CONTINUOUS REASONING FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

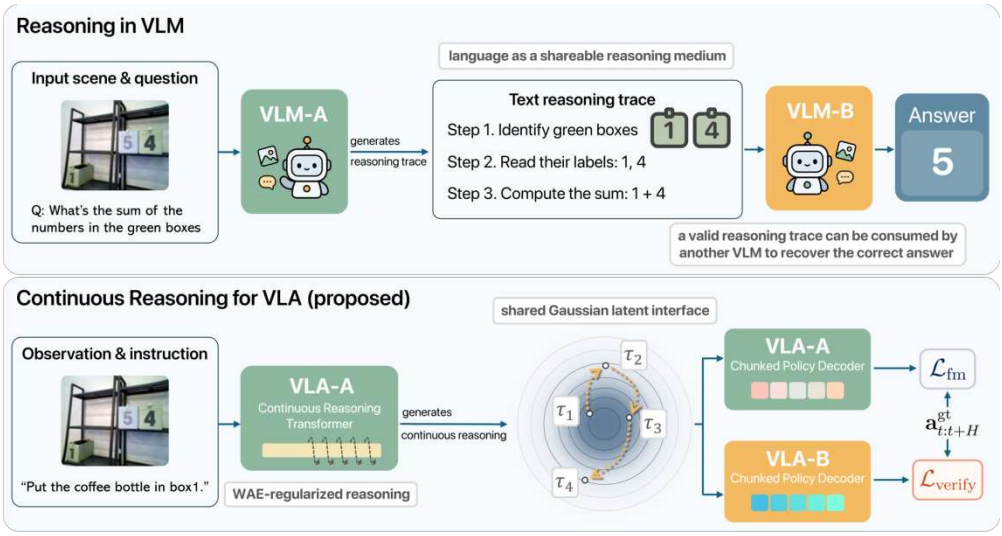


Figure 1: Continuous Reasoning overview. Top: a textual reasoning trace is a shared medium; a trace written by one VLM can be read by another. Bottom: CR trains a continuous interface through a Gaussian-prior bottleneck and action supervision from both the student (VLA-A) and its EMA consumer (VLA-B). The consumer provides self-verification during training; the student predicts actions at inference. Thought vectors are shown schematically.

ABSTRACT

A robot's intermediate thoughts can be trained through the actions they enable, and a useful thought should serve more than the network that produced it. We introduce *Continuous Reasoning* (CR), a vision-language-action (VLA) framework that learns a sequence of continuous thought vectors directly from action supervision, without annotated reasoning traces. Our key idea is *self-verification*: a thought should support action prediction by both the policy that generates it and an evolving exponential-moving-average (EMA) consumer. The consumer's action-prediction error backpropagates to the code generator, training thoughts through their use for action prediction. Built on $\pi_{0.5}$, CR generates thoughts sequentially through a Gaussian-prior bottleneck and uses the resulting prefix to condition block-causal action prediction. The EMA consumer is used only during training, so inference requires no separate verification pass. On LIBERO-PRO, CR improves mean success from 58.0% to 64.0% over its backbone and exceeds DeepThinkVLA-RL in all four suite means, with a 14.0-percentage-point overall advantage. Gains concentrate on position and task perturbations, where policies must adapt to changed layouts and goals. On TX-G2 and HSR, mean subtask success improves by 19.8 and 15.6 percentage points, respectively, with gains also extending to full-task completion. Every component ablation loses on those same two families, and freezing the consumer removes the benefit of self-verification entirely. These results establish action-supervised continuous reasoning as an effective interface for robust VLA control.

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.

1 INTRODUCTION

Vision-language-action (VLA) models translate perception and language into robot actions (Brohan et al., 2022; 2023; Driess et al., 2023; Octo Model Team et al., 2024; Kim et al., 2024; Black et al., 2025), and a growing line of work inserts a reasoning step before acting: textual chains of thought, visual subgoals, language plans, or latent planning states (Ahn et al., 2022; Huang et al., 2022; Liang et al., 2022; Huang et al., 2023; Zhao et al., 2025; Shou et al., 2026; Zhong et al., 2026b; Huang et al., 2025; 2026; Bai et al., 2026; Zhong et al., 2026a; Ling et al., 2026). These approaches typically supervise the intermediate step with signals other than the robot's actions, such as annotated traces, subgoal images, or plans. Such supervision is costly to collect, and it optimizes the intermediate step for a proxy target rather than for the control decision it must serve. We take a different position: *for a robot, the value of a thought lies in the actions it enables, and a thought that only its own decoder can read is a shortcut rather than a representation.* This paper asks whether a VLA can learn its intermediate reasoning from action supervision alone, with no reasoning annotations at all.

We answer with *Continuous Reasoning* (CR). Before decoding actions, a $\pi_{0.5}$ policy (Physical Intelligence et al., 2025) generates a short sequence of continuous thought vectors; each is computed from the observation, the instruction, and the thoughts already written, so later thoughts build on earlier ones. A Wasserstein autoencoder (Tolstikhin et al., 2017) passes each thought through a Gaussian-prior bottleneck, and the decoded sequence conditions block-causal action prediction over the horizon. The thoughts receive no annotation; they are trained through action prediction. Training such a channel with its own decoder alone, however, invites co-adaptation: the decoder can learn to read whatever the generator happens to emit, and the code need not carry anything that a different consumer could act on. Our central mechanism, *self-verification*, turns this into a training criterion: the policy verifies its own thoughts by requiring that a slower-moving copy of itself can act on them. Concretely, an exponential-moving-average (EMA) copy of the policy (Tarvainen & Valpola, 2017) decodes the same code through its own decoder and must predict the same actions; the gradient of that consumer's action loss trains the code generator. Because the EMA consumer evolves throughout training, the generator is supervised by a slowly changing sequence of action predictors in addition to its own decoder. The consumer exists only during training: at deployment, CR generates its thoughts and actions in one policy query, with no verification stage and no text generation. We use *continuous reasoning* to refer to this learned sequence of intermediate computations, generated before acting and trained only through the actions it enables.

