FORESENSE: LEARNING LATENT VISUAL INTUITION FOR VISION-LANGUAGE-ACTION MODELS VIA FUTURE EXPERIENCE DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

The success of reasoning in large language models has inspired growing efforts to equip vision-language-action (VLA) models with similar reasoning capabilities. Existing approaches, however, typically externalize such reasoning either through verbal chains of thought or through explicit predictive representations of future evolution. We argue that reasoning for action generation should take the form of an abstract intuition about future dynamics, without being constrained to explicit representational formats or predefined semantic or temporal roles for intermediate reasoning states. To this end, we introduce **ForeSense**, a VLA framework that learns to sense how a scene may evolve by distilling privileged future experience into **latent visual intuition**. ForeSense is built around three key designs. 1) A **Future-Conditioned Attention Bottleneck** channels information from future observations through an autoregressive latent chain, which is then distilled into a current-only reasoner to form latent visual intuition without predefined semantic or temporal roles. 2) To support progressive refinement within this latent reasoning process, **Detached Latent Curriculum (DeLaC)** gradually extends the chain while applying stop-gradient to the existing latent prefix, encouraging each new step to refine previously learned reasoning states while mitigating latent-chain collapse. 3) To translate latent visual intuition into actions, **Reasoning-to-Action Fusion (ReAct-Fuse)** integrates the frozen reasoner's latent key-value memory into each action-expert layer alongside the native VLA attention pathway, enabling fusion across heterogeneous reasoner and VLA backbone architectures. Extensive experiments on simulated and real environments demonstrate the strong and consistent performance of ForeSense, with success rates of 97.3% on LIBERO, 85.9% on LIBERO-Plus, and 79.2% on real-world tasks. Notably, the largest robustness gain appears under **language perturbations**, where success rises by 4.9 points from 86.3% to 91.2% over the $\pi_{0.5}$-LIBERO base policy.

1 INTRODUCTION

Vision-language-action (VLA) models have emerged as a promising paradigm for general-purpose robotic manipulation, mapping visual observations and language instructions to executable actions (Kim et al., 2025b; Black et al., 2024; 2025). Yet action generation often depends on more than the scene currently visible to the policy. To manipulate an object, a robot must anticipate how the scene may change during interaction and use that expectation to guide its actions. This need becomes especially salient when instructions vary, interactions unfold over multiple steps, or the environment differs from training conditions.

The success of reasoning in large language models has inspired efforts to give VLA models similar capabilities. Some approaches generate verbal chains of thought before acting (Zawalski et al., 2025; Lee et al., 2025). Others use predicted future observations or structured representations of future evolution to guide actions (Zhao et al., 2025; Zhang et al., 2025). These approaches show that anticipating future outcomes can benefit action generation. They also motivate a further question: can a policy acquire useful knowledge of future dynamics without requiring its intermediate reasoning states to correspond to words, rendered observations, or particular future time steps?

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.

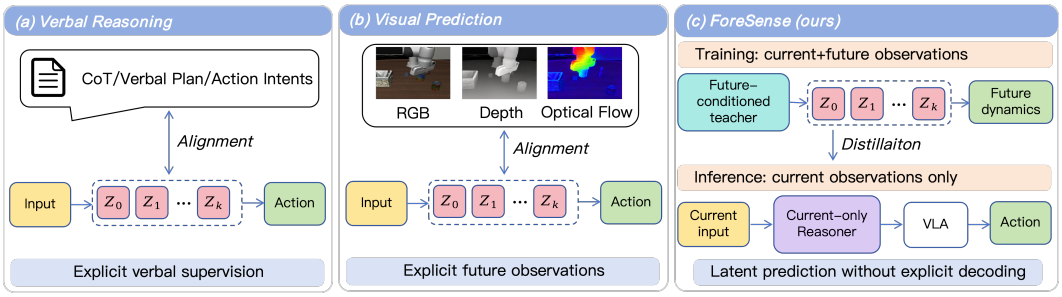


Figure 1: **Comparison of reasoning paradigms for VLA models.** (a) Verbal reasoning uses explicit textual intermediate representations. (b) Visual prediction uses explicit representations of future evolution, such as RGB, depth, or optical flow. (c) ForeSense distills a future-conditioned teacher's autoregressive latent chain into a current-only reasoner. Action generation is conditioned on the reasoner through an additional cross-attention pathway, while keeping the original VLA backbone. Individual latent states are not assigned or aligned with predefined semantic or temporal roles.

We view this capability as *latent visual intuition*: an internal, action-relevant anticipation of how an interaction may unfold over a short horizon. In familiar manipulation, humans often act from experience without first verbalizing future steps or constructing a precise mental image of the next scene. This analogy motivates a VLA that represents near-term interaction dynamics directly in latent space rather than generating a long verbal plan or rendered future frames. During training, recorded future observations show how interactions actually develop, while descriptions generated by a strong pretrained VLM provide future-dynamics targets. Learning this latent process poses several challenges. Short chains may lack capacity, whereas naively extending them can lead to collapse. The model can learn from observed futures during training, but must anticipate future dynamics from the current observation and instruction at inference time. Finally, a backbone well suited to reasoning may not be equally suited to action generation, motivating its integration with an existing VLA policy.

To address these challenges, we introduce ForeSense (Figure 1c), a four-stage framework that distills future experience into autoregressive latent reasoning for action generation. We construct a LIBERO dataset of demonstrations annotated with future-dynamics descriptions generated by GPT-6 Astra. First, a future-conditioned teacher learns to predict these descriptions from the instruction and current and future observations. Second, an attention bottleneck forces the model to reason through an autoregressive latent chain before describing the observed future dynamics. We train this chain with DeLaC, gradually extending it while applying stop-gradient to the existing prefix to encourage refinement and mitigate collapse. Third, we distill the teacher's latent chain into a current-only reasoner while retaining future-dynamics prediction as a training objective. Finally, ReAct-Fuse integrates the frozen reasoner's latent key-value memory into each action-expert layer alongside the native VLA attention pathway, enabling the policy to generate actions guided by latent visual intuition about future dynamics.

We evaluate ForeSense on LIBERO (Liu et al., 2023), LIBERO-Plus (Fei et al., 2026), and six real-world manipulation tasks. Relative to the $\pi_{0.5}$-LIBERO base policy, overall success increases from 96.9% to 97.3% on LIBERO and from 84.3% to 85.9% on LIBERO-Plus. The largest gain among LIBERO-Plus perturbation categories occurs under language perturbations, where success rises from 86.3% to 91.2%. Average success across the six real-world tasks increases from 60.0% to 79.2%. We further examine how the number of latent states affects future-dynamics prediction and downstream action success.

Our contributions are:

- We introduce **ForeSense**, a four-stage training framework that distills future-conditioned autoregressive latent reasoning into a current-only VLA policy.
- We develop **Detached Latent Curriculum (DeLaC)**, which gradually extends the chain while applying stop-gradient to its existing prefix to mitigate latent-chain collapse.

- We propose **Reasoning-to-Action Fusion (ReAct-Fuse)**, which conditions each action-expert layer on the frozen reasoner's latent key-value memory alongside the native VLA attention pathway, allowing different reasoning and action backbones to be combined.
- We construct a LIBERO dataset with VLM-generated future-dynamics text annotations to supervise the learning and distillation of latent reasoning.

2 Related Work

2.1 Vision-Language-Action Models

VLA models use pretrained vision-language representations for robot control (Zitkovich et al., 2023; Kim et al., 2025b). Recent systems, including π_0 and $\pi_{0.5}$, combine multimodal backbones with continuous action generation (Black et al., 2024; 2025). Other work studies efficient action representations and spatial generalization (Pertsch et al., 2025; Qu et al., 2025). Our action expert follows transformer diffusion and flow-matching formulations (Peebles & Xie, 2023; Lipman et al., 2023), while our focus is on the reasoning representation that conditions it.

2.2 Reasoning for Vision-Language-Action Models

Recent VLA methods introduce intermediate reasoning before action prediction. ECoT and ThinkAct use explicit verbal reasoning (Zawalski et al., 2025; Huang et al., 2025), while ACoT-VLA and MolmoAct introduce structured or action-centric traces (Zhong et al., 2026; Lee et al., 2025). Continuous latent reasoning is explored in Fast-ThinkAct, LaST_0, and LaRA-VLA (Huang et al., 2026; Liu et al., 2026; Bai et al., 2026), building on related ideas in language models (Hao et al., 2025). Temporal memory and spatial guidance offer complementary mechanisms for grounded or extended-horizon control (Koo et al., 2026; Ye et al., 2026). In ForeSense, an autoregressive bottleneck instead induces the latent chain from privileged future experience, without assigning predefined semantic or temporal roles to individual states.

