designs—where a large VLM backbone produces a feature representation that a separate, lightweight action head decodes into actions—have achieved state-of-the-art results on diverse manipulation benchmarks. Unlike single-system VLAs that autoregressively generate action tokens through the LLM itself (e.g., RT-2 (Brohan et al., 2023b), OpenVLA (Kim et al., 2024)), dual-system VLAs (e.g., GR00T (Bjorck et al., 2025), $\pi_{0.5}$ (Physical Intelligence et al., 2025), Octo (Ghosh et al., 2024)) decompose inference into a **VLM backbone** stage, in which a vision encoder and language model process the observation o_t into features $z_t \in \mathbb{R}^{N \times D}$ and which dominates inference time, and an **action head** stage, in which a lightweight policy network decodes z_t into the action a_t far faster. This two-stage description is an exposition abstraction: it is literal for GR00T (a VLM feature grid feeding a separate DiT head), whereas $\pi_{0.5}$ interleaves the two systems—the action (suffix) tokens cross-attend the backbone's per-layer prefix KV cache at

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.

every shared layer—which is why our $\pi_{0.5}$ instantiation predicts per-layer KV deltas rather than a single feature vector (Section 3.5).

This decomposition creates a fundamental **asymmetric latency bottleneck**: the VLM backbone must execute at every control step, yet produces outputs that change slowly over time. The action head is not the bottleneck. Flow-matching heads (GR00T's DiT (Peebles & Xie, 2023; Bjorck et al., 2025), $\pi_{0.5}$'s flow expert (Physical Intelligence et al., 2025)) sample the time variable during training from a $\mathrm{Beta}(1.5, 1.0)$ distribution biased toward pure noise, so the model learns the full noise-to-data velocity in one step and single-step Euler integration is accurate at inference. Reducing denoising from 10 steps to one therefore incurs no statistically meaningful loss: $\pi_{0.5}$ Sync averages 96.96% at 1-step vs. 97.16% at 10-step on LIBERO, a 0.20 pp gap well below the $\sigma \approx 0.35$ pp of the 4-suite average (60 episodes per task; Table 1). The VLM backbone is therefore the clear bottleneck, and for real-time manipulation at 20–50Hz it severely limits deployment.

Key insight: Feature temporal redundancy. Consecutive VLM outputs exhibit high temporal similarity: each expensive backbone call produces features largely redundant with the previous timestep, so a much smaller model could *predict* the output delta $\Delta_t = z_{t+1} - z_t$ and let the action head operate on predicted features while the VLM backbone is called only every f steps.

Our approach. We propose **Latent Bridge**, a lightweight model that predicts one-step VLM output deltas:

$$\hat{z}_{t+1} = \hat{z}_t + \mathcal{B}(\hat{z}_t, s_t, q_t, a_{t-1}) \tag{1}$$

where $\hat{z}_t$ is the most recent representation (the fresh VLM feature on a VLM step, or the bridge's own previous prediction on a chained bridge step), s_t provides observation-dependent context (stable intermediate features or fresh vision embeddings), and q_t, a_{t-1} are the robot state and previous action. Between periodic VLM calls (every $f = 2$–4 steps), the bridge chains predictions autoregressively, replacing the VLM backbone at orders-of-magnitude lower latency.

A bridge trained on sync data alone—rollouts collected with the VLM running every step, so the bridge always sees clean VLM features as inputs—learns from a clean-input distribution; we call this supervised stage and the resulting model **R0**. At deployment, however, the bridge must consume its own predictions during autoregressive chaining, creating a *distribution shift* that sync-only training does not cover. We close this gap with a **task-agnostic DAgger round** (Ross et al., 2011): the R0 bridge is rolled out in the simulator while the VLM runs in parallel to provide ground-truth feature targets along the bridge's own induced trajectory, and the bridge is retrained on this mixed distribution; we call the refined model **R1**. This pipeline transfers across all four LIBERO task suites without modification.

Our full pipeline therefore consists of three stages applied task-adaptively: **Stage 1** is the bridge learning stage (sync collection, R0 supervised training, DAgger refinement to R1) and is the only stage required for simpler benchmarks; **Stage 2** (LoRA action-head adaptation) and **Stage 3** (phase-aware VLM scheduling) are added for long-horizon tasks where autoregressive bridge drift compounds over longer rollouts.

Our contributions are:

1. **Latent Bridge**: to our knowledge the first method to bring learned, cross-timestep feature/KV-delta drafting into closed-loop dual-system VLA control—a lightweight delta predictor that skips 50–75% of VLM backbone calls, with a pre-RoPE (Su et al., 2024) per-layer KV-delta predictor for interleaved architectures and a DAgger stage that stabilizes autoregressive chaining; orthogonal to and composable with existing VLM acceleration (pruning, quantization, layer skipping) and with action chunking.
2. We instantiate the bridge on two architecturally distinct VLAs—feature-space for GR00T and KV-cache for $\pi_{0.5}$—and evaluate on LIBERO (4 suites), RoboCasa (24 kitchen tasks), and ALOHA sim (transfer-cube), achieving **95–100% SR retention** with **1.55–1.73× net speedup**.
3. A complete three-stage pipeline, applied task-adaptively: Stage 1 alone recovers near-sync performance on shorter-horizon benchmarks, and all three stages together **recover near-synchronous VLA performance** (96% retention) on long-horizon tasks; the bridge itself trains in a few hours on a single GPU.

2 Related Work

Vision-Language-Action models. VLA models combine pretrained vision-language understanding with action prediction for robotic control. *Single-system* VLAs—RT-1 (Brohan et al., 2023a), RT-2 (Brohan et al., 2023b), OpenVLA (Kim et al., 2024), and RT-X (Padalkar et al., 2024)—tokenize actions and generate them autoregressively through the LLM, inheriting its sequential decoding bottleneck. *Dual-system* VLAs decouple perception from action generation: GR00T (Bjorck et al., 2025) uses a DiT action head, $\pi_{0.5}$ (Physical Intelligence et al., 2025) uses an interleaved flow-matching expert, Octo (Ghosh et al., 2024) and HPT (Wang et al., 2024) use diffusion decoders, and CogACT (Li et al., 2024a) uses a continuous action decoder. (Diffusion Policy (Chi et al., 2023), a behavior-cloning policy rather than a VLA, is cited for the diffusion action head these designs inherit.) We target dual-system VLAs, whose separable backbone/head interface lets us skip backbone computation on intermediate steps.

Reuse-then-forecast prediction of intermediate representations. Predicting a model's internal representations instead of recomputing them is an established primitive, and we position our contribution relative to it explicitly. EAGLE (Li et al., 2024c) drafts at the feature level within a single LLM decode and verifies drafts against the full model, making it lossless; KV Prediction (Horton et al., 2025) maps a small model's KV cache to a large model's in a one-shot, cross-model transfer for the same prompt; TaylorSeer (Liu et al., 2025a) extrapolates features training-free across diffusion denoising steps; and Deep Feature Flow (Zhu et al., 2017) propagates features across video frames by optical-flow warping for open-loop recognition. Our bridge differs from all four on the same axis: it predicts *learned deltas across environment control steps*, chained autoregressively inside a closed reactive loop, with a DAgger stage correcting the induced distribution shift—a regime none of these methods address.

Efficient VLM/VLA inference. General VLM acceleration spans KV-cache compression, speculative decoding, visual token pruning, and weight quantization (Li et al., 2024b; Xiao et al., 2024; Leviathan et al., 2023; Cai et al., 2024; Chen et al., 2024; Bolya et al., 2023; Lin et al., 2024; Frantar et al., 2023). VLA-specific adaptations span token pruning (VLA-Cache (Xu et al., 2025), SpecPrune-VLA (Wang et al., 2025), LightVLA (Jiang et al., 2026), VLA-Pruner (Liu et al., 2025b), Compressor-VLA (Gao et al., 2025)), layer skipping (MoLe-VLA (Zhang et al., 2026)), and quantization (SQAP-VLA (Fang et al., 2025), DyQ-VLA (Zheng et al., 2026)), reporting 1.08–1.62$\times$ *per-step* speedups with various SR trade-offs. None measure net episode wall-clock time, and all still execute the full backbone, which makes them complementary to our approach: we *skip the VLM forward pass entirely* on bridge steps, and the bridge composes with any of the above on the VLM steps that do execute.

World models and latent dynamics. World models learn environment dynamics for planning or data augmentation (Ha & Schmidhuber, 2018; Hafner et al., 2020; 2023; Micheli et al., 2023; Yang et al., 2024; Bruce et al., 2024), and JEPA (LeCun, 2022; Bardes et al., 2024) advocates prediction in representation space rather than pixel space. Our bridge also predicts future latent states, but its closest relatives are the reuse-then-forecast methods above: it operates in the VLM's hidden representation space to accelerate closed-loop inference, not to enable planning or representation learning.

3 Method

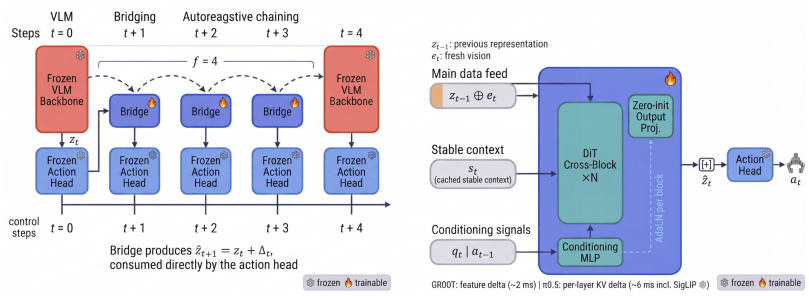
3.1 Problem Formulation

Consider a dual-system VLA with policy $\pi(a_t \mid o_t) = \mathcal{A}(\mathcal{V}(o_t))$, where $\mathcal{V}$ is the VLM backbone and $\mathcal{A}$ is the action head. At each control step t, the backbone produces features $z_t = \mathcal{V}(o_t) \in \mathbb{R}^{N \times D}$ (N: sequence length, D: feature dimension), and the action head decodes an action $a_t = \mathcal{A}(z_t, q_t)$ conditioned on proprioceptive state q_t.