The evidence supports this position and reveals a consistent pattern. On LIBERO-PRO, CR raises mean success over its backbone from 58.0% to 64.0%, and the gains concentrate where a policy must re-anchor its behavior: success under *position* perturbations rises from 26.8% to 39.3% and under *task* perturbations from 24.8% to 37.1%, while object and semantic scores change little. Against DeepThinkVLA-RL (Yin et al., 2025), an explicit text-reasoning policy, CR attains a higher mean in every suite, 64.0% versus 49.9% overall, with the widest margin again under position changes. Removing any component of CR lowers success on precisely these two perturbation families and leaves the others comparatively stable, and freezing the EMA consumer removes the benefit of self-verification entirely: the gain comes from a reader that keeps evolving. On two physical robots, mean subtask success improves by 19.8 percentage points on the bimanual TX-G2 and by 15.6 on the mobile HSR, and full-task completion moves from 0/10 to 4/10 on bowl stacking and from 0/6 to 4/6 on a two-bottle mobile task.

Our contributions are threefold:

- **Reasoning learned from actions.** We introduce CR, a VLA framework that generates a sequence of continuous thoughts before acting and learns this interface entirely from action supervision, without annotated reasoning traces.
- **Self-verification: thoughts that serve a second, evolving reader.** We train thoughts through their use by a second, EMA-updated policy whose action-prediction gradients flow back to the generator, with Gaussian-prior regularization and block-causal decoding structuring the interface and its use.
- **Robustness and physical task execution.** CR improves LIBERO-PRO robustness over its backbone and over an explicit text-reasoning policy, with component ablations tracing the gains to the perturbations that demand re-anchoring, and it improves subtask and full-task completion on two physical robot platforms.

2 Related Work

Reasoning in VLA. Recent VLA research inserts an intermediate reasoning step before action generation, and existing designs differ mainly in what that step is and what supervises it. Explicit approaches produce textual or multimodal chain-of-thought traces, learned from annotated traces or future frames, and condition the policy on the decoded trace (Zhao et al., 2025; Shou et al., 2026; Zhong et al., 2026b; Wen et al., 2025; Yin et al., 2025). Latent approaches replace the trace with compact planning states, but these states are still trained toward targets of their own, such as visual planning rewards, verbalizable plans, or predicted future observations (Huang et al., 2025; 2026; Bai et al., 2026; Liu et al., 2026); a related line grounds reasoning in action-level or spatial abstractions and external guidance (Zhong et al., 2026a; Lee et al., 2025; Ling et al., 2026). Across these designs, the intermediate step typically has an objective separate from the actions it precedes; ThinkAct, for instance, trains a separate reasoning model with action-aligned visual rewards and conditions an action model on its plan latent. CR removes that separation: its thoughts have no target of their own and are trained only through the action prediction they enable, by the student and by an evolving EMA consumer, within one policy query and without a separate reasoning model. Instruction-oriented systems that preserve broader pretrained reasoning inside embodied models are complementary (Driess et al., 2023; Zhou et al., 2025b; Yang et al., 2025; Qu et al., 2025). Our experiments compare CR with its backbone, component ablations, and DeepThinkVLA-RL as an external text-reasoning baseline.

Backbones, action generation, and planning. General-purpose VLA models establish the large-scale embodied policy-learning setting in which we operate (Brohan et al., 2022; 2023; Driess et al., 2023; Octo Model Team et al., 2024; Kim et al., 2024; 2025; Black et al., 2025; Physical Intelligence et al., 2025). In parallel, robot policy learning has developed structured action-generation and planning paradigms, including chunked action prediction, behavior tokenization, diffusion-based control, language-guided decomposition, and latent subgoal structure (Zhao et al., 2023; Shafiullah et al., 2022; Lee et al., 2024; Chi et al., 2023; Hou et al., 2024; Haldar et al., 2024; Ahn et al., 2022; Huang et al., 2022; Liang et al., 2022; Huang et al., 2023; 2024; Liu et al., 2024; Chen et al., 2025; Lynch et al., 2019). Our focus is not on scaling or decoding efficiency, but on what kind of reasoning interface should sit on top of a modern generative action policy.

Self-verification and latent structuring. Exponential-moving-average teachers are widely used to stabilize targets and encourage consistency across views or networks (Tarvainen & Valpola, 2017; Grill et al., 2020; Caron et al., 2021), and Wasserstein autoencoders impose an organized latent geometry (Tolstikhin et al., 2017). CR uses both differently: the EMA copy is not a target provider but a consumer whose action-prediction gradients flow back to the student's code generator, and WAE regularization constrains the aggregate distribution of the codes that both consumers read.