2.3 Visual Foresight and Predictive Robot Policies

Visual-foresight methods generate future observations or subgoals to guide action generation, as in CoT-VLA, ForeAct, and VideoVLA (Zhao et al., 2025; Zhang et al., 2026; Shen et al., 2025). Other robot policies predict structured or latent future representations (Zhang et al., 2025; Routray et al., 2026). Predictive world models such as DINO-WM and V-JEPA 2 also learn representations of future states (Zhou et al., 2025; Assran et al., 2025). ForeSense uses future observations as privileged training evidence: its latent chain collectively explains future interaction dynamics, while individual states are not tied to predefined future targets.

3 Method

3.1 Overview

Given a current visual observation o_t and a language instruction l, our goal is to learn a VLA policy that internally anticipates how an interaction may evolve before generating actions. The current observation o_t contains both the third-person main-camera view and the wrist-camera view. During training, we additionally have access to future observations $o_t^+ = \{o_{t+j\Delta}\}_{j=1}^{H}$, where each future observation contains only the third-person main-camera view and is sampled every Δ control steps ($\Delta = 5$ in our experiments). These future observations record how the interaction actually unfolds. We denote the textual future-dynamics target by $y = (y_1, \ldots, y_M)$ and the autoregressive latent chain by $z_{1:K} = (z_1, \ldots, z_K)$.

We define *latent predictive reasoning* as the autoregressive generation of continuous latent embeddings before action generation. Given the current observation and instruction $X = (o_t, l)$, the reasoner produces a fixed-length chain $z_i = R_\theta(X, z_{<i})$, for $i = 1, \ldots, K$. The backbone's contextualized representations at these latent positions form a memory $\mathcal{M}_z$, which is trained to support prediction of the future-dynamics target y and later conditions the action policy. The index i specifies generation order; it does not assign an individual latent embedding to a word, future frame, or

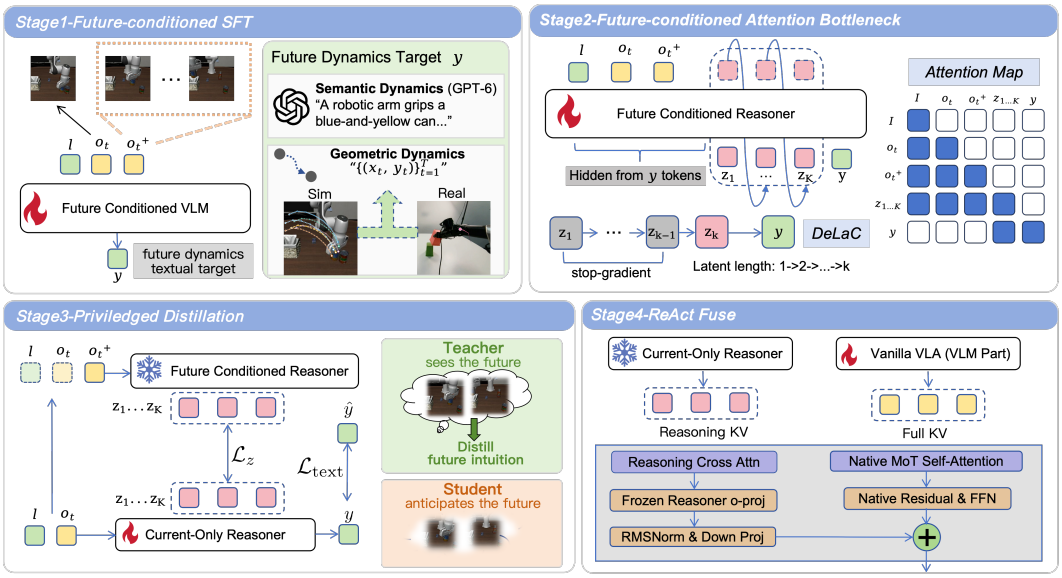


Figure 2: **Overview of ForeSense.** Stage 1 trains a future-conditioned VLM to predict textual future-dynamics targets from current and future observations. Stage 2 routes target prediction through an autoregressive latent bottleneck. The lower part of the Stage 2 panel illustrates DeLaC, which progressively extends the latent chain while detaching earlier latent input embeddings. The attention map shows block-level visibility, with causal attention within the latent and text blocks. Stage 3 distills the future-conditioned teacher into a current-only reasoner. Stage 4 conditions the VLA action expert on the frozen reasoner's latent KV memory through ReAct-Fuse. Flame and snowflake icons indicate trainable and frozen modules.

fixed temporal interval. Figure 2 shows how this latent chain is learned and integrated into the action policy across four stages.

3.2 Future-Dynamics Supervision

Rather than reconstructing future observations, we construct a textual target y describing action-relevant changes in each current–future segment. GPT-6 Astra describes gripper motion, robot–object interactions, and outcomes. We combine this description with gripper-point trajectories in the third-person camera view, obtained from simulator projection or MolmoPoint-8B (Clark et al., 2026) tracking for real-world videos. The resulting semantic and geometric annotations are serialized into y.

3.3 Stage 1: Future-Conditioned Supervised Fine-Tuning

We initialize the future-conditioned reasoner from Qwen3-VL-4B (Bai et al., 2025) and fine-tune it to predict the future-dynamics target y from the instruction and the current and future observations:

$$\mathcal{L}_{\text{stage1}} = -\sum_{j=1}^{M} \log p_\theta \left(y_j \mid y_{<j}, o_t, o_t^+, l \right). \tag{1}$$

3.4 Stage 2: Future-Conditioned Attention Bottleneck

Let P^F denote the multimodal prefix containing (o_t, o_t^+, l), and let h_0 be the hidden state at its last valid position. A learned linear bridge B_ϕ produces each continuous latent embedding from the preceding hidden state. The vision-language backbone processes these embeddings autoregressively using its accumulated key–value cache:

$$\begin{aligned} z_i &= B_\phi(h_{i-1}), \\ h_i &= F_\theta\big(z_i; \mathrm{KV}(P^F, z_{<i})\big), \qquad i = 1, \ldots, K. \end{aligned} \tag{2}$$

Here $\mathrm{KV}(P^F, z_{<i})$ denotes the cached representations of the prefix and preceding latent positions.

The information bottleneck is imposed through attention visibility. Ignoring padding, the keys visible to each query are

$$\mathcal{A}(q) = \begin{cases} P^F, & q \in P^F, \\ P^F \cup \{z_1, \ldots, z_i\}, & q = z_i, \\ \{z_1, \ldots, z_K\} \cup \{y_1, \ldots, y_j\}, & q = y_j. \end{cases} \tag{3}$$

Target-text positions cannot attend directly to the multimodal prefix; information from the observed future reaches them only through the contextualized latent positions. We train Stage 2 to predict the same future-dynamics target y as in Stage 1, under the attention mask in Eq. 3. We denote this next-token prediction loss by $\mathcal{L}_{\mathrm{Stage2}}$.

Detached Latent Curriculum (DeLaC). To encourage each added latent position to contribute predictive information, we train sequentially with $K_s = 1, \ldots, 8$, initializing each step from the preceding checkpoint. At step K_s, we use the recursion in Eq. 2 with

$$\widetilde{z}_i = \begin{cases} \mathrm{sg}(z_i), & i < K_s, \\ z_i, & i = K_s, \end{cases} \tag{4}$$

in place of z_i as the backbone input. Consequently, the gradient to the shared bridge at a curriculum step of length K passes directly through the newest latent embedding:

$$\frac{\mathrm{d}\mathcal{L}_{\mathrm{Stage2}}}{\mathrm{d}\phi} = \frac{\partial \mathcal{L}_{\mathrm{Stage2}}}{\partial \widetilde{z}_K} \frac{\partial B_\phi(h_{K-1})}{\partial \phi}. \tag{5}$$

The backbone remains trainable; detachment blocks the bridge-gradient paths through earlier latent inputs without freezing their backbone computations.

3.5 Stage 3: Privileged Distillation

The future-conditioned chain depends on o_t^+ and therefore cannot be generated directly at deployment. We distill it into a student reasoner that receives only the current observation and instruction. Let the frozen teacher R_T and trainable student R_S generate

$$z^T_{1:K} = R_T(o_t, o_t^+, l), \qquad z^S_{1:K} = R_S(o_t, l). \tag{6}$$

Latent-chain distillation. We align the teacher and student latent embeddings position by position using mean squared error:

$$\mathcal{L}_z = \frac{1}{K d_z} \sum_{i=1}^{K} \left\| z_i^S - \mathrm{sg}(z_i^T) \right\|_2^2, \tag{7}$$

where d_z is the latent embedding dimension.