We seek a lightweight bridge $\mathcal{B}$ that approximates VLM features without running $\mathcal{V}$. We define the *VLM call period* f: the backbone $\mathcal{V}$ executes once every f control steps (i.e., at $t = 0, f, 2f, \ldots$), and the bridge $\mathcal{B}$ fills the remaining $f-1$ intermediate steps. At intermediate offsets $t' \in \{1, \ldots, f-1\}$ from the last VLM call at step t, the bridge predicts:

$$\hat{z}_{t+t'} = \hat{z}_{t+t'-1} + \mathcal{B}\big(\hat{z}_{t+t'-1},\ s_t,\ q_{t+t'},\ a_{t+t'-1}\big), \qquad t' = 1, \ldots, f-1 \tag{2}$$

$$a_{t+t'} = \mathcal{A}\big(\hat{z}_{t+t'},\ q_{t+t'}\big) \qquad \text{(bridge step action)}$$



(a) Inference Timeline (b) Bridge Internal Structure

Figure 2: Architecture comparison. Both variants use a DiT backbone with AdaLN conditioning. GR00T operates on a single feature vector; $\pi_{0.5}$ operates on per-layer KV pairs. Zero-initialized output ensures the bridge starts at the copy baseline.

with the boundary convention $\hat{z}_t := z_t$ (the fresh VLM feature on VLM steps), and where $s_t \in \mathbb{R}^{N \times D}$ denotes stable intermediate-layer VLM features cached from the last VLM call, providing visual context without rerunning the backbone (Section 3.2). The bridge therefore chains *autoregressively*: at $t' = 1$ the input is the fresh VLM feature z_t; at $t' \geq 2$ the input is the bridge's own previous prediction $\hat{z}_{t+t'-1}$. This autoregressive chaining motivates the DAgger training stage (Section 3.3).

The simplest baseline is **feature caching**, which sets $\mathcal{B} = \mathrm{Id}$ (i.e., $\hat{z}_{t+1} = z_t$, equivalently $\Delta_t = 0$), reusing stale VLM features on non-VLM steps: the zero-function bridge, a lower bound any learned bridge must beat.

The period f trades speedup against prediction error accumulated over consecutive bridge steps; we seek the largest f that maintains $>95\%$ task performance retention.

3.2 Latent Bridge Architecture

Feature-space bridge (GR00T). For VLAs with a single feature vector interface between backbone and action head, the bridge predicts one-step feature deltas via residual prediction:

$$\hat{z}_{t+1} = \hat{z}_t + \Delta_t, \qquad \Delta_t = \mathcal{B}(\hat{z}_t,\ s_t,\ q_t,\ a_{t-1}) \tag{3}$$

where $\hat{z}_t \in \mathbb{R}^{N_{\text{img}} \times D}$ is the most recent feature consumed by the action head (fresh from the VLM at a VLM step, or autoregressively chained at a bridge step; boundary $\hat{z}_t := z_t$ as in Eq. 2), s_t are stable-layer features providing scene context, q_t is the proprioceptive state, and a_{t-1} is the previous action. The bridge $\mathcal{B}$ is a stack of DiT blocks (Peebles & Xie, 2023) with **self-attention** over the input features z_t (learned positional embeddings), **cross-attention** to the stable context s_t (visual grounding without a vision encoder), **AdaLN conditioning** on (q_t, a_{t-1}) at every block, and a **zero-initialized output projection**, so the untrained bridge starts at the feature-caching baseline ($\Delta_t = 0$).

Temporal similarity of prefix representations (measured). To ground the design, we measure consecutive-step cosine similarity of the per-layer prefix hidden states (768 image tokens plus prompt tokens) of the pretrained $\pi_{0.5}$ backbone over closed-loop observation trajectories: cosine per token, averaged over tokens, then step-pairs, then episodes; layers indexed 0 (embedding) to 18 (final); 15 real (DROID) and 15 sim (LIBERO) episodes, 705 consecutive-step pairs each (per-layer values in Table 9). Image tokens show a stable-to-dynamic structure: early and middle layers are nearly invariant ($\approx$0.999) while the final layer is dynamic (0.983 real / 0.963 sim). This motivates the decomposition the bridge uses: it predicts the action-head input z_t and consumes the stable-layer features s_t as cross-attention context; on bridge steps, s_t is reused from the last VLM call while z_t is freshly predicted.

Image-only processing. Text tokens are far more stable than image tokens at every layer: averaged over all 19 layers, their consecutive-step cosine is 0.9994 (real) / 0.9993 (sim), against 0.9969 / 0.9933 for image tokens. They are not perfectly invariant, however: through bidirectional prefix attention to visual tokens, the deepest layer drifts to 0.9966 (real) / 0.9954 (sim), which is still five to eight times less drift than the image tokens' 0.983 / 0.963 at the same depth, so predicting near-zero text deltas would only dilute the gradient on image-token deltas. The bridge therefore operates exclusively on image tokens ($N_{\text{img}} \ll N$) and copies text tokens from the last VLM cache; dropping text from the bridge *input* is safe because $\hat{z}_t$ was already produced by a text-conditioned

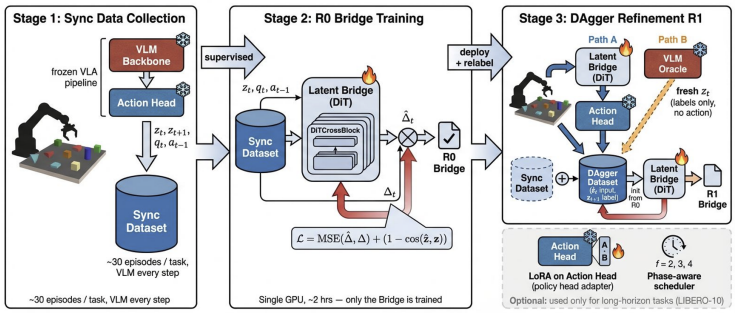


Figure 3: Task-agnostic three-stage pipeline. The same pipeline transfers across all LIBERO suites, RoboCasa, and ALOHA sim without modification. DAgger closes the distribution gap between sync training and bridge deployment.

VLM and implicitly encodes the instruction. Empirically this yields no SR loss while making the bridge sequence-length-agnostic.

3.3 Training Pipeline

Sync data collection. We roll out the pretrained VLA in sync mode (VLM every step) in the simulator and record tuples $(z_t, z_{t+1}, s_t, q_t, a_{t-1})$ along its closed-loop observation trajectory; bridge training is then fully offline on this fixed dataset.

R0 bridge training. We denote the initial bridge trained on sync-only data as the "round-0" bridge (**R0**). The bridge is trained on sync pairs with a combined MSE and cosine loss, restricted to image tokens:

$$\mathcal{L} = \|\hat{z}_{t+1} - z_{t+1}\|^2 + \alpha \left(1 - \frac{\hat{z}_{t+1} \cdot z_{t+1}}{\|\hat{z}_{t+1}\| \, \|z_{t+1}\|}\right) \tag{4}$$

where all terms operate over image tokens only ($\alpha = 1.0$); text-token deltas are masked out (Section 3.2).

DAgger refinement (R1). After R0 training, we apply one round of DAgger refinement to produce the "round-1" bridge (**R1**). The R0 bridge is deployed in simulation with VLM call period f, while a VLM oracle runs in parallel to provide ground-truth features at every control step (the robot still follows the bridge policy's actions). This yields DAgger pairs $(\hat{z}_{t+t'-1},\ z_{t+t'})$ for each bridge step at offset $t' \in \{1, \ldots, f-1\}$ from a VLM call at step t: the input is exactly what Eq. 2 feeds the bridge—which reduces to the fresh VLM feature z_t at $t'{=}1$ via the boundary convention $\hat{z}_t := z_t$, and is the bridge's own (potentially noisy) prediction at $t' \geq 2$—paired with the oracle's ground-truth feature $z_{t+t'}$ as target. The bridge is retrained on mixed sync + DAgger data, resuming from R0 weights with a reduced learning rate. DAgger is essential because at $t' \geq 2$ the bridge's input is its own prediction, not a clean VLM feature; without DAgger, errors compound rapidly at higher f.

3.4 Optional Enhancements

Phase-aware VLM scheduling. For complex tasks with diverse motion phases, we dynamically adjust f based on the previous action's translation magnitude $\|a_t^{\text{trans}}\|$: high motion ($\|a_t^{\text{trans}}\| > \tau_{\text{nav}}$) $\rightarrow f{=}2$, medium ($\tau_{\text{manip}} < \|a_t^{\text{trans}}\| \leq \tau_{\text{nav}}$) $\rightarrow f{=}3$, low ($\|a_t^{\text{trans}}\| \leq \tau_{\text{manip}}$) $\rightarrow f{=}4$. This allocates more VLM compute during high-dynamics phases, where the features carried between VLM calls change fastest and bridge predictions are least reliable; on the long-horizon LIBERO-10 suite it is the stage that closes most of the remaining gap to sync (Section 4.4).

LoRA action head adaptation. When bridge features are significantly noisier than clean VLM features, we apply Low-Rank Adaptation (LoRA) (Hu et al., 2022) to the action head's DiT, training on a 50/50 mix of bridge and sync features with Gaussian noise augmentation. This increases the action head's tolerance to bridge prediction noise.

3.5 Adaptation to $\pi_{0.5}$: KV-Cache Bridge

Unlike GR00T's single feature-vector interface, $\pi_{0.5}$ (Physical Intelligence et al., 2025) uses an interleaved design where the Gemma-2B backbone and Gemma-300M action expert share transformer

layers: at each of L=18 layers, action (suffix) tokens cross-attend to the backbone's (prefix) per-layer KV cache. We adapt the bridge to predict pre-RoPE key and value deltas for all L layers simultaneously:

$$\{\Delta K_l, \Delta V_l\}_{l=1}^{L} = \mathcal{B}_{\text{KV}}(\Delta e_t,\ e_t,\ \widehat{\text{KV}}_{t-1},\ q_t,\ a_{t-1}), \tag{5}$$

where $e_t = \text{SigLIP}(o_t)$ is the vision embedding, $\Delta e_t = e_t - e_{t-1}$ encodes what changed visually between steps, and $\widehat{\text{KV}}_{t-1}$ is the most recent per-layer KV cache (fresh from the VLM at a VLM step, or autoregressively chained from the bridge's previous prediction at a bridge step; same boundary convention as Eq. 2). $\mathcal{B}_{\text{KV}}$ uses a shared DiT backbone with 18 lightweight per-layer output heads. SigLIP (∼5ms) provides fresh visual context far more cheaply than the full Gemma prefix (∼46ms); total bridge-step cost is ∼6ms with bf16 + `torch.compile`. Architectural details and RoPE handling are in Appendix A.

3.6 Inference

At deployment with VLM call period f, **VLM steps** ($t \bmod f = 0$) run the full backbone $\mathcal{V}$ and cache z_t, s_t, and KV pairs for bridge use; **bridge steps** ($t \bmod f \neq 0$) have $\mathcal{B}$ predict Δ from the most recent representation—the fresh VLM feature z_t on the first bridge step, or its own previous output $\hat{z}_{t'-1}$ on subsequent steps (autoregressive chaining)—and the action head runs on the predicted $\hat{z}_t$, at a cost of 2–6ms vs. 46–63ms for the replaced VLM backbone.