3 Continuous Reasoning

CR augments a $\pi_{0.5}$ flow-matching policy with a continuous intermediate interface. At each policy query, the model generates thought vectors, maps them through a Gaussian-prior bottleneck, and uses the decoded vectors as context for action prediction. The interface is computed once for the current prediction horizon and refreshed at the next query. Figure 2 summarizes the student and EMA training paths.

3.1 Prediction horizon and continuous thoughts

Given observations o and instruction x, the policy predicts an action sequence $a \in \mathbb{R}^{H \times d}$, where H is the prediction horizon and d is the action dimension. We partition this horizon into K action blocks of length C, with $H = KC$. The reported default is $H = 16$, $C = 4$, and $K = 4$: one policy query predicts 16 action steps, arranged into four attention blocks. Thus H corresponds to the full action chunk in common action-chunking terminology; C denotes an internal subdivision for the attention mask. It does not specify how many actions the robot executes before the next query.

We represent the intermediate computation by N_τ thought vectors $\tau = [\tau_1, \ldots, \tau_{N_\tau}]$, with $\tau_i \in \mathbb{R}^D$. The thought count is separate from the number of action blocks; the reported configuration supports

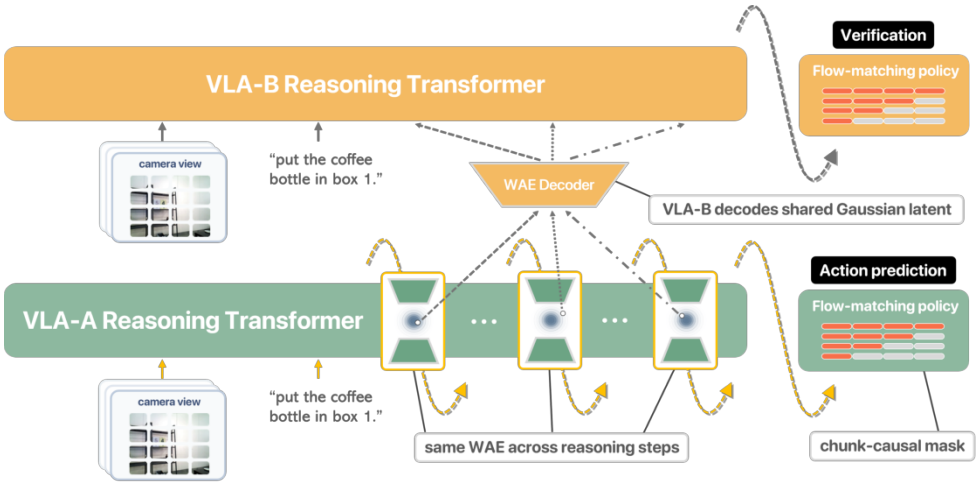


Figure 2: Continuous Reasoning architecture. VLA-A generates thought vectors, encodes them into Gaussian-regularized latent codes, and decodes them for block-causal action prediction. VLA-B is the EMA consumer: an EMA copy of VLA-A with its own EMA decoder. It receives the same latent codes and provides auxiliary action supervision during training.

Algorithm 1 Continuous reasoning at one policy query

Input Current observations o, instruction x, and active thought count N_τ.
Construct the observation–instruction prefix with ordered thought slots.
For each slot $i = 1, \ldots, N_\tau$:
  Compute τ_i at the current slot, attending to the context and previously written thoughts.
  Encode $z_i = q_\phi(\tau_i)$ and decode $\hat{\tau}_i^S = p_\psi(z_i)$.
  Replace the current slot embedding with $\hat{\tau}_i^S$ for subsequent computation.
Condition block-causal flow-matching action generation on the completed prefix.
Output Predicted action horizon $a \in \mathbb{R}^{H \times d}$.

two slots. Slots are generated sequentially. At each slot, the transformer reads the observation–instruction context and the decoded thoughts already written into the prefix. Its hidden state becomes the next thought, which is encoded, decoded, and written back at that slot before the next is generated. Thus, later thoughts can build on earlier intermediate computations. Algorithm 1 summarizes this forward process. The resulting interface conditions the whole prediction horizon and is regenerated from the current inputs at the next policy query. There is no one-to-one assignment between thought slots and action blocks.

3.2 Gaussian-regularized latent interface

A WAE (Tolstikhin et al., 2017) encodes the thoughts as $z = q_\phi(\tau)$ and reconstructs the student interface as $\hat{\tau}^S = p_\psi(z)$. The objective combines reconstruction with matching the aggregate latent distribution to a Gaussian prior:

$$\mathcal{L}_{\text{wae}} = \lambda_{\text{rec}} \|\tau - \hat{\tau}^S\|_2^2 + \lambda_{\text{mmd}} \text{MMD}(z, \mathcal{N}(0, I)). \tag{1}$$

We use an inverse-multiquadratic kernel for MMD; Appendix A describes its distributed computation. Prior matching regularizes the aggregate latent distribution; the code z is the quantity shared by the student and the EMA consumer.

3.3 Block-causal flow-matching action prediction

Given a target action sequence a, noise $\epsilon \sim \mathcal{N}(0, I)$, and time $t \in (0, 1]$, we form $x_t = (1-t)a + t\epsilon$ with target field $u_t = \epsilon - a$, following the flow-matching backbone (Physical Intelligence et al.,

2025). The student predicts the field conditioned on its decoded interface:

$$\mathcal{L}_{\text{fm}} = \mathbb{E}\big[\|v_\theta(o, x, x_t, t; \hat{\tau}^S) - u_t\|_2^2\big]\,. \tag{2}$$

Within each action block, tokens attend bidirectionally. Across blocks, later tokens can attend to earlier blocks but not conversely. This is a block-causal attention pattern within a flow-matching decoder, not left-to-right sampling of individual motor steps. It organizes dependencies within the horizon while retaining continuous action generation (Zhao et al., 2023; Chi et al., 2023; Black et al., 2025; Kim et al., 2025). The decoded thought prefix remains available throughout decoding.