The MSE term has a useful interpretation. For $X = (o_t, l)$ and any student prediction $s_i(X)$,

$$\begin{aligned} \mathbb{E}\|s_i(X) - z_i^T\|_2^2 = \mathbb{E}\left\|s_i(X) - \mathbb{E}[z_i^T \mid X]\right\|_2^2 \\ + \mathbb{E}\left\|z_i^T - \mathbb{E}[z_i^T \mid X]\right\|_2^2. \end{aligned} \tag{8}$$

Thus, MSE alone favors the component of the teacher representation predictable from the current observation and instruction. To preserve future-dynamics prediction, we also apply a next-token loss $\mathcal{L}_{\mathrm{text}}$ to y using the student's contextualized latent memory. The Stage-3 objective is

$$\mathcal{L}_{\mathrm{distill}} = \lambda_z \mathcal{L}_z + \mathcal{L}_{\mathrm{text}}. \tag{9}$$

We use $\lambda_z = 4$ to bring the latent- and text-loss gradient magnitudes on the shared backbone closer; the measurement is reported in Appendix A.

3.6 Stage 4: Reasoning-to-Action Fusion

We freeze the current-only reasoner and connect it to a $\pi_{0.5}$ action expert through Reasoning-to-Action Fusion (ReAct-Fuse). Given (o_t, l), the reasoner autoregressively generates $K = 8$ latent positions. From each of its upper 18 layers, we retain only the keys and values corresponding to those positions:

$$\mathcal{M}_z^{(m)} = \left(K_z^{(m)}, V_z^{(m)}\right), \qquad m = 18, \ldots, 35. \tag{10}$$

Separately, the VLA backbone encodes the current observation and task instruction into its native prefix cache. The action expert continues to read this cache through its original attention pathway.

At action layer $a \in \{0, \ldots, 17\}$, paired with reasoner layer $m(a) = 18 + a$, let $\bar{u}_a = \mathrm{AdaRMS}_a^{\mathrm{attn}}(u_a; \tau)$. The native pathway produces $n_a = \mathrm{Attn}_a^{\mathrm{native}}(\bar{u}_a; \mathcal{M}_{\mathrm{VLA}}^{(a)})$, using the VLA prefix cache and the action expert's own attention context.

The separate reasoning pathway projects the same action states into Qwen's query-head space and attends to the frozen latent memory:

$$\begin{aligned} Q_a &= \mathrm{RoPE}_Q\left(\mathrm{QNorm}_{m(a)}\left(W_a^Q \bar{u}_a\right)\right), \\ A_a &= \mathrm{softmax}\left(\frac{Q_a (K_z^{(m(a))})^\top}{\sqrt{d_{\mathrm{head}}}}\right) V_z^{(m(a))}, \\ r_a &= W_a^{\downarrow}\, \mathrm{RMSNorm}\left(W_{m(a)}^O A_a\right). \end{aligned} \tag{11}$$

Qwen's grouped key–value heads are repeated to match its query-head count before attention, where $d_{\mathrm{head}} = 128$ is the per-head attention dimension of Qwen3-VL-4B. $\mathrm{QNorm}_{m(a)}$ and $W_{m(a)}^O$ are the frozen query normalization and output projection of the corresponding Qwen layer. The query projection W_a^Q and the projection back to the action hidden dimension $W_a^{\downarrow}$ are trainable. The new queries use text-continuation rotary positions following the latent chain, while the cached latent keys retain their original rotary positions. Native and reasoning attention use separate softmax operations.

We combine the two updates within the action expert's existing gated attention residual, without introducing a separate reasoning gate:

$$\begin{aligned} u'_a &= u_a + g_a^{\mathrm{attn}}(\tau) \odot (n_a + r_a), \\ u_{a+1} &= u'_a + g_a^{\mathrm{ffn}}(\tau) \odot \mathrm{MLP}_a\left(\mathrm{AdaRMS}_a^{\mathrm{ffn}}(u'_a; \tau)\right). \end{aligned} \tag{12}$$

Here g_a^{attn} and g_a^{ffn} are the action expert's native residual gates. Reasoning modifies the attention update, while the feed-forward computation remains native to the action expert.

For an action target x_0, we sample noise ϵ and time τ, form $x_\tau = (1 - \tau)x_0 + \tau\epsilon$, and train the policy to predict the corresponding flow:

$$\mathcal{L}_{\mathrm{action}} = \mathbb{E}_{x_0, \epsilon, \tau}\left[\left\| v_\psi\left(x_\tau, \tau, o_t, l, \mathcal{M}_z\right) - (\epsilon - x_0)\right\|_2^2\right]. \tag{13}$$

The reasoner remains frozen throughout action-policy training; the VLA policy and ReAct-Fuse interface are optimized with this objective.

4 Experiments

4.1 Experimental Setup

We build ForeSense on top of $\pi_{0.5}$-LIBERO. Unless otherwise specified, our action-policy variants share the VLA backbone, action expert, deployment observations, and action-policy training data. The reasoner is trained through the first three stages in Sec. 3 and frozen during action-policy training. We evaluate on LIBERO (Liu et al., 2023) with 50 rollouts per task (2,000 in total), and on LIBERO-Plus (Fei et al., 2026) with one rollout per perturbed task instance, following the respective benchmark protocols. Success is determined by the benchmark's task completion criterion. Our policies are trained on standard LIBERO and evaluated on LIBERO-Plus without perturbation-specific training. Overall success pools successful rollouts across all evaluated tasks.

Table 1: **Performance on standard LIBERO.** Success rate (%) on four task suites; overall success aggregates all evaluation rollouts.

Group	Method	Spatial	Object	Goal	Long	Overall
General VLA	OpenVLA-OFT (Kim et al., 2025a)	97.6	98.4	97.9	94.5	97.1
	$\pi_{0.5}$ (Black et al., 2025)	**98.8**	98.2	98.0	92.4	96.9
Explicit / Future	ThinkAct (Huang et al., 2025)	88.3	91.4	87.1	70.9	84.4
	DeepThinkVLA (Yin et al., 2025)	96.6	99.0	96.4	**96.2**	97.0
	DreamVLA (Zhang et al., 2025)	97.5	94.0	89.5	89.5	92.6
Latent Reasoning	Fast-ThinkAct (Huang et al., 2026)	92.0	97.2	90.2	79.4	89.7
	VLA-JEPA (Sun et al., 2026)	96.2	**99.6**	97.2	95.8	97.2
	CoWVLA (Yang et al., 2026)	97.2	97.8	94.6	92.8	95.6
Ours	**ForeSense**	98.0	99.2	**98.4**	93.6	**97.3**

Table 2: **Zero-shot robustness on LIBERO-Plus.** Success rate (%) under seven perturbation categories. Cam., Lang., and Backg. denote Camera, Language, and Background. Overall success pools all rollouts rather than averaging the rates. ForeSense is trained on standard LIBERO only.

Method	Cam.	Robot	Lang.	Light	Backg.	Noise	Layout	Overall
General VLA								
OpenVLA-OFT (Kim et al., 2025a)	56.4	31.9	79.5	88.7	93.3	75.8	74.2	69.6
π_0 (Black et al., 2024)	13.8	6.0	58.8	85.0	81.4	79.0	68.9	53.6
$\pi_{0.5}$ (Black et al., 2025)	70.7	74.9	86.3	96.9	95.7	86.8	**86.2**	84.3
Explicit / Future								
DeepThinkVLA (Yin et al., 2025)	**88.5**	40.5	84.5	90.0	75.3	**94.4**	79.9	79.0
DreamVLA (Zhang et al., 2025)	26.2	17.6	67.0	77.5	71.5	53.6	43.5	48.9
Cosmos Policy (Kim et al., 2026)	69.6	51.0	89.6	97.7	85.7	87.3	83.7	79.7
Latent Reasoning								
VLA-JEPA (Sun et al., 2026)	63.3	67.1	85.4	95.6	93.6	66.3	85.1	79.5
ForeSense	70.4	**76.7**	**91.2**	**98.3**	**97.4**	88.5	**86.2**	**85.9**

On LIBERO, we use the officially reported result for the released $\pi_{0.5}$-LIBERO checkpoint (Black et al., 2025). Since the OpenPI repository[1] does not report its LIBERO-Plus performance, we evaluate the released checkpoint on the same LIBERO-Plus task instances as ForeSense. Results for other external baselines are taken from their papers or official project pages. For real-world evaluation, we compare OpenVLA (Kim et al., 2025b), π_0 (Black et al., 2024), $\pi_{0.5}$ (Black et al., 2025), and ForeSense on six tasks with 20 trials per task; task-specific success criteria are given in Appendix C.

4.2 Simulation Results

Table 1 compares ForeSense with representative VLA and reasoning-based methods on standard LIBERO. ForeSense achieves the highest overall success rate of 97.3%, improving upon its $\pi_{0.5}$ base policy (96.9%). It also improves on the Long suite, consistent with a benefit from future-aware reasoning in multi-stage manipulation.