4 Experiments

4.1 Setup

Base models. Two dual-system VLAs: **GR00T-N1.6-3B** (Bjorck et al., 2025) (Eagle backbone—SigLIP2+Qwen3—with a DiT action head, fine-tuned per LIBERO suite) and $\pi_{0.5}$ (Physical Intelligence et al., 2025) (PaliGemma backbone—SigLIP+Gemma-2B—with a Gemma-300M action expert, public LIBERO checkpoint).

Benchmarks. Four LIBERO (Liu et al., 2023) suites (Spatial, Object, Goal, and the long-horizon LIBERO-10), plus **RoboCasa** (Nasiriany et al., 2024) (24 kitchen tasks, GR00T zero-shot) and **ALOHA sim** (Zhao et al., 2023) (bimanual transfer-cube, $\pi_{0.5}$) to test cross-platform generalization.

Metrics. **Success rate (SR)** per task is the fraction of episodes completing within the suite's step cap; per-suite SR averages over its tasks. Each task uses 3 seeds × 20 episodes on LIBERO (600 per suite) and RoboCasa (1440 total); ALOHA uses 3 seeds × 50 episodes; run-to-run std-dev is <0.5 pp throughout, and Wilson 95% CIs are given with Table 1. **Avg latency** is ms / inference call, averaged across the VLM/bridge mix within one period f. **Episode length** is over successful episodes only (failures hit the step cap). **Net speedup** is the episode-length-corrected wall-clock ratio $(t_{\text{sync}} \cdot L_{\text{sync}})/(t_{\text{method}} \cdot L_{\text{method}})$.

Hardware and bridge. All latency on a single A100 80GB, bfloat16+`torch.compile`; per-component costs are in the headers of Table 1. A 19M small variant matches the 148M default (§4.4); bridge training takes 2–4 hours per suite on one GPU, 8–12 GPU-hours for the full pipeline including data collection (breakdown and amortization in Appendix A).

Baselines. We compare against FastV (Chen et al., 2024) and VLA-Cache (Xu et al., 2025) (token pruning), plus feature caching (Δ_t=0). Most VLA pruning methods target single-system VLAs (OpenVLA), where the LLM emits action tokens directly, and dual-system action heads consume dense VLM conditioning, which makes pruning adaptation harder; SpecPrune-VLA (Wang et al., 2025) reports $\pi_{0.5}$ numbers but is not open-sourced, so we adapt FastV and VLA-Cache, the open-sourced representatives, to both VLAs.

4.2 Main Results

We evaluate Latent Bridge against four baselines—FastV, VLA-Cache, feature caching (stale reuse), and training-free Taylor extrapolation (Liu et al., 2025a) ported to control steps—on the four LIBERO suites (Table 1). Taylor extrapolation falls below even stale reuse on all four suites (46.67 vs. 56.38 mean): the per-step KV dynamics are not smooth enough for a low-order polynomial, so closed-form forecasting cannot substitute for the learned delta. At-or-above-sync cells (marked ≈) are within

Table 1: **Comparison of VLA inference methods on LIBERO.** Success rate (%, mean across 3 random seeds $\times$ 20 episodes per task) across four suites, average per-step latency (averaged over VLM and non-VLM steps within one call period f), average episode length on successful episodes, and net wall-clock speedup. Feature Caching ($\Delta_t = 0$) reuses stale VLM outputs at matched f without learned prediction. Baseline adaptation details are in Appendix H.

Method	Spatial	Object	Goal	Long	Avg SR↑	Avg Latency (ms)↓	Ep. Len↓	Speedup↑
GR00T-N1.6-3B VLM Backbone: 63ms Action Head (1-step): 27ms Bridge: 2ms f=2–4 (phase-aware)								
Sync (every step)	96.17	99.83	97.33	93.00	96.58	90	19	1.00×
+ FastV	86.17	84.00	85.00	58.83	78.50	77 (−15%)	23 (+23%)	0.97×
+ VLA-Cache	91.17	90.00	89.00	85.17	88.84	75 (−17%)	21 (+8%)	1.09×
+ Feature Caching	35.67	3.00	56.00	42.33	34.25	48 (−47%)	—†	—†
+ Latent Bridge (ours)	**95.83**	**97.83**	**95.33**	**89.17**	**94.54**	**49** (−45%)	20 (+6%)	**1.73**×
$\pi_{0.5}$ VLM Backbone: 46ms Action Head (1-step): 30ms Bridge: 6ms (SigLIP 5ms + DiT 1ms) f=4								
Sync (every step)	98.83	98.17	97.00	93.83	96.96	76	31	1.00×
+ FastV	88.17	86.00	86.83	62.17	80.79	67 (−12%)	37 (+20%)	0.95×
+ VLA-Cache	93.17	91.00	92.17	86.83	90.79	65 (−14%)	33 (+6%)	1.10×
+ Feature Caching	46.67	57.33	68.83	52.67	56.38	45 (−41%)	—†	—†
+ Taylor extrapolation	35.67	45.67	59.33	46.00	46.67	45 (−41%)	—†	—†
+ Latent Bridge (ours)	**99.00**≈	**97.67**	**97.33**≈	**93.67**	**96.92**	**46** (−39%)	33 (+5%)	**1.55**×

† Feature Caching: low SR, most episodes fail to max steps.

Ep. Len = avg control steps per successful episode (GR00T: 1 step = 8 env steps; $\pi_{0.5}$: 1 step = 5 env steps).

≈ At or above sync but within its Wilson 95% CI (n=600: $\pm$1.4pp per suite cell, $\pm$0.7pp on the average), i.e. not statistically distinguishable from sync.

evaluation noise (RoboCasa Coffee p=0.71; f=3 Object 100.0, p=0.06, and Goal 99.0, p=0.15, Appendix D), whereas the operative comparison, bridge vs. stale reuse at matched f, is far outside it (z=7.2–21.8 across benchmarks, all $p<10^{-4}$). Latent Bridge retains **95–100%** of sync SR (avg 94.5% on GR00T, 96.9% on $\pi_{0.5}$) while feature caching collapses to 3–56%; that gap is the evidence that the bridge predicts the per-step VLM update rather than reusing a stale one. The token-pruning baselines preserve more SR than naive caching but still execute the full backbone, so their net wall-clock gains are limited to 0.95–0.97× for FastV and 1.09–1.10× for VLA-Cache. Both report ∼1.3–1.5× on single-system VLAs, but on dual-system VLAs the speedup applies only to the backbone fraction of per-step cost (60–70%), an Amdahl ceiling on the net gain (Appendix H). Latent Bridge instead *skips the backbone entirely* on bridge steps—replacing a 46–63 ms VLM call with a 2–6 ms bridge—and reaches **1.55–1.73× net speedup** despite a small (+5–6%) increase in episode length.

4.3 Cross-Benchmark Evaluation

We evaluate on two additional benchmarks beyond LIBERO to test cross-platform generalization: (1) **RoboCasa** (Nasiriany et al., 2024) with GR00T-N1.6-3B (pretrained checkpoint without task-specific finetuning), covering 24 kitchen tasks (door/drawer articulation, appliance control, coffee, pick-and-place) on the Panda robot; and (2) **ALOHA sim transfer-cube** (Zhao et al., 2023) with $\pi_{0.5}$ fine-tuned from the pretrained base, testing the KV bridge on a bimanual 14-DoF setup. Both use the LIBERO training pipeline unchanged.

Table 2: Cross-benchmark results at f=3 (all values are success rate in %, mean across 3 seeds). RoboCasa: 24 kitchen tasks, GR00T, zero-shot. ALOHA: transfer-cube, $\pi_{0.5}$, 50 episodes per seed, 1-step denoise. ≈: above sync but within its 95% CI (p=0.71).

Benchmark	Tasks	Sync	Cache	Bridge	Retention	Cache Retention
RoboCasa (GR00T)						
Door/Drawer	6	78.33%	66.67%	**75.83%**	96.81%	85.12%
Appliance	7	80.67%	65.00%	**77.17%**	95.66%	80.58%
Coffee	3	53.33%	41.67%	**56.67%**≈	106.26%	78.13%
Pick-and-Place	8	49.33%	26.83%	**43.83%**	88.85%	54.39%
Average (24)	24	**66.22%**	49.78%	**63.16%**	**95.38%**	75.17%
ALOHA sim ($\pi_{0.5}$)	1	**88.00%**	46.00%	**86.00%**	**97.73%**	52.27%

Bridge achieves **63.16% SR on RoboCasa** (95.38% retention) and **86.00% on ALOHA** (97.73% retention) versus 49.78% and 46.00% for caching, +13.38pp and +40.00pp gaps that are largest on precise-manipulation tasks (Table 2); the same pipeline therefore transfers across embodiments (Panda, ALOHA bimanual), action spaces (7- vs. 14-DoF), and task distributions.

4.4 Ablation Studies

All ablations use the $\pi_{0.5}$ KV bridge on LIBERO suites with 1-step denoise unless otherwise noted.

Vision input. We ablate the fresh vision embedding Δe_t by zeroing it during training, forcing the bridge to predict purely from previous KV cache, state, and action.

Variant	Sp	Ob	Go	L-10
Full (148M)	99.00	97.67	97.33	93.67
w/o vision	98.17	95.50	86.17	65.33
w/o stable	98.00	97.83	92.17	82.67
Small (19M)	99.17	98.83	97.50	93.50

Table 3: Bridge ablations on $\pi_{0.5}$ (all values are success rate in %, mean across 3 seeds). Vision and stable context both matter with task complexity; 19M matches 148M.

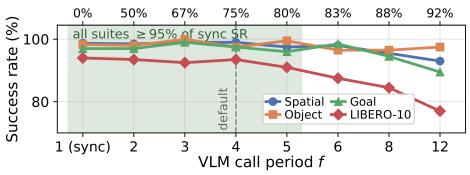


Figure 4: VLM call period vs. performance on $\pi_{0.5}$. Spatial/Object/Goal stay at or above 94.5% SR through f=8; LIBERO-10 degrades earlier due to long-horizon error compounding.

As Table 3 shows, both vision and stable context matter, with degradation scaling with task complexity: removing vision costs −11.16pp on Goal and −28.34pp on LIBERO-10, and removing stable context −5.16pp and −11.00pp, as the bridge loses its anchor representation of slowly-changing scene structure. The 19M bridge matches the 148M bridge within 1.2pp on every suite, so capacity is not the bottleneck.