3.4 EMA-MEDIATED SELF-VERIFICATION

A latent channel trained only through its own decoder can co-adapt with it: the decoder learns to read whatever the generator emits, and nothing requires the code to be usable by any other action predictor. Self-verification adds that requirement. At each optimization step, the current EMA consumer reads the codes generated by the student and supplies an action-prediction loss; across steps, EMA updates expose the code generator to a sequence of slowly changing consumer parameter states, so the codes cannot specialize to any single reader. This follows the momentum-teacher principle used in representation learning (Tarvainen & Valpola, 2017; Assran et al., 2023): weight averaging smooths the consumer's evolution while allowing it to track the student throughout training.

Let $\bar{\theta}$ and $\bar{\psi}$ denote the EMA action-model and WAE-decoder parameters. The EMA consumer decodes the student's codes as $\hat{\tau}^{\text{ema}} = p_{\bar{\psi}}(z)$ and predicts the same target field:

$$\mathcal{L}_{\text{verify}} = \mathbb{E}\big[\|v_{\bar{\theta}}(o, x, x_t, t; \hat{\tau}^{\text{ema}}) - u_t\|_2^2\big]\,. \tag{3}$$

The consumer parameters are held fixed during differentiation and updated after each student update by $\bar{\theta} \leftarrow \gamma\bar{\theta} + (1-\gamma)\theta$, with the same rule for $\bar{\psi}$. Gradients through the consumer with respect to the injected codes supervise the student-side code generator. In I-JEPA, the momentum encoder supplies representation targets (Assran et al., 2023); here, the EMA consumer supplies an action-prediction loss, and its latent input remains on the differentiation path. Each step uses the single current consumer; the sequence of consumer states arises over training rather than by replaying codes through multiple checkpoints, and the objective supervises the action utility of the codes.

The full training objective is

$$\mathcal{L} = \mathcal{L}_{\text{fm}} + \mathcal{L}_{\text{wae}} + \lambda_{\text{verify}}\mathcal{L}_{\text{verify}}. \tag{4}$$

At inference, the policy generates the thought interface and decodes actions without a separate verification pass. The EMA consumer therefore adds training computation rather than an additional inference-time verification stage.

4 EXPERIMENTS

We evaluate whether training a continuous reasoning interface through action supervision improves control under changing scenes and goals. The results form one pattern: CR's gains over its backbone concentrate on position and task perturbations, every component ablation loses on the same two families, and the improvement carries over to subtask and full-task completion on two physical robots. We first measure robustness on LIBERO-PRO, then ablate the design, then test the physical robots, and finally examine how reasoning states evolve with the instruction and a mid-execution target displacement.

4.1 EXPERIMENTAL SETUP

Backbone and comparisons. CR augments the $\pi_{0.5}$ flow-matching VLA. The primary comparison is with $\pi_{0.5}$ without CR. OpenVLA-OFT (Kim et al., 2025), X-VLA (Zheng et al., 2025), VLA-Adapter (Wang et al., 2025), and DeepThinkVLA (Yin et al., 2025) provide additional policy baselines. We select DeepThinkVLA as an explicit text-reasoning baseline because it provides publicly available LIBERO-trained checkpoints and an evaluation implementation.[1] We evaluate

[1]ThinkAct (Huang et al., 2025) has not publicly released LIBERO checkpoints or evaluation code as of September 2026 (`https://jasper0314-huang.github.io/thinkact-vla/`).

Table 1: LIBERO-PRO success (%) across four suites. Within each suite, Obj., Pos., Sem., and Task denote object, position, semantic, and task perturbations. DeepThinkVLA uses the released RL checkpoint. Bold marks the highest score in each column.

Method	libero_10				libero_goal				libero_object				libero_spatial			
	Obj.	Pos.	Sem.	Task	Obj.	Pos.	Sem.	Task	Obj.	Pos.	Sem.	Task	Obj.	Pos.	Sem.	Task
OpenVLA-OFT	5.5	0.0	38.5	0.0	8.5	0.0	43.5	2.5	66.5	9.5	89.0	0.0	30.0	13.5	65.0	0.0
X-VLA	61.0	1.0	70.5	19.5	71.5	1.0	94.0	7.5	91.5	4.5	98.0	0.0	89.5	0.0	69.0	38.5
VLA-Adapter	47.0	0.0	91.0	10.0	61.0	0.0	75.0	12.0	89.0	0.0	99.0	8.0	98.0	0.0	**98.0**	49.0
$\pi_{0.5}$	66.0	6.0	91.0	17.0	**90.0**	29.0	95.0	17.0	89.0	19.0	95.0	10.0	**99.0**	**53.0**	97.0	55.0
DeepThinkVLA	48.5	0.0	**94.5**	12.0	63.5	1.5	95.5	8.5	92.5	0.0	**99.5**	8.5	95.5	30.0	96.0	53.0
CR (Ours)	**66.5**	**15.5**	88.5	**28.5**	81.5	**36.0**	**96.0**	**33.0**	**93.0**	**54.5**	99.0	**27.0**	97.5	51.0	96.0	**60.0**

its released RL checkpoint under the same official static LIBERO-PRO protocol and episode budget as the other policies. Component ablations use the same $\pi_{0.5}$ backbone and training budget. Appendix A summarizes the reported CR configuration.