We further evaluate zero-shot robustness under the seven LIBERO-Plus perturbation categories. As shown in Table 2, ForeSense improves overall success from 84.3% to 85.9% relative to the same base policy, with the largest gain under language perturbations (86.3% to 91.2%). It improves five of seven categories, while camera performance decreases slightly and layout remains unchanged.

[1] https://github.com/Physical-Intelligence/openpi

Table 3: **Effect of latent-chain length.** Stage-3 token accuracy and LIBERO-Plus Spatial success for different latent-prefix lengths.

K	Token Acc.	Spatial SR
1	93.54	90.34
2	95.50	90.80
4	98.13	91.22
8	**99.95**	**91.55**

Table 4: **Reasoning ablations on LIBERO-Plus.** Token accuracy is teacher-forced accuracy on the annotated training set; overall success pools all evaluation rollouts. Token accuracy is not applicable to the KV w/o reasoning control.

Variant	Token Acc.	Spatial	Object	Goal	Long	Overall
KV w/o reasoning	N/A	91.51	87.57	80.47	79.99	84.78
w/o Distillation	98.52	91.22	88.48	80.74	79.48	84.88
w/o DeLaC	91.62	91.22	88.44	79.62	79.04	84.47
ForeSense	**99.95**	**91.55**	**88.92**	**82.17**	**81.42**	**85.92**

4.3 Ablation Studies

Latent-chain length. Table 3 reports future-dynamics token accuracy for the Stage-3 current-only reasoner and zero-shot LIBERO-Plus Spatial success for the Stage-4 policy. Both are trained with $K = 8$; at evaluation, we use prefixes of length $K = 1, 2, 4, 8$ from the same trained models, without changing their parameters. Token accuracy is measured on the annotated training set with teacher forcing over the full target sequence, including formatting and coordinate tokens. The aligned trends show that using more latent states improves modeling of the supervised future-dynamics targets and is accompanied by higher downstream action success.

Learned reasoning memory. To test whether pretrained VLM features alone account for the gain, we replace ForeSense's reasoning memory with KV representations from the original Qwen3-VL backbone (Bai et al., 2025). Memory length, representation dimensions, and the action expert's cross-attention interface are matched between the two variants. As shown in Table 4, ForeSense improves overall LIBERO-Plus success from 84.78% to 85.92%, indicating that the additional KV input alone does not account for its performance.

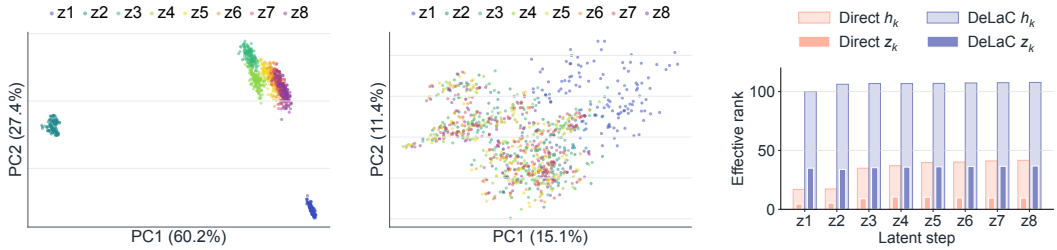
Future-to-current distillation. The *w/o Distillation* control trains a $K = 8$ current-only reasoner directly on the same future-dynamics targets, with all other model and training settings matched. Relative to this control, ForeSense improves overall success from 84.88% to 85.92% and current-only reasoner token accuracy from 98.52% to 99.95%. This comparison supports the value of learning from future observations before distilling the reasoner to current-only inputs.

Effect of DeLaC. We compare the complete DeLaC strategy with direct training of an eight-token Stage-2 chain without curriculum or detached-prefix training. Note that these two components are coupled in DeLaC and are evaluated together. As shown in Table 4, direct eight-token training achieves 91.62% future-dynamics token accuracy and 84.47% overall LIBERO-Plus success, compared with 99.95% and 85.92% for DeLaC, respectively. Using the same 128 LIBERO examples, we further collect the latent embeddings at each position. Under direct training, the first two principal components explain 87.6% of the variance, and the effective rank ranges from 4.5 to 10.5 across positions. With DeLaC, the corresponding values are 26.5% and 33.8–36.8. As shown in Fig. 3, DeLaC preserves variation across examples in substantially more directions, while also improving modeling of the supervised future-dynamics targets and downstream success, consistent with reduced low-rank representation collapse.

4.4 Real-World Evaluation

To evaluate ForeSense in real-world manipulation, we conduct experiments on the GALAXEA R1 Lite dual-arm robotic platform. All models are deployed on a workstation equipped with an NVIDIA RTX 5090 GPU. For each task, we collect 80 expert demonstrations for training. During evaluation, each model is independently rolled out for 20 trials per task under the same experimental setup, and we report the task success rate as the evaluation metric.

As shown in Fig. 4, across the six tasks, ForeSense achieves an average success rate of 79.2%, compared with 60.0% for $\pi_{0.5}$ (Black et al., 2025), 51.7% for π_0 (Black et al., 2024), and 20.8% for OpenVLA (Kim et al., 2025b). ForeSense consistently outperforms all three baselines across every task. Compared with $\pi_{0.5}$ (Black et al., 2025), the largest gains are observed on towel-wrapped



(a) Direct 8z: latent PCA (b) DeLaC 8z: latent PCA (c) Effective rank by latent position

Figure 3: **Effect of DeLaC on latent representation diversity.** (a,b) PCA of the eight Stage-2 latent positions for the same 128 LIBERO examples. PCA is fitted separately for each model; coordinates between panels are not directly comparable. (c) Effective rank of the centered representations at each position. DeLaC reduces the low-rank concentration observed under direct eight-token training.

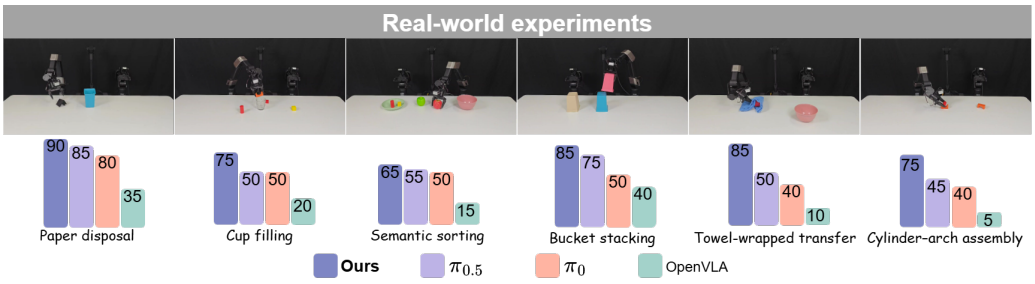


Figure 4: Real-world task setups and evaluation results. The figure illustrates the real-world experimental scenarios and the evaluation results of different models. ForeSense consistently outperforms $\pi_{0.5}$ (Black et al., 2025) and other baselines across all tasks.

transfer (+35 percentage points) and cylinder–arch assembly (+30 percentage points). These results suggest that the learned reasoning process remains effective in real-world manipulation, particularly for tasks involving coordinated object handling and precise spatial placement.

5 Limitations and Conclusion

ForeSense uses a fixed latent-chain length at inference time. Allowing the reasoner to select its length according to different tasks could help balance reasoning cost and action gains, but we have not trained or evaluated such a mechanism. Our approach also depends on future dynamics as a supervision signal; the effects of target quality, the future-observation horizon, and the choice of geometric targets remain to be studied. Finally, the first three training stages do not require action annotations, making large-scale video pretraining a natural extension of the framework. Whether such pretraining improves the learned intuition and its transfer to downstream control has yet to be established.

In conclusion, ForeSense presents a new perspective on VLA reasoning by distilling future dynamics into a current-only latent reasoner, directly guiding action generation with its internal representations. Comprehensive evaluation across LIBERO, LIBERO-Plus, and real-world manipulation tasks demonstrates that learning an unconstrained ”latent visual intuition” provides substantial gains, especially under domain and language perturbations. More broadly, by showing that visual intuition can be learned independently of action labels, this work provides key insights into how future physical reasoners might be pretrained directly from large-scale internet video datasets.

AI USE STATEMENT

Generative AI tools were used to assist with translation, drafting and editing parts of the manuscript, improving readability, brainstorming presentation choices, discovering and summarizing relevant literature, and formatting references. They were also used to assist with data annotation, including generating semantic descriptions used as training targets, and to suggest edits to software code. All AI-assisted outputs were critically reviewed by the authors. References were verified against their original sources, and suggested code changes were manually inspected and tested. The authors take full responsibility for the final content of this work.

REFERENCES

Mahmoud Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report, 2025.