Table 4: DAgger progression (success rate in %, mean across 3 seeds): R1 (+DAgger) adds a consistent 3–13pp over R0 (sync-only) across all suites and both architectures.

	$\pi_{0.5}$ KV Bridge				GR00T Feature Bridge			
Stage	Spatial	Object	Goal	L-10	Spatial	Object	Goal	L-10
Feature caching	46.67	57.33	68.83	52.67	35.67	3.00	56.00	42.33
R0 (sync-only)	95.17	93.67	92.83	89.17	91.83	93.17	91.17	54.33
R1 (+DAgger)	99.00	97.67	97.33	93.67	95.83	97.83	95.33	67.17

DAgger training progression. Table 4 traces SR through the pipeline: R0 alone recovers ∼93–96% of sync SR on the three easier LIBERO suites, and DAgger adds +4pp on $\pi_{0.5}$ and +4–13pp on GR00T (largest on long-horizon tasks, where autoregressive drift accumulates), pushing retention to 97–100%. For GR00T LIBERO-10, Stages 2–3 (§3.4) further lift R1 67.17% → 73.17% (Stage 2 LoRA) → 89.17% (Stage 3 phase-aware), a +34.84pp total gain; on shorter-horizon suites these stages are within seed noise, so Stage 1 alone suffices.

VLM call period vs. performance. SR degrades *gracefully* as f grows from 1 (sync) to 12 (Figure 4): Spatial, Object, and Goal stay at or above 94.5% through f=8, where LIBERO-10 still retains 84.5%; the default f=4 gives 75% VLM savings with <1pp average SR drop, and f=5–6 remains a further speedup knob on shorter-horizon tasks.

4.5 Case Study: Why Caching Fails

On a representative LIBERO-Spatial episode at f=3, caching's KV cosine to ground truth drops to ∼0.91 in a sawtooth pattern as the robot moves, while the bridge stays above 0.98 (Fig. 5a); the resulting off-distribution features push the action head off-trajectory and the task fails (Fig. 5b). A cached KV encodes only the stale scene, whereas the bridge actively predicts the update from cheap signals—SigLIP delta (∼5ms), state, and previous action—closing the gap with minimal cost.

4.6 Fidelity Governs Success: An Action-Level Chain

Monotone fidelity–action–SR chain. Table 5 pairs, for each reconstruction method, its per-step recovered-KV cosine, its per-step action deviation from the true VLM-conditioned actions (identical observations and flow-matching noise, so only the KV fill differs; f=4), and its closed-loop SR. All three columns rank identically on every suite: sync ≈ bridge ≫ stale reuse > Taylor. The open-loop fidelity the bridge is trained to maximize is therefore predictive of closed-loop success, which licenses reading open-loop measurements as evidence: across the operating points of Figure 4 (sync, bridge

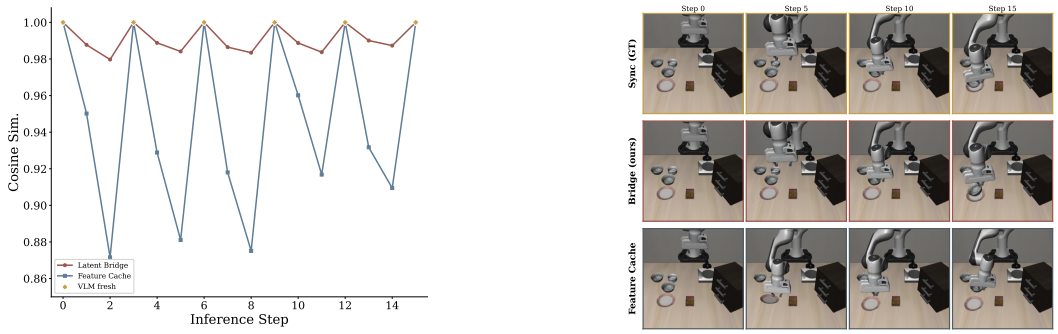


Figure 5: Case study, LIBERO-Spatial episode at f=3. **(a)** KV cosine to ground truth: caching (red) drops to ∼0.91 in a sawtooth; bridge (blue) stays above 0.98. **(b)** Keyframes: caching drifts off-trajectory and fails; bridge tracks the sync trajectory and succeeds.

Table 5: Per-step recovered-KV cosine, per-step action RMSE vs. the sync-conditioned reference, and closed-loop SR ($\pi_{0.5}$, LIBERO, 3 seeds, f=4). All three quantities rank identically. The same measurement on real robot streams is in Table 17.

Method	KV cos	action RMSE	Spatial	Object	Goal	Long	Mean SR
Sync	1.00	0 (ref.)	98.8	98.2	97.0	93.8	96.96
Bridge	**0.976**	**0.136**	99.0	97.7	97.3	93.7	**96.92**
Stale reuse (caching)	0.951	0.279	46.7	57.3	68.8	52.7	56.38
Taylor	0.910	0.298	35.7	45.7	59.3	46.0	46.67

at f=2–8, caching) recovered-KV cosine and SR correlate at Spearman ρ=0.82, and on real robot camera streams the same replay protocol yields the same ranking, with the bridge's action deviation roughly half that of stale reuse and within 2.6× of the policy's own sampling-noise floor, against ≈5× for the closed-form baselines (Appendix K).

Composability with action chunking. Chunking amortizes the whole stack by acting open-loop; the bridge amortizes only the backbone and keeps the action head reactive, so the two compose. Re-training the $\pi_{0.5}$ bridge for a length-10 chunk (3 seeds): sync is unchanged (96.96 at chunk 5 vs. 97.50 at chunk 10, measured directly, n=240), stale reuse collapses (56.38→38.92) because a longer open-loop chunk widens the stale-vs-fresh gap, and the bridge holds near sync (96.92→95.83), widening its margin over stale reuse from 40 to 57pp (Appendix F).

5 Conclusion

Latent Bridge predicts VLM output deltas for efficient dual-system VLA inference. Instantiated on GR00T (feature-space) and $\pi_{0.5}$ (KV-cache), it achieves 95–100% SR retention with 50–75% VLM savings and 1.55–1.73× net speedup across LIBERO, RoboCasa, and ALOHA sim, via a three-stage task-adaptive pipeline and no changes to the pretrained VLA.

Limitations. **Simulation only.** Our contribution is an inference-stage algorithm, not a claim about physical manipulation, which enters only through the fixed policy. Because the bridge never touches the environment, a real robot changes only the observation stream and executor; Appendix K tests the transfer-relevant links open-loop on real robot streams, but whether per-step deviations compound differently under physical dynamics is unmeasured. **A100-only latency.** The saving depends only on $t_{\text{bridge}}/t_{\text{VLM}}$ and the backbone fraction (both less favorable to the VLM on weaker accelerators); we have no edge-device numbers. **Safety.** Acting open-loop on predicted features between syncs has error bounded by f, the zero-initialized caching fallback, and phase-aware scheduling (Table 5; safeguards in Appendix J). **Other.** DAgger needs a simulator at training time; the bridge is checkpoint-specific (Appendix A); $\pi_{0.5}$'s action expert leaves a ∼30ms floor, addressable by suffix-level bridging (Song et al., 2023).

REPRODUCIBILITY STATEMENT

An anonymized archive of the code accompanies this submission as supplementary material; the public repository and the trained bridge checkpoints will be released on acceptance. The release contains every stage behind the reported numbers: sync data collection, R0 training, DAgger data collection and R1 refinement for both the GR00T feature bridge and the $\pi_{0.5}$ KV bridge (hyperparameters in Appendix A); the FastV, VLA-Cache, feature-caching and Taylor-extrapolation baselines with their adaptation choices (Appendix H); the closed-loop evaluation clients for LIBERO, RoboCasa and ALOHA sim with the seeds and episode counts of Section 4, which use paired initial states across fill modes; the latency measurement scripts (Appendix E); and the scripts for the open-loop real-stream evaluation of Appendix K (KV extraction, chained recovery, seeded-noise action-fidelity replay, motion binning). Base models (GR00T-N1.6, $\pi_{0.5}$), benchmarks and the DROID episodes used for the real-stream evaluation are public.

THE USE OF LARGE LANGUAGE MODELS

An LLM-based coding assistant was used, under the authors' direction and with the authors responsible for every claim, to help draft and polish portions of the manuscript text, to implement evaluation and analysis scripts (the baseline ports, the action-fidelity replay and the real-stream evaluation) under the authors' specifications, and to search for and summarize related work. All generated text and code were reviewed, tested and edited by the authors; every citation was verified against the cited source; every number reported here was produced by the released scripts. No generative model was used to produce data, figures of results or scientific claims.

REFERENCES

Adrien Bardes et al. Revisiting feature prediction for learning visual representations from video. *arXiv:2404.08471*, 2024.

Johan Bjorck et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv:2503.14734*, 2025.

Kevin Black et al. π_0: A vision-language-action flow model for general robot control. *arXiv:2410.24164*, 2024.

Daniel Bolya et al. Token merging: Your ViT but faster. In *ICLR*, 2023.

Anthony Brohan et al. RT-1: Robotics transformer for real-world control at scale. In *RSS*, 2023a.

Anthony Brohan et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *CoRL*, 2023b.

Jake Bruce et al. Genie: Generative interactive environments. In *ICML*, 2024.

Tianle Cai et al. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *ICML*, 2024.

Liang Chen et al. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *ECCV*, 2024.

Cheng Chi et al. Diffusion policy: Visuomotor policy learning via action diffusion. In *RSS*, 2023.

Hengyu Fang et al. SQAP-VLA: Synergistic quantization-aware pruning for VLAs. *arXiv:2509.09090*, 2025.

Elias Frantar et al. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *ICLR*, 2023.

Juntao Gao et al. Compressor-VLA: Instruction-guided visual token compression for efficient robotic manipulation. *arXiv:2511.18950*, 2025.

Dibya Ghosh et al. Octo: An open-source generalist robot policy. In *RSS*, 2024.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *NeurIPS*, 2018.

Danijar Hafner et al. Dream to control: Learning behaviors by latent imagination. In *ICLR*, 2020.

Danijar Hafner et al. Mastering diverse domains through world models. In *ICLR*, 2023.

Maxwell Horton, Qingqing Cao, Chenfan Sun, Yanzi Jin, Sachin Mehta, Mohammad Rastegari, and Moin Nabi. KV prediction for improved time to first token. In *ICLR Workshop on Scalable Optimization for Efficient and Adaptive Foundation Models (SCOPE)*, 2025.

Edward J Hu et al. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Titong Jiang et al. The better you learn, the smarter you prune: Towards efficient vision-language-action models via differentiable token pruning. In *ICRA*, 2026.