Benchmarks and metrics. LIBERO-PRO (Zhou et al., 2025a) tests four perturbation families across four LIBERO suites under the official static protocol. Each table cell contains 200 evaluation episodes: 20 trials for each of ten tasks. A suite mean equally weights its four perturbation rates; the overall mean equally weights all four suites. CALVIN ABC→D provides a complementary long-horizon evaluation. On physical robots, we report mean subtask success and full-task completion (E2E). Main-text robot comparisons include X-VLA, $\pi_{0.5}$, and CR; VLA-Adapter results and full E2E results appear in Appendix E.

Statistical reporting. We report nominal 95% Wilson confidence intervals (CIs) for real-robot success rates. E2E intervals use independent episode outcomes (ten per TX-G2 task; six per HSR task); subtask intervals approximate prescribed subtask opportunities as independent Bernoulli outcomes. Appendix E gives full intervals and sampling assumptions; bold marks the best point estimate.

4.2 Robustness to changes in scenes and goals

Evaluation under distribution shifts. Table 1 reports the main simulation comparison against strong VLA baselines. We report four official perturbation types used throughout this paper: *object*, which changes object appearance such as color, texture, or size; *position*, which changes object placement; *semantic*, which paraphrases the task instruction; and *task*, which evaluates transfer to different target task variants. Among these, *position* and *task* are the most revealing because they require the policy to re-anchor action under a changed layout or pursue a modified goal, rather than merely tolerate appearance shifts or instruction paraphrases.

CR improves robustness over its backbone. CR achieves the highest suite mean among the compared methods in each suite (Table 1), increasing overall mean success from 58.0% for $\pi_{0.5}$ to 64.0%. Position success averaged across suites rises from 26.8% to 39.3%, and task success from 24.8% to 37.1%. Object and semantic scores change less. CR improves task success in all four suites and position success in three; the gains concentrate on spatial retargeting and changed task goals.

Comparison with explicit text reasoning. DeepThinkVLA-RL achieves an overall mean of 49.9%, compared with 64.0% for CR, a gap of 14.0 percentage points computed from unrounded means. CR has a higher mean in all four suites and matches or exceeds DeepThinkVLA-RL in 14 of the 16 suite–perturbation cells. The largest difference is under position perturbations (39.3% versus 7.9%), followed by task perturbations (37.1% versus 20.5%). DeepThinkVLA-RL performs better on semantic perturbations (96.4% versus 94.9%). It uses a different backbone and training recipe; the gap decomposes into 8.1 points from the $\pi_{0.5}$ backbone alone (58.0% versus 49.9%) and a further 6.0 points from CR.

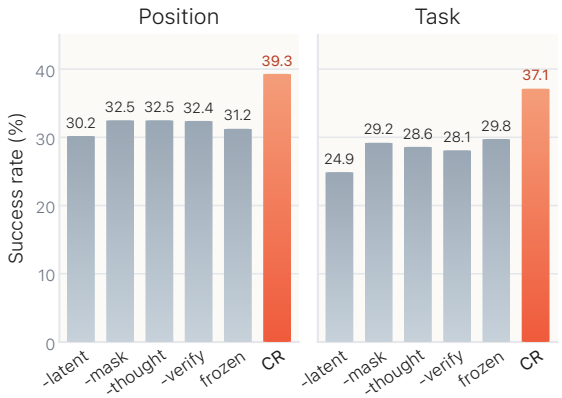


Figure 3: Component ablations on LIBERO-PRO, averaged across four suites. Labels denote removed components; *frozen* freezes the EMA consumer after warmup. Object and semantic scores move by less than 2 points in every variant (Table A2).

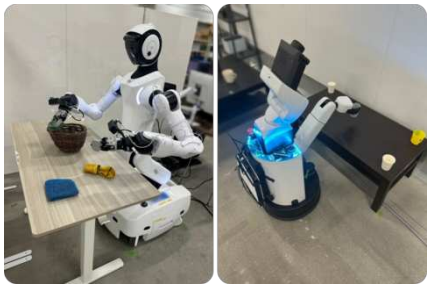


Figure 4: Real-robot platforms: TX-G2 (left) and HSR (right).

4.3 Component ablations

The gains above motivate a component-level question: which parts of CR support adaptation to changed positions and tasks? We remove Gaussian latent structuring, block-causal action masking, continuous thoughts, or self-verification, or freeze the EMA consumer after warmup, while retaining the same $\pi_{0.5}$ backbone and training budget. Figure 3 reports position and task success; Appendix C provides all perturbation types and full variant definitions.

Both the thought interface and its supervision contribute. Removing self-verification reduces overall success from 64.0% to 60.5%, supporting the benefit of EMA-mediated action supervision beyond the student's own action loss. Freezing the consumer after warmup leaves position and task success at 31.3% and 29.8% (overall 60.8%), indistinguishable from removing self-verification within evaluation noise: the generator adapts to a fixed reader as readily as it co-adapts with its own decoder, so the benefit requires a reader that keeps tracking the student. This is the property self-verification was designed for: a moving reader is one the generator cannot co-adapt to. Removing the entire thought branch yields 60.6% despite retaining the block-structured action decoder. This variant also removes the WAE and verification objectives, so it measures the contribution of the complete reasoning branch.