Shuanghao Bai, Jing Lyu, Wanqi Zhou, Zhe Li, Dakai Wang, Lei Xing, Xiaoguang Zhao, Pengwei Wang, Zhongyuan Wang, Cheng Chi, et al. Latent reasoning VLA: Latent thinking and prediction for vision-language-action models. In *Forty-third International Conference on Machine Learning*, 2026.

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

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, 2024.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proceedings of The 9th Conference on Robot Learning*, pp. 17–40, 2025.

Christopher Clark, Yue Yang, Jae Sung Park, Zixian Ma, Jieyu Zhang, Rohun Tripathi, Mohammadreza Salehi, Sangho Lee, Taira Anderson, Winson Han, et al. Molmopoint: Better pointing for vlms with grounding tokens. *arXiv preprint arXiv:2603.28069*, 2026.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 38574–38583, 2026.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Conference on Language Modeling*, 2025.

Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Frank Wang, and Fu-En Yang. Thinkact: Vision-language-action reasoning via reinforced visual latent planning. In *Advances in Neural Information Processing Systems*, pp. 82782–82802, 2025.

Chi-Pin Huang, Yunze Man, Zhiding Yu, Min-Hung Chen, Jan Kautz, Yu-Chiang Frank Wang, and Fu-En Yang. Fast-thinkact: Efficient vision-language-action reasoning via verbalizable latent planning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5070–5081, 2026.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. In *Proceedings of Robotics: Science and Systems*, 2025a.

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*, pp. 2679–2713, 2025b.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. In *International Conference on Learning Representations*, pp. 71531–71552, 2026.

Myungkyu Koo, Daewon Choi, Taeyoung Kim, Kyungmin Lee, Changyeon Kim, Younggyo Seo, and Jinwoo Shin. Hamlet: Switch your vision-language-action model into a history-aware policy. In *International Conference on Learning Representations*, pp. 101537–101558, 2026.

Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, et al. Molmoact: Action reasoning models that can reason in space, 2025.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow Matching for Generative Modeling. In *International Conference on Learning Representations*, 2023.

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*, pp. 44776–44791, 2023.

Zhuoyang Liu, Jiaming Liu, Hao Chen, Jiale Yu, Ziyu Guo, Chengkai Hou, Chenyang Gu, Xiangju Mi, Renrui Zhang, Kun Wu, et al. LaST$_{0}$: Latent spatio-temporal chain-of-thought for robotic vision-language-action model. In *Forty-third International Conference on Machine Learning*, 2026.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, 2023.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient Action Tokenization for Vision-Language-Action Models. In *Proceedings of Robotics: Science and Systems*, 2025.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Jiayuan Gu, Zhigang Wang, Yan Ding, Bin Zhao, Dong Wang, et al. SpatialVLA: Exploring Spatial Representations for Visual-Language-Action Models. In *Proceedings of Robotics: Science and Systems*, 2025.

Sandeep Kumar Routray, Hengkai Pan, Unnat Jain, Shikhar Bahl, and Deepak Pathak. Vipra: Video prediction for robot actions. In *International Conference on Learning Representations*, pp. 68823–68857, 2026.

Yichao Shen, Fangyun Wei, Zhiying Du, Yaobo Liang, Yan Lu, Jiaolong Yang, Nanning Zheng, and Baining Guo. Videovla: Video generators can be generalizable robot manipulators. In *Advances in Neural Information Processing Systems*, pp. 95597–95621, 2025.

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. Vla-jepa: Enhancing vision-language-action model with latent world model, 2026.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

Fuxiang Yang, Donglin Di, Lulu Tang, Xuancheng Zhang, Lei Fan, Hao Li, Wei Chen, Tonghua Su, and Baorui Ma. Chain of world: World model thinking in latent motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6675–6684, 2026.

Jinhui Ye, Fangjing Wang, Ning Gao, Junqiu Yu, Yangkun Zhu, Bin Wang, Jinyu Zhang, Weiyang Jin, Yanwei Fu, Feng Zheng, et al. Spatially guided training for vision-language-action model. In *International Conference on Learning Representations*, pp. 32395–32434, 2026.

Cheng Yin, Yankai Lin, Wang Xu, Sikyuen Tam, Xiangrui Zeng, Zhiyuan Liu, and Zhouping Yin. Deepthinkvla: Enhancing reasoning capability of vision-language-action models, 2025.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In *Proceedings of The 8th Conference on Robot Learning*, pp. 3157–3181, 2025.

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 (ICCV)*, pp. 11975–11986, 2023.

Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, XinQiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, He Wang, Zhizheng Zhang, et al. Dreamvla: A vision-language-action model dreamed with comprehensive world knowledge. In *Advances in Neural Information Processing Systems*, pp. 24195–24228, 2025.

Zhuoyang Zhang, Shang Yang, Qinghao Hu, Luke J. Huang, James Hou, Yufei Sun, Yao Lu, and Song Han. Foreact: Steering your vla with efficient visual foresight planning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 37195–37205, 2026.

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

Linqing Zhong, Yi Liu, Yifei Wei, Ziyu Xiong, Si Liu, and Guanghui Ren. Acot-vla: Action chain-of-thought for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8152–8162, 2026.

Gaoyue Zhou, Hengkai Pan, Yann Lecun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 79115–79135, 2025.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Kevin Lin, Abhiram Maddukuri, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning, 2020.

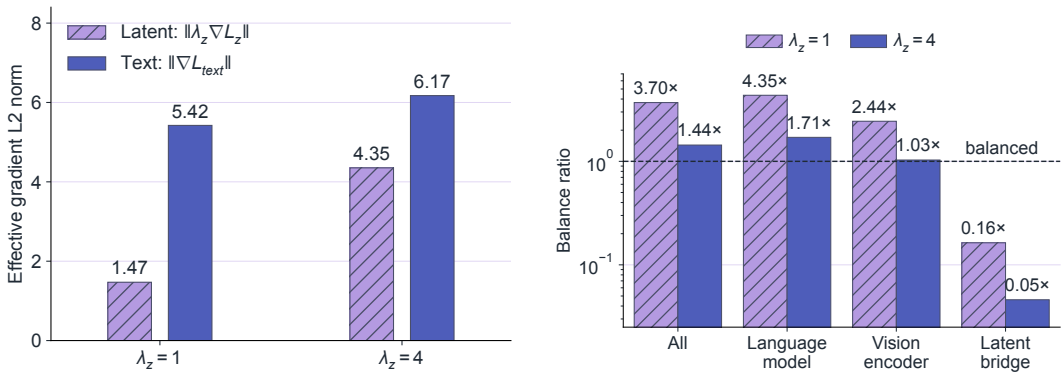
Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of The 7th Conference on Robot Learning*, pp. 2165–2183, 2023.

A Implementation Details

A.1 Reasoner Training

We initialize the reasoner from Qwen3-VL-4B (Bai et al., 2025). Stage 1 trains it to predict the future-dynamics target from the instruction and current and future observations. Stage 2 introduces a bias-free linear bridge $B_\phi : \mathbb{R}^{d_{\text{model}}} \to \mathbb{R}^{d_{\text{model}}}$ that maps the preceding hidden state to the next latent input embedding, where $d_{\text{model}} = 2560$ for Qwen3-VL-4B. Thus, the latent embedding dimension is $d_z = d_{\text{model}} = 2560$. Target-text positions attend to latent positions and causally available target text, but cannot attend directly to the multimodal prefix.

For Detached Latent Curriculum (DeLaC), we extend the chain in the order $K = 1 \to 2 \to \cdots \to 8$. At each transition, the Qwen backbone and latent bridge are initialized from the final checkpoint of the preceding curriculum step. Earlier latent input embeddings are detached when the new position



(a) Whole-model update magnitude

(b) Distance from equal contribution

Figure 5: **Stage-3 gradient balance analysis.** (a) Effective gradient magnitudes of the latent and text objectives under $\lambda_z = 1$ and $\lambda_z = 4$. (b) Corresponding latent-to-text gradient balance ratios for the whole model and individual modules. The dashed line at $1\times$ denotes equal contribution from the two objectives. The $\lambda_z = 4$ values illustrate the effect of loss weighting rather than measurements from a separate training run.

is added. The backbone remains trainable, so detachment does not freeze its computations at earlier positions.

In Stage 3, the student is initialized from the final $K = 8$ Stage-2 checkpoint, while the teacher is a frozen copy of the same checkpoint. The student receives only the current observation and instruction, whereas the teacher retains access to future observations. The student is optimized with position-wise latent MSE and future-dynamics next-token prediction. At equal loss weights, gradient norms measured over three mini-batches at the final eight-token checkpoint show that the text objective contributes approximately 3.7 times the gradient magnitude of latent alignment over all parameters, and 4.4 times within the language backbone. We therefore set $\lambda_z = 4$ in the main experiments to bring the two contributions to the shared backbone to similar magnitudes (Fig. 5).