Moo Jin Kim et al. OpenVLA: An open-source vision-language-action model. In *CoRL*, 2024.

Yann LeCun. A path towards autonomous machine intelligence. *OpenReview*, 2022.

Yaniv Leviathan et al. Fast inference from transformers via speculative decoding. In *ICML*, 2023.

Qixiu Li et al. CogACT: A foundational vision-language-action model for synergizing cognition and action. *arXiv:2411.19650*, 2024a.

Yuhong Li et al. SnapKV: LLM knows what you are looking for before generation. In *NeurIPS*, 2024b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 28935–28948, 2024c.

Ji Lin et al. AWQ: Activation-aware weight quantization for LLM compression and acceleration. In *MLSys*, 2024.

Bo Liu et al. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *NeurIPS*, 2023.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with TaylorSeers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15853–15863, 2025a.

Ziyan Liu et al. VLA-Pruner: Temporal-aware dual-level visual token pruning for efficient VLA inference. *arXiv:2511.16449*, 2025b.

Vincent Micheli et al. Transformers are sample-efficient world learners. In *ICLR*, 2023.

Soroush Nasiriany et al. RoboCasa: Large-scale simulation of everyday tasks for generalist robots. In *RSS*, 2024.

Abhishek Padalkar et al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *ICRA*, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4172–4182, 2023.

Physical Intelligence, Kevin Black, Noah Brown, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv:2504.16054*, 2025.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

Yang Song et al. Consistency models. In *ICML*, 2023.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Hanzhen Wang et al. SpecPrune-VLA: Accelerating VLAs via action-aware self-speculative pruning. *arXiv:2509.05614*, 2025.

Lirui Wang et al. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. *arXiv:2409.20537*, 2024.

Guangxuan Xiao et al. Efficient streaming language models with attention sinks. In *ICLR*, 2024.

Siyu Xu et al. VLA-Cache: Efficient vision-language-action manipulation via adaptive token caching. *arXiv:2502.02175*, 2025.

Sherry Yang et al. Learning interactive real-world simulators. In *ICLR*, 2024.

Rongyu Zhang et al. MoLe-VLA: Dynamic layer-skipping VLA via mixture-of-layers. In *AAAI*, 2026.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems (RSS)*, 2023.

Zihao Zheng et al. DyQ-VLA: Temporal-dynamic-aware quantization for embodied VLAs. *arXiv:2603.07904*, 2026.

Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep feature flow for video recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4141–4150, 2017.

A Implementation Details

Bridge architecture hyperparameters. Both VLA variants use the same DiT backbone with AdaLN conditioning. Table 6 lists the key hyperparameters.

Table 6: Bridge model hyperparameters. The small variant achieves comparable SR; we use the full variant in all main experiments.

	GR00T feature bridge		$\pi_{0.5}$ KV bridge	
	Full	Small	Full	Small
Hidden dim	768	384	768	384
Num blocks	12	4	10	4
Num heads	12	6	12	6
Input feature dim	2048	2048	2048	2048
Output dim per layer	2048	2048	512 (K256+V256)	512
Num output heads	1	1	18 (per Gemma layer)	18
Seq length (image-only)	162	162	768	768
State dim	8	8	8 / 14	8 / 14
Action dim	7	7	7 / 14	7 / 14
Total params	186.2M	18.6M	148.5M	18.9M

Training hyperparameters. All bridges are trained with AdamW (weight decay 10^{-4}), cosine LR schedule, and gradient clipping at norm 1.0. Key hyperparameters:

- **R0 (sync-only)**: learning rate 3×10^{-4}, 200 epochs (GR00T) / 50 epochs ($\pi_{0.5}$), batch size 64 (GR00T) / 4 ($\pi_{0.5}$).
- **R1 (DAgger refinement)**: resumed from R0 weights with `-reset_best` (best val cos reset to 0). Learning rate 3×10^{-4} for GR00T Goal (T_max=16), 3×10^{-5} for LIBERO-10 DAgger (LR 10× lower to stabilize long-horizon distribution shift).
- **Loss**: $\mathcal{L} = \|\hat{z}_{t+1} - z_{t+1}\|^2 + (1 - \cos(\hat{z}_{t+1}, z_{t+1}))$, restricted to image tokens.

Data collection. Sync data is collected by deploying the VLA in sync mode (VLM every step):

- **GR00T**: 300 sync episodes per LIBERO suite (≈4,500–6,600 inference samples after filtering).
- $\pi_{0.5}$: 300 sync episodes per suite, 50 episodes for ALOHA.
- **DAgger data (R1)**: 300 episodes of bridge rollout with VLM oracle labels, collected at f=3.

Hardware and runtime. All training and inference use a single NVIDIA A100 80GB with bfloat16 and `torch.compile(mode="max-autotune")`. Bridge training completes in 2–4 hours per suite. The end-to-end pipeline takes roughly 8–12 GPU-hours per suite, decomposed approximately as sync data collection 4–6 h, R0 supervised training 2–4 h, DAgger data collection 1–2 h, and R1 fine-tuning ≈0.5 h (10–16 epochs resumed from R0).

Training-cost amortization. On $\pi_{0.5}$ LIBERO the bridge saves about 0.84 s of GPU inference time per episode (31 inference steps at 76 ms sync vs. 33 steps at 46 ms). Charging the full 8–12 h pipeline, the training cost is recovered after roughly 34k–51k episodes; charging only the marginal compute (3.5–6.5 h: sync rollouts are ordinary policy rollouts that reuse existing development and evaluation logs, and oracle labeling is an offline pass over those frames), break-even is roughly 15k–28k episodes, which a deployed fleet running one checkpoint reaches within days. The aggregate-compute framing understates the practical benefit: the bridge's primary value is the per-step latency reduction (76 to 46 ms on average), which is a real-time control capability rather than an energy saving.

$\pi_{0.5}$ KV bridge: architecture and RoPE handling. Given the flattened per-layer KV cache $\text{KV}_{t-1} \in \mathbb{R}^{S \times (18 \cdot 512)}$ (256 pre-RoPE K + 256 V per layer, flattened along the layer dimension), embedding delta Δe_t, and current embedding e_t, the bridge computes:

$$\hat{K}_{l,t} = K_{l,t-1} + \Delta K_l, \qquad \hat{V}_{l,t} = V_{l,t-1} + \Delta V_l.$$

Δe_t is projected and added to the KV input as a residual signal of "what changed visually"; e_t serves as cross-attention context. RoPE is applied to $\hat{K}_{l,t}$ after prediction: since RoPE is a deterministic position-dependent rotation, we extract pre-RoPE K from the model's post-RoPE cache via the inverse rotation $k_{\text{pre}} = k_{\text{post}} \cos - \text{rotate_half}(k_{\text{post}}) \sin$, predict pre-RoPE deltas, then re-apply RoPE before reinserting into the cache. The output head for each layer is a lightweight `LayerNorm +`

`Linear` mapping the shared DiT hidden state (d=768) to a 512-dim vector (K delta concatenated with V delta); all output heads are zero-initialized so the untrained bridge starts at the copy baseline Δ=0. This 18-head design amortizes the DiT backbone cost across all layers while letting each head specialize in its layer's dynamics.

B PER-TASK RESULTS

LIBERO per-task breakdown. Table 7 reports per-task success rate for the $\pi_{0.5}$ KV bridge at f=4 (default configuration from main table). Each task uses 20 evaluation episodes with fixed initial states.

Table 7: Per-task success counts (out of 20 episodes) for the $\pi_{0.5}$ KV bridge at f=4 across the four LIBERO suites. Suite averages (bottom row) match the headline LIBERO results in the main paper.

LIBERO-Spatial	Bridge	LIBERO-Object	Bridge
pick up black bowl between the plate	20/20	pick up alphabet soup	19/20
pick up black bowl next to the ramekin	20/20	pick up cream cheese	19/20
pick up black bowl from table center	20/20	pick up salad dressing	20/20
pick up black bowl on the cookie box	19/20	pick up bbq sauce	19/20
pick up black bowl in the top drawer	20/20	pick up ketchup	20/20
pick up black bowl on the ramekin	20/20	pick up tomato sauce	18/20
pick up black bowl next to the cookie	20/20	pick up butter	20/20
pick up black bowl on the stove	19/20	pick up milk	20/20
pick up black bowl next to the plate	20/20	pick up chocolate pudding	20/20
pick up black bowl on the wooden cabinet	20/20	pick up orange juice	20/20
Suite total / SR	**198/200 (99.0%)**	**Suite total / SR**	**195/200 (97.5%)**

LIBERO-Goal	Bridge	LIBERO-10	Bridge
open the middle drawer of the cabinet	20/20	put both alphabet soup and tomato sauce	20/20
put the bowl on the stove	19/20	put both cream cheese box and butter	20/20
put the wine bottle on top of cabinet	20/20	turn on the stove and put the moka pot	19/20
open the top drawer and put the bowl	19/20	put the black bowl in the bottom drawer	20/20
put the bowl on top of cabinet	20/20	put the white mug on the left plate	19/20
push the plate to the front of stove	20/20	pick up the book and place it in back	19/20
put the cream cheese in the bowl	20/20	put the white mug on the plate	18/20
turn on the stove	20/20	put both alphabet soup and cream cheese	19/20
put the bowl on the plate	20/20	put both moka pots on the stove	18/20
put the wine bottle on the rack	17/20	put the yellow and white mug in micro	15/20
Suite total / SR	**195/200 (97.5%)**	**Suite total / SR**	**187/200 (93.5%)**

RoboCasa per-task breakdown. Table 8 lists the 24 RoboCasa kitchen tasks with per-task SR, grouped by category (matching the main paper's RoboCasa cross-benchmark row).

C BRIDGE QUALITY ANALYSIS

Validation cosine during training. The R0 bridge converges within 30–50 epochs to $\sim$0.97 per-token cosine on held-out sync data (image tokens, averaged over layers for $\pi_{0.5}$). R1 (DAgger fine-tuning) resumes from R0 and converges rapidly in 10–16 epochs, adding only 0.002–0.005 in cosine but translating to 3–13pp SR gains in closed-loop simulation. LR 10$\times$ lower (3×10^{-5}) is required for GR00T LIBERO-10 DAgger to avoid collapse; shorter-horizon suites tolerate the full LR.

Temporal similarity of prefix representations. Table 9 lists the consecutive-step cosine similarity of the per-layer prefix hidden states of the pretrained $\pi_{0.5}$ backbone summarized in Section 3.2 (cosine per token, averaged over tokens, then step-pairs, then episodes; layers indexed 0 (embedding) to 18 (final); 15 real DROID and 15 sim LIBERO episodes, 705 consecutive-step pairs each). Image tokens are nearly invariant in early and middle layers and dynamic at the final layer; text tokens are an order of magnitude more stable at every depth but not perfectly invariant, drifting to 0.9966 (real) / 0.9954 (sim) at the deepest layer through bidirectional prefix attention to visual tokens.