Latent structuring and temporal organization matter. Removing the Gaussian bottleneck produces the largest observed drop: overall success falls to 59.2%, with position and task success declining from 39.3% and 37.1% to 30.2% and 24.9%. This variant feeds uncompressed thoughts directly to both consumers, removing compression and WAE regularization together. Removing block-causal masking reduces overall success to 61.1%. Every ablation reduces position and task success, while object and semantic scores remain comparatively stable. This pattern links the combined design to the same perturbation families that account for CR's gains over its backbone.

We report CALVIN ABC→D in Appendix D as a complementary long-horizon benchmark, where Continuous Reasoning remains competitive, indicating that these LIBERO-PRO gains do not come at the expense of standard long-horizon execution.

4.4 From robustness benchmarks to physical task completion

On physical robots, CR improves mean subtask success over $\pi_{0.5}$ on both TX-G2 and HSR, with gains extending to full-task completion. These platforms test bimanual manipulation and mobile manipulation, respectively (Figure 4). Table 2 summarizes subtask success; we examine E2E outcomes alongside each platform below. All reported real-robot baselines are fine-tuned for 150k

Table 2: Real-robot subtask success on TX-G2 and HSR. Each cell shows success (%) above its nominal 95% Wilson CI, assuming independent subtask outcomes. TX-G2 denominators are 40, 40, 60, 40; HSR denominators are 12, 24, 12, 12. VLA-Adapter and full E2E results appear in Appendix E.

Method	TX-G2				HSR			
	Cutl.	Bowl	Cloth.	Dish	Bott.-1	Bott.-2	Box	Mug
X-VLA	0.0 [0.0, 8.8]	7.5 [2.6, 19.9]	5.0 [1.7, 13.7]	5.0 [1.4, 16.5]	8.3 [1.5, 35.4]	4.2 [0.7, 20.2]	0.0 [0.0, 24.2]	25.0 [8.9, 53.2]
$\pi_{0.5}$	5.0 [1.4, 16.5]	47.5 [32.9, 62.5]	83.3 [72.0, 90.7]	60.0 [44.6, 73.7]	**83.3** [55.2, 95.3]	45.8 [27.9, 64.9]	41.7 [19.3, 68.0]	66.7 [39.1, 86.2]
CR (Ours)	**22.5** [12.3, 37.5]	**70.0** [54.6, 81.9]	**95.0** [86.3, 98.3]	**87.5** [73.9, 94.5]	**83.3** [55.2, 95.3]	**83.3** [64.1, 93.3]	**58.3** [32.0, 80.7]	**75.0** [46.8, 91.1]

steps. We exclude OpenVLA-OFT from this comparison because our RTX 5070 evaluation setup runs out of memory on both platforms under the deployed real-robot inference pipeline.

TX-G2: bimanual task execution. TX-G2 (an AgiBot G2-compatible variant) has three cameras and 1198 finetuning trajectories across four task families, and we query the deployed policy at 10 Hz. We evaluate each task with ten episodes, of which eight use in-distribution placements and two use substantially shifted out-of-distribution placements. For fairness, all methods are evaluated on the same ten initial states and object placements for each task. Objects may appear on either the left or right side of the workspace, so the policy must also decide which arm to use. We consider four long-horizon tasks: *Cutlery Transfer*, *Bowl Stacking*, *Clothes Sorting*, and *Dish Racking*.

CR improves TX-G2 subtask success on all four tasks, raising the equally weighted task mean from 49.0% to 68.8%. These gains extend to full-task completion: E2E success increases from 0% [0.0, 27.8] to 40% [16.8, 68.7] on *Bowl Stacking*, from 50% [23.7, 76.3] to 80% [49.0, 94.3] on *Clothes Sorting*, and from 20% [5.7, 51.0] to 60% [31.3, 83.2] on *Dish Racking* ($\pi_{0.5}$ to CR; brackets denote 95% Wilson CIs in percent). Both methods complete 0/10 cutlery-transfer trials.

HSR: mobile task execution. HSR uses 1205 finetuning trajectories across four mobile-manipulation task families, and we query the deployed policy at 2 Hz. We use four tasks that combine navigation, manipulation, and constrained placement geometry: *Coffee Bottle* $\rightarrow$ *Box*, *Coffee Bottles* $\rightarrow$ *Table*, *Box Rearrangement*, and *Mug Rectangle*. Full task descriptions appear in Appendix E.

Continuous Reasoning also improves HSR subtask success on three of the four mobile-manipulation tasks and ties the fourth, with the largest gain on *Coffee Bottles* $\rightarrow$ *Table*. The task requires repeated locomotion and manipulation over a long action chain: approach the shelf, pick a bottle, carry it to the table, return to the shelf, and repeat for the second bottle. The equally weighted HSR task mean increases from 59.4% to 75.0%. On this two-bottle task, E2E success also increases from 0% [0.0, 39.0] for $\pi_{0.5}$ to 66.7% [30.0, 90.3] for CR (0/6 versus 4/6 episodes; 95% Wilson CIs).

4.5 Reasoning states and behavior during execution

We now examine the intermediate states and behavior behind individual rollouts: how trajectories differ with the instruction and how they change after a target displacement.