Stages 1–3 use eight NVIDIA H200 GPUs and bfloat16 precision. We use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, zero weight decay, and gradient clipping at 1.0. Gradient checkpointing is disabled in the main runs. All batch sizes in the table are DDP global batch sizes per optimizer step.

Stage 1 and Stage 3 are each trained for 430 optimizer steps. Each of the eight Stage-2 curriculum stages ($K = 1, \ldots, 8$) is also trained for 430 optimizer steps and initialized from the final checkpoint of the preceding curriculum stage. For every run, the learning rate linearly warms up to 5×10^{-5} over the first 43 optimizer steps and then follows cosine decay to 5×10^{-6} at step 430. With eight GPUs, Stage 1 and the $K = 1, 2$ Stage-2 runs use a per-GPU micro-batch size of 8 with four gradient-accumulation steps, giving a global batch size of 256. The remaining Stage-2 runs and Stage 3 use a per-GPU micro-batch size of 4 with eight accumulation steps, retaining the same global batch size. Stage 3 trains the eight-token student from the final $K = 8$ Stage-2 checkpoint. Checkpoints are taken at the prescribed final step rather than selected by a validation metric. The latent-length ablation evaluates shorter prefixes of the trained eight-token Stage-3 reasoner and Stage-4 policy without retraining either model.

A.2 Action Policy and Reasoning Fusion

We build the action policy on $\pi_{0.5}$-LIBERO and freeze the current-only reasoner during Stage 4. At each policy call, the reasoner generates eight latent positions. ReAct-Fuse exports only the key–value entries at these positions from Qwen layers 18–35 and pairs them with action-expert layers 0–17, respectively (zero-based indexing). For an action sequence of length N, reasoning-attention query rotary positions are assigned consecutively beginning at the position immediately after the final latent token, while the cached latent keys retain their original rotary positions.

Native VLA attention and reasoning-memory attention use separate softmax operations. Their outputs are combined within the action expert's existing gated attention residual, followed by the native

feed-forward block. The query projection into Qwen's attention space and the projection back to the action hidden dimension are trainable. The RMS normalization applied to the transferred Qwen attention update is non-affine, with $\epsilon = 10^{-6}$. The Stage-3 Qwen backbone and latent bridge remain frozen.

Stage 4 trains all parameters of the $\pi_{0.5}$ policy, including its SigLIP visual encoder (Zhai et al., 2023), native PaliGemma backbone (Beyer et al., 2024), 18-layer action expert, and action input, time, and output projections. All newly introduced ReAct-Fuse parameters are also trained. The policy is trained for 30,000 optimizer steps on eight NVIDIA H200 GPUs, with a global batch size of 256 (32 examples per GPU). We use the same AdamW hyperparameters as in Stages 1–3, including zero weight decay and gradient clipping at 1.0, and train in bfloat16 precision. The learning rate linearly warms up to 5×10^{-5} over 10,000 steps and remains constant thereafter.

The policy predicts an action chunk of length 10. During evaluation, it executes the first five actions and then replans using a new observation. The LIBERO action has seven dimensions and is represented internally with dimension 32. Action generation uses 10 denoising steps.

A.3 Simulation Evaluation

The simulator provides 256×256 main- and wrist-camera images. Images are resized and padded to 224×224 before model input. The evaluation action horizon is 10, with replanning every five control steps.

Standard LIBERO contains four suites of 10 tasks each. We evaluate 50 initial states per task, for 2,000 rollouts in total. The maximum numbers of effective control steps are 220 for Spatial, 280 for Object, 300 for Goal, and 520 for Long (LIBERO-10). Each rollout uses 10 settling steps. The standard-LIBERO main run uses the `legacy_shard` reset procedure.

LIBERO-Plus evaluates one rollout per perturbed task instance using that instance's first initial state, for 10,030 evaluated instances in total. Policies are trained on standard LIBERO without perturbation-specific action training. Overall success pools successful rollouts across evaluated instances:

$$\mathrm{SR} = 100 \frac{\sum_q s_q}{\sum_q n_q}, \tag{14}$$

where s_q and n_q are the numbers of successes and evaluated rollouts in group q. Thus, overall success is not necessarily the unweighted mean of category success rates.

We use evaluation seed 7 for NumPy, the environment, and task selection. The policy server used for these runs did not separately fix Torch diffusion noise for each request; individual action outputs are therefore not guaranteed to be identical across repeated evaluations.

On standard LIBERO, we use the officially reported result for the released $\pi_{0.5}$-LIBERO checkpoint. On LIBERO-Plus, we evaluate that checkpoint on the same task instances as ForeSense. Results for other external methods are taken from their respective papers or official project pages.

Simulation environment. We evaluate on LIBERO-Plus v0.1.0, using the benchmark implementation from `sylvestf/LIBERO-plus`. The simulation stack comprises robosuite 1.4.0 (Zhu et al., 2020), MuJoCo 3.4.0 (Todorov et al., 2012), and BDDL 1.0.1. Evaluation runs on Ubuntu 24.04.2 LTS with Python 3.12.3, NVIDIA driver 570.124.06, CUDA 12.8, and EGL off-screen rendering.

B Future-Dynamics Data Pipeline

B.1 Dataset and Segment Sampling

The underlying LIBERO demonstrations contain 1,693 trajectories and 273,465 frame-level action-training examples. We construct 22,186 annotated segments for future-dynamics supervision. Table 5 shows their distribution across suites.

Each annotated segment contains a current observation and seven future observations. We sample eight frames at offsets $[0, 5, 10, 15, 20, 25, 30, 35]$ control steps. At 20 Hz, these correspond

Table 5: **Demonstrations and future-dynamics annotations.** Stages 1–3 use annotated segments; Stage 4 uses frame-level action-training examples.

Suite	Trajectories	Frame-level examples	Annotated segments
Spatial	432	52,970	3,976
Object	454	66,984	5,311
Goal	428	52,042	3,905
LIBERO-10	379	101,469	8,994
Total	1,693	273,465	22,186

to $[0, 0.25, 0.50, 0.75, 1.00, 1.25, 1.50, 1.75]$ seconds. Starting points are sampled every 10 control steps (0.5 seconds). With $H = 7$ and $\Delta = 5$, the future sequence is

$$o_t^+ = \{o_{t+j\Delta}\}_{j=1}^{H}. \tag{15}$$

The latent-chain length is independent of the number of frames; latent positions do not correspond to individual future frames.

Only complete windows enter the annotation dataset. All 22,186 records contain eight frames; starting points with insufficient future observations are excluded. Stage 4 uses all 273,465 frame-level action-training examples. Checkpoints are selected by the prescribed training schedule, not by a validation metric.

B.2 Semantic Annotation

We provide the eight-frame clip to GPT-6 Astra for a description of the observed robot–object interaction. The prompt persisted with every annotation record is:

```
describe what happens in this video clip
```

The requested and returned model identifiers are both `openai/gpt-6-astra`. The final annotation package contains 22,186 unique requests and one persisted prompt version. It does not retain an additional system prompt or generation settings such as temperature and sampling seed; we therefore do not assign values to those fields.

Every final request has a corresponding annotation record. In the final package, the counts of target-parsing failures, missing metadata, empty observations, unknown suites, and requested/returned model mismatches are all zero. The training loader rejects malformed records, empty observations, and unknown suites. A target longer than 544 Qwen tokens raises an error rather than being silently truncated.

B.3 Geometric Annotation

For simulation data, we project six MuJoCo gripper-body reference points into each 256×256 main-camera frame. Their identities are listed in Table 6.

Table 6: **Six gripper reference points** used for simulation-derived geometric targets.

Point	MuJoCo body
B12	`gripper0_right_gripper`
B13	`gripper0_eef`
B14	`gripper0_leftfinger`
B15	`gripper0_finger_joint1_tip`
B16	`gripper0_rightfinger`
B17	`gripper0_finger_joint2_tip`

Each trajectory has shape $[6, 8, 2]$: six points, eight frames, and two image-plane coordinates. The image origin is at the upper left, with u increasing rightward and v increasing downward. Floating-point projected coordinates and validity flags are retained in the annotation data. For the textual target, coordinates are rounded half up and clipped to $[0, 255]$. An invalid projection is encoded as `[-1,-1]` and does not cause the segment to be discarded.

The target serializes the semantic description inside `<DESC>` tags and the six point trajectories inside `<TRAJ>` tags. For real-world videos, MolmoPoint-8B (Clark et al., 2026) supplies tracked gripper points instead of simulator-derived projections.

All gripper-point trajectories are annotated in the third-person camera view. For real-world segments, we run the official MolmoPoint-8B checkpoint with the prompt "track the grippers" to obtain two point tracks. We manually verify the tracks to identify tracking failures before using them as supervision.