Per-layer KV prediction quality ($\pi_{0.5}$). The $\pi_{0.5}$ bridge predicts pre-RoPE K and V for all 18 Gemma layers. Table 10 reports per-token cosine on held-out sync data. Early layers (L0–L5) are easiest to predict (cos 0.999+) because their KV depends mostly on the (fixed) text prompt; later layers (L10–L17) encode more observation-dependent content and are harder. The bridge's relative

Table 8: Per-task success rates on RoboCasa (GR00T, zero-shot bridge, 20 episodes/task, f=3). Bridge retains 95.4% of sync SR on average, matching sync or exceeding it on 9 of 24 tasks.

Category	Task	Sync	Cache	Bridge
Door/Drawer	OpenCabinet	85	75	85
Door/Drawer	CloseCabinet	85	70	80
Door/Drawer	OpenDrawer	80	70	80
Door/Drawer	CloseDrawer	80	65	75
Door/Drawer	OpenFridge	70	60	65
Door/Drawer	CloseFridge	70	60	70
Appliance	TurnOnStove	85	70	85
Appliance	TurnOffStove	85	70	80
Appliance	CoffeePressButton	80	65	80
Appliance	CoffeeSetupMug	80	65	75
Appliance	TurnOnMicrowave	80	60	75
Appliance	TurnOffMicrowave	75	60	70
Appliance	TurnOnSinkFaucet	80	65	75
Coffee	ServeCoffee	55	45	60
Coffee	CoffeePourToMug	50	40	50
Coffee	CoffeeCleanUp	55	40	60
Pick-and-Place	PnPCounterToCab	55	30	50
Pick-and-Place	PnPCabToCounter	50	25	45
Pick-and-Place	PnPCounterToStove	50	30	45
Pick-and-Place	PnPStoveToCounter	45	25	40
Pick-and-Place	PnPCounterToSink	55	30	50
Pick-and-Place	PnPSinkToCounter	50	25	40
Pick-and-Place	PnPCounterToMicro	45	20	40
Pick-and-Place	PnPMicroToCounter	45	30	40
Average (24 tasks)		**66.2**	**49.8**	**63.2**

Table 9: Consecutive-step cosine similarity of $\pi_{0.5}$ prefix hidden states by layer, for image and text tokens, on real (DROID) and simulated (LIBERO) closed-loop trajectories.

	L0	L3	L9	L15	L18
image, real	0.999	0.9995	0.999	0.993	**0.983**
image, sim	0.997	0.999	0.998	0.986	**0.963**
text, real	0.9993	0.9998	0.9997	0.9994	**0.9966**
text, sim	0.9992	0.9997	0.9996	0.9991	**0.9954**

gain over the copy baseline therefore concentrates in the mid layers (e.g., L10: $+0.004$ cosine), while at the deepest layer (L17) the copy baseline is already very strong (cos 0.995) and the bridge is essentially tied with it—consistent with consecutive deep-layer KVs varying mostly within the noise floor that a residual predictor cannot exploit.

Table 10: Per-layer per-token cosine of predicted next-step KV vs. ground truth ($\pi_{0.5}$ bridge, LIBERO-Spatial val set). Copy baseline is cosine between consecutive KV without bridge prediction.

Metric	L0	L5	L10	L17
Bridge (Ours)	0.999	0.999	0.997	0.994
Copy baseline	0.997	0.998	0.993	0.995

Error compounding across bridge offsets. The bridge chains predictions autoregressively: at bridge offset k, the input is the bridge's own prediction from $k-1$ steps back. Table 11 shows that chained prediction cosine degrades gracefully—explaining why SR remains stable at moderate f.

D Full Frequency Sweep

Table 12 presents the complete frequency sweep for $\pi_{0.5}$ across all four LIBERO suites (the data visualized in the main paper's frequency–SR figure). Spatial, Object, and Goal retain >94% SR through f=8; LIBERO-10 degrades earlier due to longer episodes compounding bridge drift.

Table 11: Chained-prediction cosine vs. bridge offset for $\pi_{0.5}$ KV bridge (LIBERO-Spatial). Offset k means the bridge has chained k times since the last VLM call.

Offset k	1	2	3	5
Bridge cos (avg over 18 layers)	0.998	0.996	0.994	0.989
Copy-only cos	0.993	0.988	0.983	0.974

Table 12: Full frequency sweep on $\pi_{0.5}$ across LIBERO suites (20 episodes/task, 1-step denoise). Boldface marks the default f=4 used in the main table.

f	Spatial	Object	Goal	LIBERO-10	VLM savings
1 (sync)	98.7	98.3	97.0	94.0	0%
2	98.5	98.0	97.0	93.5	50%
3	99.0	100	99.0	92.5	67%
4	**99.0**	**97.5**	**97.5**	**93.5**	**75%**
5	97.5	99.5	96.0	91.0	80%
6	98.0	96.5	98.5	87.5	83%
8	95.5	96.5	94.5	84.5	88%
12	93.0	97.5	89.5	77.0	92%

E Latency Breakdown

Component-wise latency on A100 80GB with bfloat16 and `torch.compile`. Values measured with GPU warmup (first 5 calls excluded).

Table 13: Per-component inference latency (ms, median). GR00T runs a lightweight feature bridge with no vision encoder on bridge steps; $\pi_{0.5}$ runs SigLIP every step to provide fresh embedding input to the KV bridge.

Component	GR00T-N1.6-3B	$\pi_{0.5}$
VLM backbone (full)	63	46 (Gemma-2B prefix)
Vision encoder (on bridge step)	–	5 (SigLIP)
Action head	27 (DiT 32L)	30 (1-step denoise)
Bridge (bf16 + compile)	2	1
Sync total (backbone + head)	90	76
Bridge total (bridge + head)	29	36
Avg per-step @ f=3	49	49
Avg per-step @ f=4	44	46
Net speedup (with ep. len correction)	1.73×	1.55×

Effect of bf16 and torch.compile. Without optimization, the $\pi_{0.5}$ KV bridge runs at 16.6ms per call (fp32, eager attention). Casting to bfloat16 reduces this to 6.0ms. Adding `torch.compile(mode="max-autotune")` further drops it to 1ms for the bridge model alone (total bridge-step cost 6ms including the 5ms SigLIP forward). The bf16 + compile stack is required to achieve the reported 1.55× net speedup.

F Additional Ablations

Sync + DAgger data mixing. R1 training concatenates the sync dataset with the DAgger dataset at their natural sizes (no reweighting). Table 14 ablates this choice.

Chunk-length study. With the executed chunk lengthened from 5 to 10 actions per inference, sync is unchanged (97.50% at chunk 10, measured directly over four suites, 3 seeds, n=240, per-suite 98.3/96.7/96.7/98.3, vs. 96.96% at chunk 5), stale reuse falls from 56.38% to 38.92%, and a bridge re-trained for the length-10 chunk holds 95.83% (per-suite 98.3/96.7/97.0/91.3). Chunking amortizes the whole stack by acting open-loop, whereas the bridge amortizes only the backbone and keeps the action head reactive; a longer open-loop chunk widens the stale-vs-fresh feature gap, which is why stale reuse degrades while the bridge composes with chunking.

Epoch count and LR schedule. DAgger R1 fine-tuning uses cosine LR decay with $T_{\max}$=16 for GR00T Goal and $T_{\max}$=100 for LIBERO-10 (longer horizon requires more gentle adaptation). Shorter schedules fail to close the distribution gap; longer schedules risk over-fitting to DAgger noise.

Table 14: Effect of sync:DAgger data ratio on R1 bridge SR (GR00T LIBERO-Goal, f=3). Mixed is the default, producing the best performance.

R1 training data	SR
Sync only (R0 baseline)	91.0
DAgger only	92.5
Sync + DAgger (mixed, default)	**95.0**

G TRANSFERABILITY

Cross-suite bridge transfer ($\pi_{0.5}$). We test whether a bridge trained on one LIBERO suite transfers zero-shot to others. Table 15 shows that the Spatial bridge partially transfers to Object (94.0%) but degrades on Goal (79.0%) and LIBERO-10 (77.5%). Bridges learn checkpoint- and scene-specific dynamics, supporting per-checkpoint training (bridge training is fast—a few hours per checkpoint on a single GPU; Appendix A).

Table 15: Cross-suite transfer of the $\pi_{0.5}$ Spatial bridge (f=3). Bridges learn suite-specific dynamics and transfer partially.

Target suite	Direct training (R1, f=3)	Spatial-bridge transfer
Spatial	99.0%	99.0% (native)
Object	100.0%	94.0%
Goal	99.0%	79.0%
LIBERO-10	92.5%	77.5%

RoboCasa 12-task → 24-task transfer (GR00T). We also test transfer within RoboCasa: train bridge on 12 tasks, evaluate on held-out 12. The bridge achieves 58.1% SR on held-out tasks (sync: 63.4%), a 91.6% retention, confirming partial within-benchmark generalization across task categories.

H BASELINE IMPLEMENTATION DETAILS

Token-pruning baselines (FastV, VLA-Cache) were originally designed and evaluated on single-system VLAs (e.g., OpenVLA) where the LLM backbone autoregressively generates action tokens. We re-implemented both from their official repositories and adapted them to the dual-system architectures studied here. We document below the full adaptation choices, hyperparameters, and failure modes.

FastV adaptation. FastV (Chen et al., 2024) performs early-layer attention-based visual token pruning.

Hyperparameters. We use the default FastV configuration from the original paper: prune ratio $R = 0.5$ (drop 50% of image tokens) at layer $K = 2$, selecting tokens by average attention weight from all text tokens to each image token across the first K layers. We also swept $R \in \{0.25, 0.50, 0.75\}$ and $K \in \{2, 4, 6\}$ per VLA and report the best SR configuration in the main table (R=0.5, K=2 in both cases); more aggressive pruning (e.g., R=0.75 at K=2) collapsed SR to <20%.

Integration point. For GR00T-N1.6-3B, pruning is applied on the Qwen3 LLM side (after the SigLIP2 vision encoder + projection), at transformer layer 2. For $\pi_{0.5}$, pruning is applied in the PaliGemma prefix computation at Gemma layer 2. In both cases the pruned tokens are dropped outright: they never appear in any layer deeper than the pruning point, so the backbone processes a genuinely shorter sequence (image tokens halved after layer 2), which is what yields the measured per-inference latency reduction.