Instruction-dependent trajectories in simulation. Figure 5 compares instruction pairs with five rollouts per instruction from the same initial scene. The cup/book trajectories approach one another near contact and separate during placement; the bowl trajectories differ before grasping and exhibit related shapes during relocation. Because each pair starts from the same scene, the separation is driven by the instruction alone.

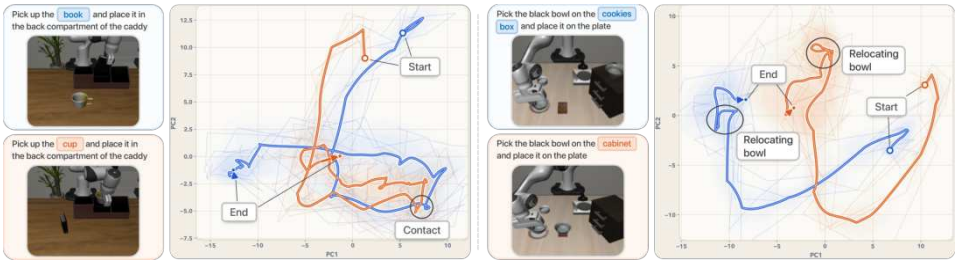


Figure 5: PCA visualization of continuous reasoning trajectories on paired LIBERO-PRO scenes. Each instruction is evaluated with five rollouts from the same initial scene; trajectory color matches the highlighted word in each instruction.

Response to a displaced target on TX-G2. Figure 6 provides a diagnostic view of continuous reasoning on the TX-G2 clothes-sorting subtask "pick up the green socks." We visualize three matched scene variants: the target pair of green socks starts on the left (top), starts on the right (bottom), or starts on the left and is abruptly thrown to the right by an external perturbation during execution (middle). This kind of within-episode target displacement does not appear in the dataset. Nevertheless, the perturbed middle rollout initially follows the left-target reasoning pattern, but its final reasoning state moves toward the right-target configuration. This example illustrates how the reasoning states change as the policy adapts to a displaced target.

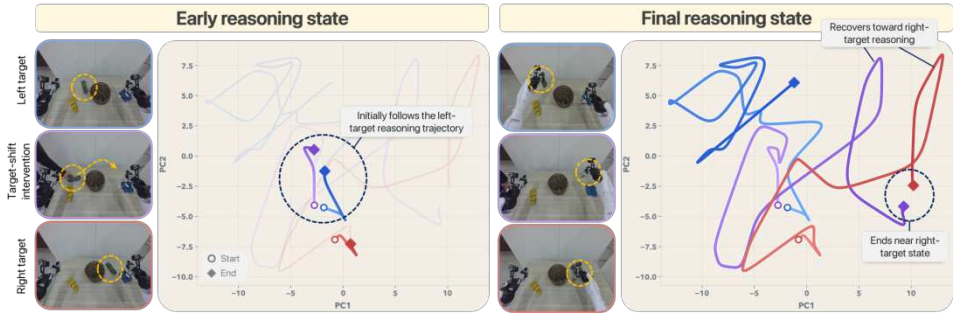


Figure 6: PCA visualization of continuous reasoning on TX-G2 "pick up the green socks": target on the left (top), thrown from left to right mid-episode (middle), or on the right (bottom). The displaced rollout starts on the left-target trajectory and ends near the right-target state.

In a complementary TX-G2 rollout, CR completes ordered clothes sorting (green socks, handkerchief, yellow socks) while a human introduces objects and moves the basket; the setup and full frame sequence appear in Figure A5 (Appendix E). Together, these rollouts show the reasoning states tracking the instruction and the target as they change, the same re-anchoring that the position and task gains measure.

5 Conclusion

Continuous Reasoning learns intermediate thoughts through the actions they enable a VLA policy to predict. Its central training mechanism, self-verification, defines a thought by its usefulness to a second, evolving reader: student-generated codes are passed to an EMA consumer whose action-prediction gradients train the code generator, with Gaussian-prior regularization and block-causal decoding structuring the interface. Over $\pi_{0.5}$, CR's LIBERO-PRO gains concentrate on position and task perturbations, every component ablation loses on those same two families, and a frozen-consumer control shows that the benefit of self-verification requires an evolving reader. It also exceeds DeepThinkVLA-RL in every LIBERO-PRO suite mean, by 14.0 points overall, and raises full-task completion on physical robots from 0/10 to 4/10 on bowl stacking and from 0/6 to 4/6 on a two-bottle mobile task. Since a frozen reader confers no benefit, a natural next step is to ask which other readers, including independently trained policies, can supervise a shared reasoning interface.

REPRODUCIBILITY STATEMENT

Section 3 specifies the latent interface and training objectives. Appendix A records the reported model configuration and distributed latent regularization, and the benchmark appendices describe the evaluation protocols, dataset composition, and full result tables.

AI USE STATEMENT

Generative AI tools assisted with code generation and debugging, LaTeX formatting and statistical-reporting scripts, literature and submission-guideline lookup, and manuscript drafting and editing. The authors are responsible for the final text, claims, code, and reported results.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can, not as i say: Grounding language in robotic affordances, 2022. URL https://arxiv.org/abs/2204.01691.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture, 2023. URL https://arxiv.org/abs/2301.08243.