B.4 A Future-Dynamics Target Example

Figure 6 shows the eight main-camera frames for sample `ep000098_t000090`: the current observation and seven future observations at $[0, 0.25, 0.50, 0.75, 1.00, 1.25, 1.50, 1.75]$ seconds. Each frame has a resolution of 256×256 pixels. The target contains a semantic description and image-plane trajectories for six projected gripper points. Each point has one (u, v) coordinate per frame, giving an array of shape $[6, 8, 2]$. All 48 projected points are valid in this example.

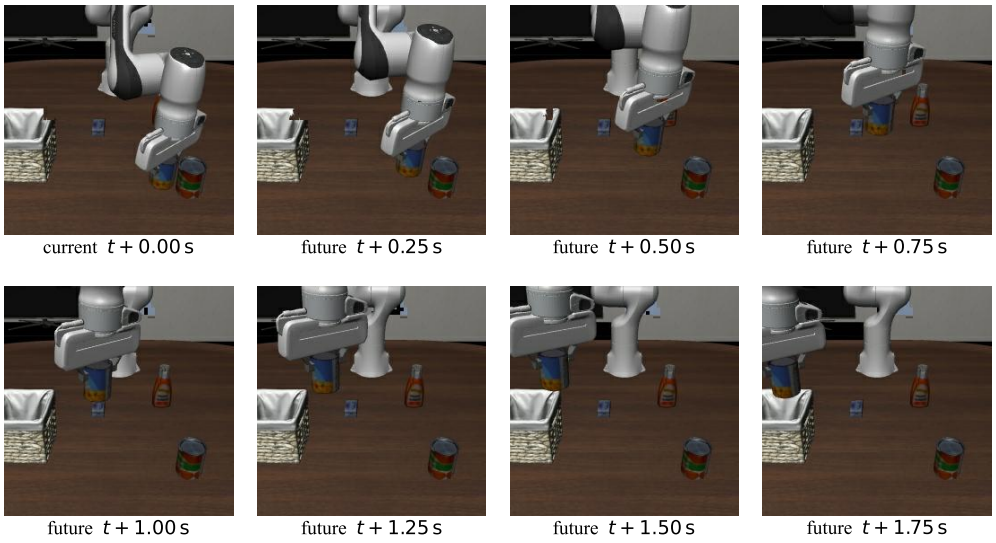


Figure 6: **Input frames for a future-dynamics target example.** The first frame is the current observation; the remaining seven are sampled future observations. Frames are shown without geometric overlays.

The complete target is shown below. Line breaks within its description and trajectories are added for readability.

```
<DESC>
A robotic arm grips a blue-and-yellow can, lifts it from the
wooden table, and moves it left toward an open woven basket.
The clip ends with the can held above the basket's opening.
A second can, a small rectangular object, and a sauce bottle
remain on the table.
</DESC>
<TRAJ>
B12:[[187,140],[181,122],[163,93],[137,67],
     [106,48],[76,36],[46,33],[18,44]]
B13:[[180,176],[174,160],[157,135],[133,111],
```

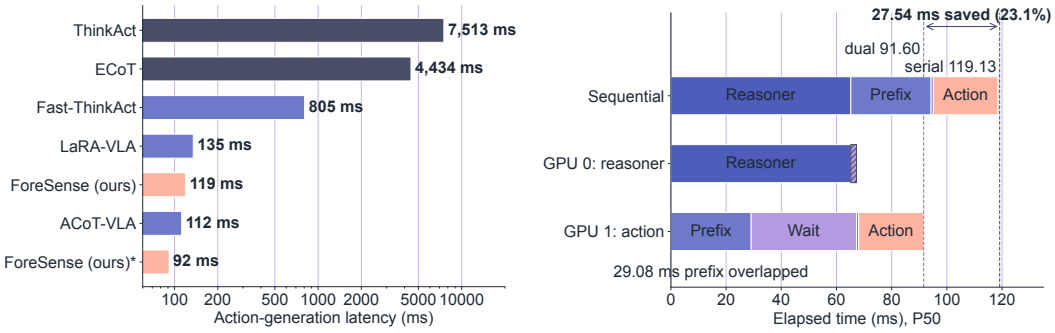
```
     [105,95],[77,85],[49,82],[25,91]]
B14:[[191,155],[186,138],[170,112],[146,88],
     [118,71],[89,61],[62,58],[36,68]]
B15:[[189,173],[184,158],[169,133],[146,111],
     [120,95],[93,86],[66,83],[43,91]]
B16:[[172,166],[166,149],[148,122],[121,97],
     [91,78],[60,67],[31,64],[5,74]]
B17:[[166,188],[160,172],[142,147],[116,123],
     [87,106],[58,95],[29,92],[5,101]]
</TRAJ>
```

Here `B12`–`B17` identify the six gripper points. Coordinates follow the frame order shown in Figure 6.

B.5 Inference Efficiency

We measure the end-to-end latency of generating one action chunk. ForeSense takes 119.13 ms on a single NVIDIA H200 and 91.60 ms with two H200 GPUs using parallel inference, a 1.30× speedup. Figure 7(a) places these measurements alongside latency figures reported for other reasoning-based VLAs to illustrate the range of inference costs. The prior-work figures use their respective hardware and timing protocols, so they do not support direct speedup comparisons.

Figure 7(b) explains the speedup within our own setup at $K = 8$. On one GPU, reasoner generation precedes native VLA prefix computation and action denoising. With two GPUs, the native prefix computation overlaps with reasoner generation; latent KV transfer and action denoising follow. This overlaps 29.08 ms of prefix computation with the reasoner and reduces median end-to-end latency by 27.53 ms (23.1%). The gain comes from overlapping independent computations while generating the same eight latent states.



(a) Cross-method latency **(b) Parallel inference timeline**

Figure 7: **Inference latency and parallel execution.** (a) End-to-end latency per action chunk. ForeSense is measured on one H200; ForeSense * denotes parallel inference on two H200s. Other values are reported by the respective papers under different hardware and inference settings; the axis is logarithmic. (b) Median (P50) timeline at $K = 8$, showing native-prefix overlap with reasoner generation on separate GPUs, followed by latent KV transfer and action denoising.

C Real-World Experiments

We evaluate OpenVLA (Kim et al., 2025b), $\pi_{0.5}$ (Black et al., 2025), π_0 (Black et al., 2024), and ForeSense on six real-world manipulation tasks. Each method is tested in 20 trials per task, and the success rates summarized in Fig. 4 are calculated from the resulting trial outcomes. The tasks cover fine-grained and long-horizon tasks, multiple action types (pick-and-place and pushing), deformable-object manipulation, and bimanual coordination.

Robot Platform. All experiments use a Galaxea R1 Lite platform equipped with two 6-DoF arms and parallel grippers. Policy inference runs on an NVIDIA GeForce RTX 5090 GPU. Given the

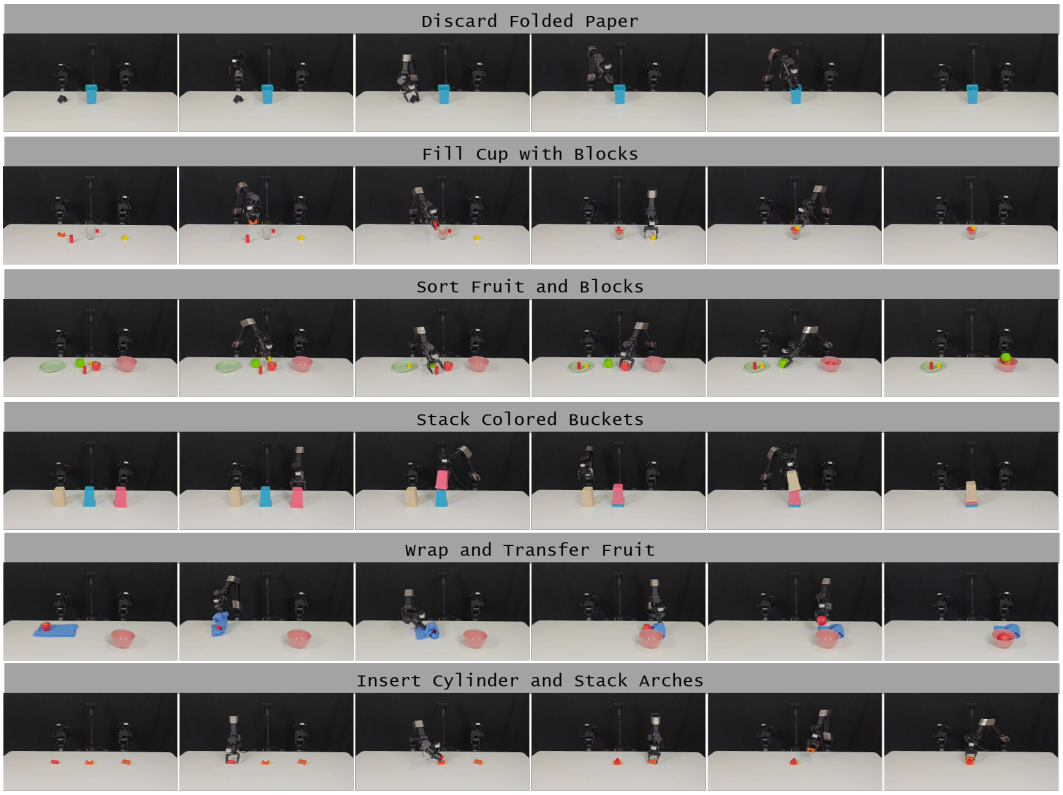


Figure 8: Overview of the six real-world manipulation tasks used in our experiments. The evaluation spans fine-grained and long-horizon tasks, multiple action types (pick-and-place and pushing), deformable-object manipulation, and bimanual coordination.

current visual observation and task instruction, the policy predicts a chunk of future actions, which is passed to the robot controller for execution. For ForeSense, the latent reasoner uses the current observation and instruction; future observations are not provided during deployment.