Why SR degrades. Dual-system action heads attend densely to *all* VLM hidden-state positions (either the final feature grid for GR00T's DiT head, or the per-layer KV cache for $\pi_{0.5}$'s flow-matching expert). Dropping 50% of image-token positions removes half of the visual context the pretrained action head was trained to consume and re-shapes the backbone-to-head interface. SR drops by an average of ∼18pp; the degradation is largest on long-horizon LIBERO-10 (−34pp), consistent with the action head compounding errors from the missing context across the 8-step action chunks. The large drop on dual-system VLAs is thus an architectural-sensitivity finding inherent to dense-attention action heads.

VLA-Cache adaptation. VLA-Cache (Xu et al., 2025) reuses image-token KV entries across consecutive timesteps when inter-frame patch-wise visual similarity is high.

Hyperparameters. We use the original paper's reported hyperparameters: per-patch cosine-similarity threshold $\tau = 0.1$, reuse policy "cache if cosine$\geq 1-\tau$, else refresh". We also swept $\tau \in \{0.05, 0.10, 0.20\}$ and report the best SR ($\tau = 0.10$) in the main table; a tighter threshold (τ=0.05) yielded higher SR at the cost of almost no latency reduction (<5%).

Integration point. Identical refresh mask is applied to every transformer layer's KV cache for image-token positions; text-token KVs are always refreshed. We do not modify the action head; it consumes the (partially cached) KV cache as-is.

Why the speedup ceiling is low. Because VLA-Cache still runs a full forward pass through the LLM backbone—only cached-position K/V are skipped—the latency reduction is bounded by the fraction of backbone compute spent on image-token K/V projections (roughly 15–20% on both VLAs). Observed per-step latency reduction is 14–17%, yielding a net 1.08× wall-clock speedup. This is the upper bound of the method on dual-system architectures; Latent Bridge exceeds it by replacing the backbone entirely on bridge steps, achieving 39–45% per-step reduction and 1.55–1.73× net speedup.

Amdahl ceiling: why token-pruning speedups shrink on dual-system VLAs. The modest net speedup of FastV and VLA-Cache in the main paper's headline results table is not an implementation artifact but a structural consequence of Amdahl's law applied to the dual-system latency breakdown.

On *single-system* VLAs (e.g., OpenVLA), the LLM backbone is the *entire* inference pipeline—the same LLM produces action tokens directly—so any reduction in LLM latency translates 1-to-1 into end-to-end speedup. In that setting, FastV and VLA-Cache each report roughly 1.3–1.5× wall-clock speedup in their original papers.

On *dual-system* VLAs the same pruning only accelerates the ***first*** system (the VLM backbone), while the second system (the action head) runs unchanged at every step. Using our measured breakdown (Table 13), the backbone fraction of per-step sync cost is $63/90 = 70\%$ on GR00T and $46/76 = 60\%$ on $\pi_{0.5}$. Even if token pruning gave a 1.5× local speedup on the backbone (its single-system ceiling), the per-step net speedup on dual-system would be bounded by Amdahl's law at

$$\frac{1}{(1-p) + p/s_{\text{local}}} = \frac{1}{0.30 + 0.70/1.5} \approx 1.30\times \quad \text{(GR00T)}, \qquad \approx 1.23\times \quad (\pi_{0.5}),$$

and in practice the local speedup is well below 1.5× on both backbones (VLA-Cache: ∼15% backbone reduction $= s_{\text{local}} \approx 1.18$; FastV: similar). The observed 1.08× net speedup for VLA-Cache in the main results tracks this bound.

Latent Bridge escapes this ceiling by replacing the backbone *entirely* on bridge steps: bridge step cost is 2–6 ms versus a 46–63 ms VLM backbone, a ≈10–30× local speedup at the backbone-step level. Even after amortizing over the remaining action-head cost and the mix of VLM and bridge steps within period f, the net per-step latency drops to 46–49 ms on both VLAs—yielding the reported 1.55–1.73× net wall-clock speedup, well beyond the Amdahl-bounded ceiling of token-pruning methods on dual-system architectures. This is the central architectural reason why delta prediction dominates token pruning in this setting: skipping a call is fundamentally cheaper than making an existing call faster.

I Qualitative Results

We present qualitative comparisons on three representative tasks spanning LIBERO-Spatial, LIBERO-Object, and LIBERO-10. For each task, we show (i) agentview trajectory frames at four representative timesteps under Sync (ground truth), Latent Bridge (f=3), and Feature Cache (Δ_t=0); and (ii) per-step KV cache cosine similarity to ground truth. All runs share the same initial environment state and random seed.

Across all cases, the Latent Bridge maintains near-unity cosine similarity to the ground-truth KV cache on non-VLM steps, producing trajectories visually indistinguishable from the Sync baseline. Feature Cache exhibits sawtooth degradation in KV fidelity between VLM refreshes, leading to compounding action errors and task failures.

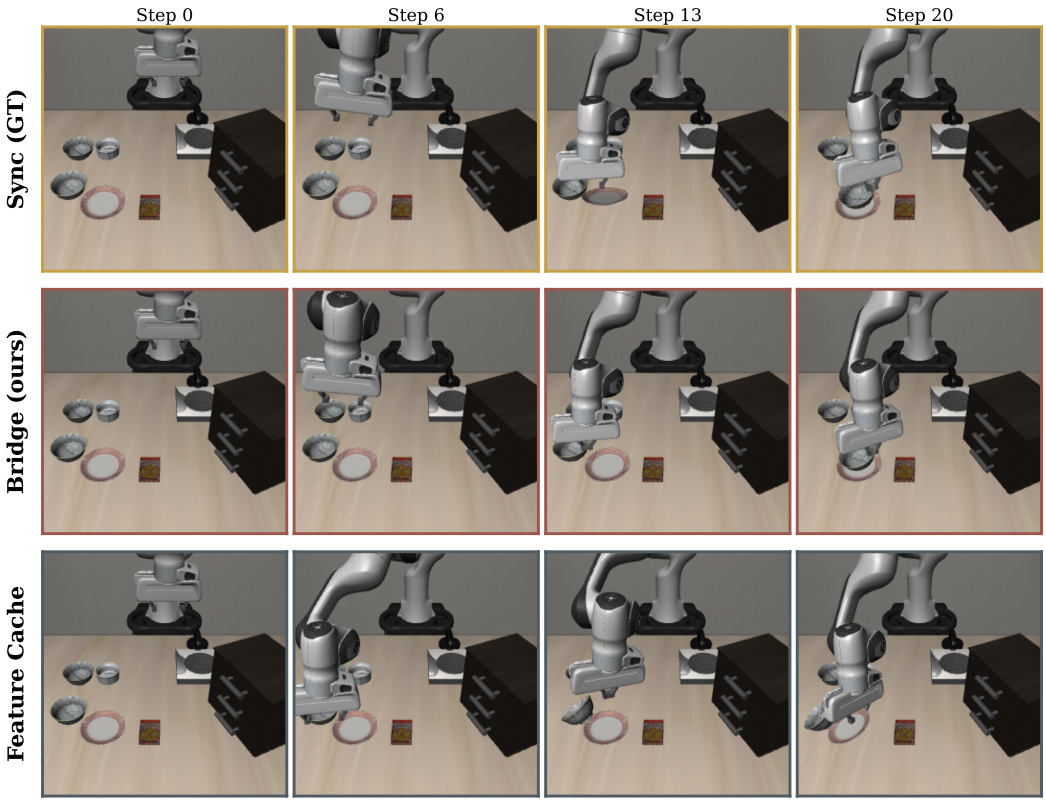


Figure 6: LIBERO-Spatial (*pick up the black bowl next to the plate and place it on the plate*): trajectory comparison. **Row 1 (Sync):** VLM runs every step—task succeeds. **Row 2 (Bridge):** KV deltas predicted by Latent Bridge (f=3)—task succeeds with near-identical trajectory. **Row 3 (Feature Cache):** stale KV reused without update—robot deviates and fails.

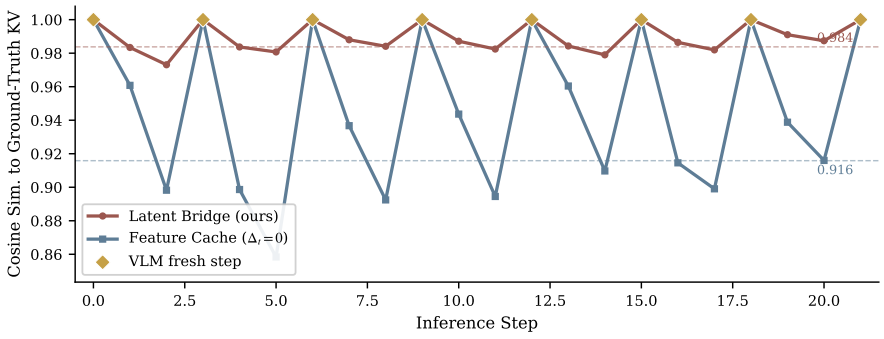


Figure 7: LIBERO-Spatial Task 8: KV cache cosine similarity to ground truth over one episode.

J LIMITATIONS

State-in-prompt configurations. For VLA variants that tokenize proprioceptive state into the language prompt (e.g., $\pi_{0.5}$ with `discrete_state_input=True`), the bridge's cached prefix KV carries stale state when reused across bridge steps, degrading performance. In our experiments, the ALOHA $\pi_{0.5}$ model required fine-tuning with `discrete_state_input=False` (state routed through the action expert instead of the language prompt) to enable bridge compatibility. This limitation is specific to state-in-prompt designs and does not affect the standard LIBERO $\pi_{0.5}$ or GR00T configurations.

Online DAgger requires a simulator. The DAgger refinement step of our pipeline requires deploying the bridge in simulation to collect on-policy data. For real-robot deployment, the bridge would need to be trained entirely offline on sync data (R0), potentially with augmented data generation. Our R0 bridges recover 93–96% of sync SR on LIBERO, and an R0-only bridge trained purely offline



Figure 8: LIBERO-Object (*pick up the ketchup and place it in the basket*): trajectory comparison. Bridge produces a near-identical trajectory to Sync; Feature Cache deviates due to stale KV.