Demonstration Collection. We collect 80 expert demonstrations for each task, giving 480 real-world trajectories in total, corresponding to approximately 4–5 hours of robot interaction data. Of the 80 demonstrations per task, 60 use the nominal task setup and 20 introduce variations in object placement or scene arrangement. The collected trajectories cover the complete task behavior, including intermediate interactions when a task requires multiple placements or an arm-to-arm transfer. In addition to providing the real-world action data used for policy training, these demonstrations contain temporally synchronized observations and robot actions, which are also used to support consistent observation–action alignment during real-world deployment.

Training and Annotation. We use the same four-stage annotation and training procedure as in simulation. Semantic future-dynamics descriptions are generated from recorded observations in the same way. For geometric targets, we obtain two gripper-point tracks with MolmoPoint-8B instead of projecting six simulator reference points. The reasoner is trained with these future-dynamics targets and frozen when coupled to the real-world action policy. OpenVLA, π_0, $\pi_{0.5}$, and ForeSense use the same demonstrations, camera observations, and action-policy training settings. Each method retains its own model architecture and corresponding initialization.

Evaluation Protocol. We conduct 20 physical trials for every method and task. Initial configurations vary across trials, including changes to object positions and scene arrangements. For tasks with multiple required steps, we count a trial as successful only when the full task is completed; reaching an intermediate state alone does not count. The task-specific criteria are listed in Table 7; representative setups are shown in Fig. 8.

Table 7: **Real-world task definitions.** Each method is evaluated over 20 trials per task.

Task	Required behavior	Success criterion
T1	Place folded paper into a trash bin.	The paper is inside the bin at the end of the trial.
T2	Place blocks into a cup through repeated placements.	All required blocks are inside the cup at the end of the trial.
T3	Place a fruit in a bowl and a block on a plate.	The fruit is in the bowl and the block is on the plate.
T4	Stack a red bucket on a blue bucket, then a yellow bucket on the red bucket.	The buckets form the specified stack and remain in place at the end of the trial.
T5	Wrap and transfer a fruit with a towel using one arm; uncover it and place it into a basket using the other.	The transfer and uncovering steps are completed, and the fruit is in the basket at the end of the trial.
T6	Place a red cylinder inside an arch-shaped block and stack a second arch-shaped block on top.	The cylinder is inside the first arch, and the second arch remains stacked on top at the end of the trial.

D Additional Analysis

D.1 Evolution of the Latent Chain

Figure 9 projects the eight latent states of four example instructions onto the first two principal components, which explain 31.5% and 20.2% of the variance. Each instruction follows a distinct trajectory. The large initial shifts suggest that early latent states establish a representation of the anticipated interaction, while the smaller subsequent changes suggest that later states refine it. This coarse-to-fine pattern is consistent with the the main-text results: extending the chain improves modeling of the supervised future-dynamics targets and downstream success, and DeLaC yields more diverse latent representations than direct training.

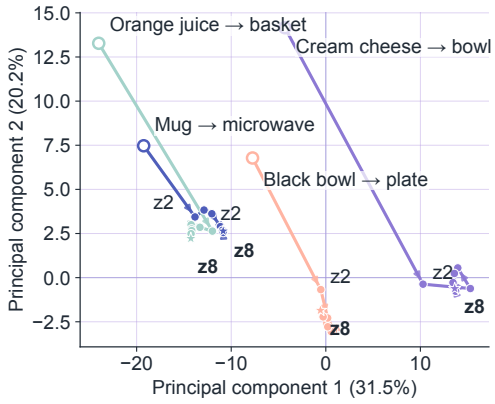


Figure 9: **Latent trajectories for four instructions.** PCA projection of $z_1, \ldots, z_8$ in generation order. The first two components explain 31.5% and 20.2% of the variance. Projected distances do not directly measure reasoning quality.

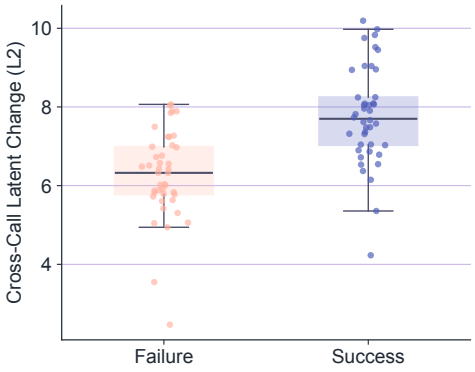


Figure 10: **Cross-call variation of latent states.** Euclidean distance between corresponding latent positions in consecutive policy calls, grouped by episode outcome. Boxes show interquartile ranges; lines show medians.

D.2 Latent Variation During Closed-Loop Execution

At each policy call, the reasoner generates a new latent chain from the latest observation. Figure 10 compares the change in corresponding latent states between consecutive calls. Successful rollouts show a higher median change than failed rollouts, suggesting that the reasoner more actively updates its internal representation as successful interactions unfold. This behavior complements the within-

chain trajectories in Figure 9: latent states evolve both while the chain is generated and as the policy receives new visual evidence.

D.3 Inference Cost of the Latent Chain

Figure 11 reports inference latency as the chain length varies from one to eight latent states. Reasoner latency increases with K, from approximately 45 ms at $K = 1$ to 73 ms at $K = 8$. At $K = 8$, complete action-chunk latency is approximately 119 ms on one GPU and 92 ms with the reasoner and action model placed on separate GPUs. Thus, the full eight-state chain used in our main experiments has a measurable cost, while parallel execution reduces its effect on end-to-end latency.

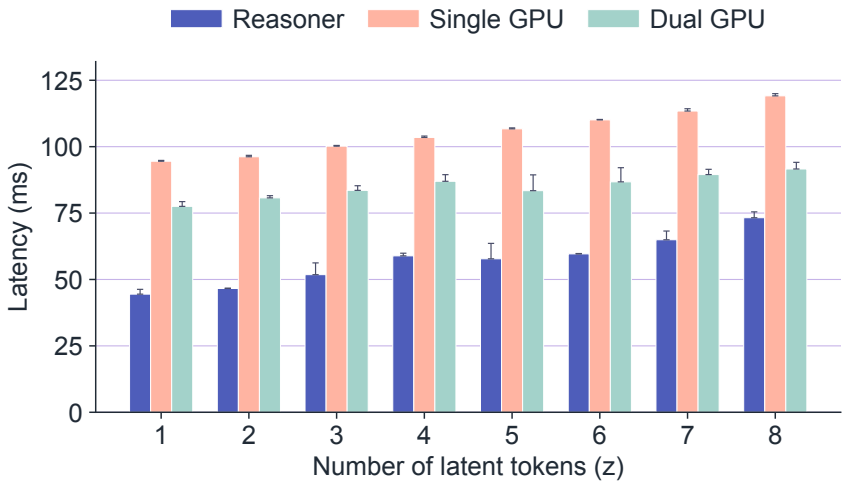


Figure 11: **Latency versus latent-chain length.** Median (P50) reasoner latency and end-to-end action-chunk latency on one or two GPUs for $K = 1, \ldots, 8$. Upper whiskers indicate P90. The dual-GPU configuration executes reasoner and action-model computations in parallel where their dependencies permit.

D.4 Parallel Inference Breakdown

Figure 7 shows where the two-GPU speedup arises at $K = 8$. In sequential execution, the reasoner runs before the native VLA prefix computation and action denoising. With two GPUs, the native prefix computation overlaps with reasoner generation; the action model then uses the exported latent memory for denoising. The profile shows 29.08 ms of prefix computation hidden under reasoner execution. Median end-to-end latency decreases from 119.13 ms to 91.60 ms, saving 27.53 ms (23.1%) per action chunk. This is a scheduling gain from overlapping independent computations, rather than a reduction in the number of generated latent states.