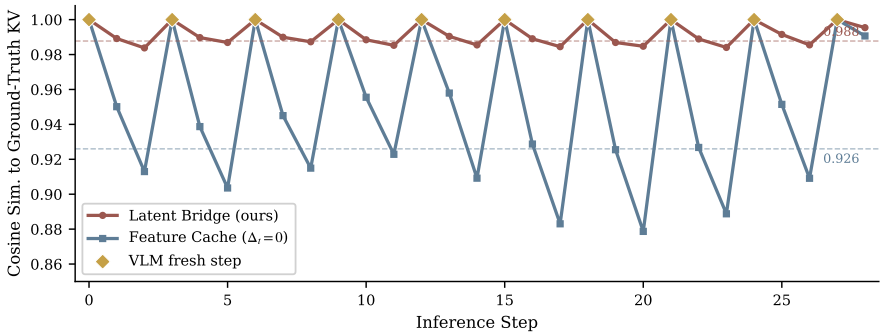


Figure 9: LIBERO-Object Task 4: KV cache cosine similarity to ground truth over one episode.

on real robot streams beats stale reuse at every skip period (Appendix K), suggesting R0-only training is viable when a simulator is unavailable.

Per-checkpoint training. The bridge is tied to a VLA checkpoint, not to a task: changing the fine-tuned model requires retraining the bridge. In LIBERO every suite has its own fine-tuned checkpoint for both GR00T and $\pi_{0.5}$, so per-suite and per-checkpoint training coincide; on RoboCasa one pretrained checkpoint serves all 24 tasks, and a bridge trained on 12 of them retains 91.6% on the 12 held-out tasks (Appendix G), so the relevant granularity is the checkpoint. Since bridge training is lightweight (2–4 hours on a single GPU; amortization in Appendix A), this adds modest overhead but limits "drop-in" deployment across frequently updated VLAs.

Episode-length side effect. Bridge-driven rollouts are slightly longer than sync rollouts (+5–6% control steps on successful episodes) because the policy occasionally takes minor detours before task completion. This partially offsets per-step savings but is already accounted for in our reported net speedup numbers (1.55–1.73×).

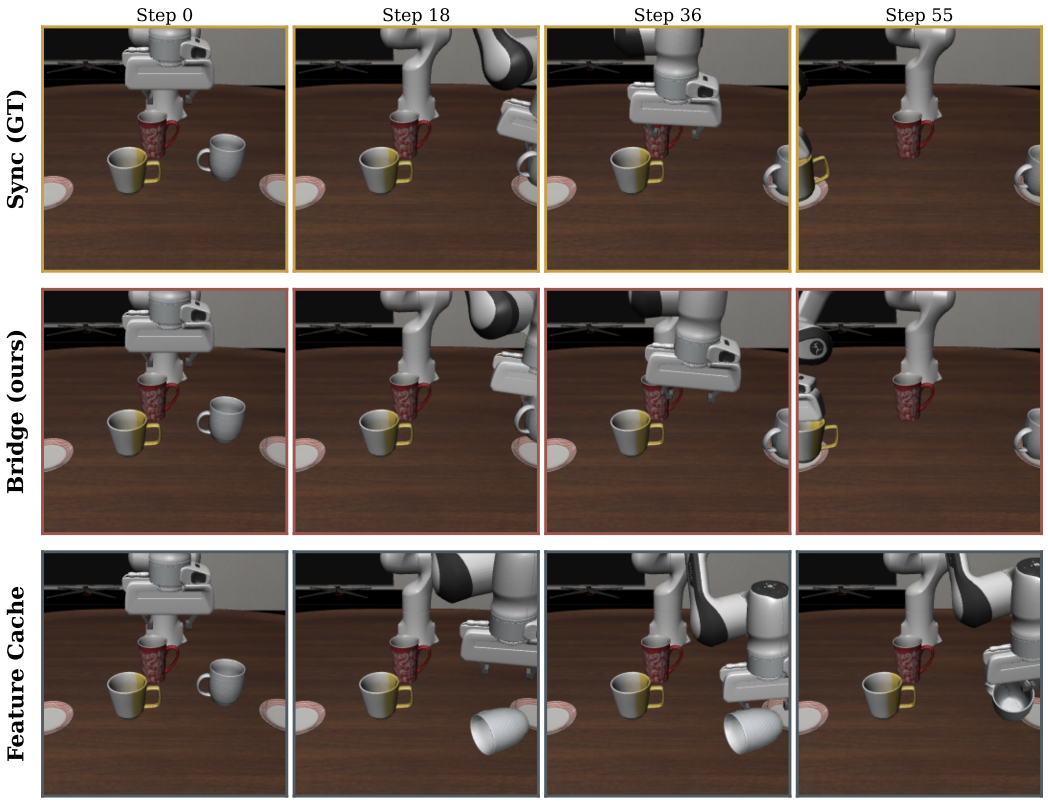


Figure 10: LIBERO-10 (*put the white mug on the left plate and put the yellow and white mug on the right plate*): trajectory comparison. This long-horizon, multi-step task highlights Bridge's ability to maintain KV fidelity over extended episodes.

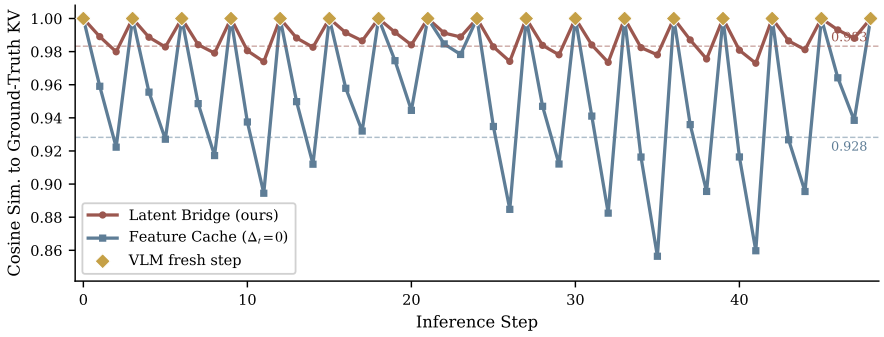


Figure 11: LIBERO-10 Task 4: KV cache cosine similarity to ground truth over one episode.

K Open-Loop Evaluation on Real Robot Streams

All closed-loop success rates in the main text are simulation numbers (Section 5). This appendix reports the open-loop evidence on real robot camera streams referred to there. The bridge never interacts with the environment: it approximates the policy's internal computation, and the environment sees only the resulting actions, so what transfer to a physical robot depends on is (i) whether the approximation holds on real observations, which the measurements below test directly at the feature and action levels, and (ii) whether the resulting small per-step action deviations compound over a closed loop, which the simulated closed-loop evaluation of Section 4 measures and the DAgger stage is designed to control. The remaining risk—that (ii) behaves differently under physical dynamics—is exactly what a hardware evaluation would answer.

Protocol. All measurements use the same pretrained $\pi_{0.5}$ backbone, the same pre-RoPE per-layer KV extraction, and the same cosine metric as the main text; the only variable between the real and

simulated arms is the pixel content. Real streams are held-out episodes from the DROID dataset (15 fps); simulated streams are LIBERO rollouts (10 fps). Bridge and stale-reuse numbers are reported with the same skip period f as in the main text, and Taylor extrapolation is the training-free order-1 (and order-2) polynomial forecast of TaylorSeer (Liu et al., 2025a) ported to control steps.

Real-vs-sim temporal redundancy. Over 25 real DROID episodes (1800 backbone inferences) and 25 sim LIBERO episodes, we compare prefix-KV cosine to the fresh backbone at matched wall-clock skip interval ($\Delta t \approx 0.2$s: stride 3 at 15 fps real, stride 2 at 10 fps sim). Real streams give 0.973 vs. 0.950 in sim, with real $\geq$ sim at every one of the 18 layers. The temporal redundancy the bridge exploits is therefore present on real sensor data and, if anything, stronger there; it is not a rendering artifact.

Offline R0-only bridge on real data. A bridge with the identical architecture is trained purely offline on real DROID KV sequences—no simulator and no DAgger, the regime a real deployment without a simulator would use—and evaluated on held-out episodes by chained recovery at skip period f (Table 16). The bridge exceeds both stale reuse and Taylor extrapolation at every f; at $f{=}2$ its next-step KV cosine of 0.982 vs. 0.975 for stale reuse is a $\sim$29% reduction of the stale-feature error.

Table 16: Chained recovery on held-out real DROID episodes: mean recovered-KV cosine to the fresh backbone over the bridge steps of each skip period f, for an R0-only bridge trained fully offline on real trajectories, stale reuse, and Taylor extrapolation (order 1 / order 2).

f	Bridge (R0, offline)	Stale reuse	Taylor (order-1)	Taylor (order-2)
2	**0.982**	0.975	0.961	0.953
3	**0.978**	0.969	0.951	0.941
4	**0.975**	0.965	0.942	0.929
6	**0.967**	0.955	0.926	0.913
8	**0.961**	0.949	0.913	0.897

Per-step action fidelity. For each held-out episode we roll out the sync policy once, saving every inference-step observation; each fill mode then replays the identical observation stream with identical per-step flow-matching noise (seeded), so the only varying factor is the injected KV. We report RMSE between the executed action chunk and the sync-conditioned reference, split into VLM steps (sanity: exactly zero for all modes with matched seeds) and skip steps. The sampling-noise floor is measured by replaying sync with re-sampled noise. Table 17 extends Table 5 of the main text with the real-stream columns: on real streams the bridge's action deviation (0.098) is roughly half that of stale reuse (0.176) and sits near the policy's own sampling-noise floor (0.037), while the closed-form baselines sit $\approx 5\times$ above it; the ranking is identical to the simulated one on every column.

Table 17: Per-step recovered-KV cosine and per-step action RMSE vs. the sync-conditioned reference on simulated (LIBERO) and real (held-out DROID) streams at $f{=}4$, alongside the simulated closed-loop SR ($\pi_{0.5}$, 3 seeds). The policy's own sampling-noise floor on real streams (sync replayed with re-sampled flow noise) is 0.037. All columns rank identically.

Method	KV cos (sim / real)	action RMSE (sim / real)	mean sim SR	SR retention
Sync	1.00 / 1.00	0 / 0 (ref.)	96.96	100.0%
Bridge	**0.976 / 0.981**	**0.136 / 0.098**	**96.92**	**99.96%**
Stale reuse	0.951 / 0.974	0.279 / 0.176	56.38	58.15%
Taylor (order-1)	0.910 / 0.951	0.298 / 0.189	46.67	48.13%

Motion-binned analysis. Consecutive real DROID frame pairs are binned by end-effector translation magnitude into terciles. Stale-reuse recovery cosine correlates with motion at $r{=}{-}0.68$ and falls from 0.988 (low bin) to 0.962 (high bin); the bridge holds 0.993 to 0.980, so its advantage grows from $+0.005$ to $+0.019$ ($4\times$) under fast motion. Delta prediction is most valuable exactly where staleness is worst, which is also the regime phase-aware scheduling (Section 3.4) targets.