Shuanghao Bai, Jing Lyu, Wanqi Zhou, Zhe Li, Dakai Wang, Lei Xing, Xiaoguang Zhao, Pengwei Wang, Zhongyuan Wang, Cheng Chi, Badong Chen, and Shanghang Zhang. Latent reasoning vla: Latent thinking and prediction for vision-language-action models, 2026. URL https://arxiv.org/abs/2602.01166.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.010. URL https://www.roboticsproceedings.org/rss21/p010.html.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale, 2022. URL https://arxiv.org/abs/2212.06817.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul

Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023. URL https://arxiv.org/abs/2307.15818.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021. URL https://arxiv.org/abs/2104.14294.

Haonan Chen, Jingxiang Guo, Bangjun Wang, Tianrui Zhang, Xuchuan Huang, Boren Zheng, Yiwen Hou, Chenrui Tie, Jiajun Deng, and Lin Shao. Goal-vla: Image-generative vlms as object-centric world models empowering zero-shot robot manipulation, 2025. URL https://arxiv.org/abs/2506.23919.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2023. URL https://arxiv.org/abs/2303.04137.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model, 2023. URL https://arxiv.org/abs/2303.03378.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning, 2020. URL https://arxiv.org/abs/2006.07733.

Siddhant Haldar, Zhuoran Peng, and Lerrel Pinto. Baku: An efficient transformer for multi-task policy learning, 2024. URL https://arxiv.org/abs/2406.07539.

Zhi Hou, Tianyi Zhang, Yuwen Xiong, Hengjun Pu, Chengyang Zhao, Ronglei Tong, Yu Qiao, Jifeng Dai, and Yuntao Chen. Diffusion transformer policy, 2024. URL https://arxiv.org/abs/2410.15959.

Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Yu-Chiang Frank Wang, and Fu-En Yang. Thinkact: Vision-language-action reasoning via reinforced visual latent planning, 2025. URL https://arxiv.org/abs/2507.16815.

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, 2026. URL https://arxiv.org/abs/2601.09708.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models, 2022. URL https://arxiv.org/abs/2207.05608.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models, 2023. URL https://arxiv.org/abs/2307.05973.

Zixuan Huang, Yating Lin, Fan Yang, and Dmitry Berenson. Subgoal diffuser: Coarse-to-fine subgoal generation to guide model predictive control for robot manipulation, 2024. URL https://arxiv.org/abs/2403.13085.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

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

Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, Winson Han, Wilbert Pumacay, Angelica Wu, Rose Hendrix, Karen Farley, Eli VanderBilt, Ali Farhadi, Dieter Fox, and Ranjay Krishna. Molmoact: Action reasoning models that can reason in space, 2025. URL https://arxiv.org/abs/2508.07917.

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. 2024. doi: 10.48550/ARXIV.2403.03181. URL https://arxiv.org/abs/2403.03181.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control, 2022. URL https://arxiv.org/abs/2209.07753.

Yiran Ling, Qing Lian, Jinghang Li, Qing Jiang, Tianming Zhang, Xiaoke Jiang, Chuanxiu Liu, Jie Liu, and Lei Zhang. Guide, think, act: Interactive embodied reasoning in vision-language-action models, 2026. URL https://arxiv.org/abs/2605.13632.

Litao Liu, Wentao Wang, Yifan Han, Zhuoli Xie, Pengfei Yi, Junyan Li, Yi Qin, and Wenzhao Lian. Foam: Foresight-augmented multi-task imitation policy for robotic manipulation, 2024. URL https://arxiv.org/abs/2409.19528.

Zhuoyang Liu, Jiaming Liu, Hao Chen, Jiale Yu, Ziyu Guo, Chengkai Hou, Chenyang Gu, Xiangju Mi, Renrui Zhang, Kun Wu, Zhengping Che, Jian Tang, Pheng-Ann Heng, and Shanghang Zhang. Last$_0$: Latent spatio-temporal chain-of-thought for robotic vision-language-action model, 2026. URL https://arxiv.org/abs/2601.05248.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play, 2019. URL https://arxiv.org/abs/1903.01973.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy, 2024. URL https://arxiv.org/abs/2405.12213.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, and Xuelong Li. Spatialvla: Exploring spatial representations for visual-language-action model, 2025. URL https://arxiv.org/abs/2501.15830.

Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone, 2022. URL https://arxiv.org/abs/2206.11251.

Quanxin Shou, Fangqi Zhu, Shawn Chen, Puxin Yan, Zhengyang Yan, Yikun Miao, Xiaoyi Pang, Zicong Hong, Ruikai Shi, Hao Huang, Jie Zhang, and Song Guo. Halo: A unified vision-language-action model for embodied multimodal chain-of-thought reasoning, 2026. URL https://arxiv.org/abs/2602.21157.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results, 2017. URL https://arxiv.org/abs/1703.01780.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders, 2017. URL https://arxiv.org/abs/1711.01558.

Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, Siteng Huang, Yifan Tang, Wenhui Wang, Ru Zhang, Jianyi Liu, and Donglin Wang. Vla-adapter: An effective paradigm for tiny-scale vision-language-action model, 2025. URL https://arxiv.org/abs/2509.09372.

Junjie Wen, Minjie Zhu, Jiaming Liu, Zhiyuan Liu, Yicun Yang, Linfeng Zhang, Shanghang Zhang, Yichen Zhu, and Yi Xu. dvla: Diffusion vision-language-action model with multimodal chain-of-thought, 2025. URL https://arxiv.org/abs/2509.25681.

Shuai Yang, Hao Li, Bin Wang, Yilun Chen, Yang Tian, Tai Wang, Hanqing Wang, Feng Zhao, Yiyi Liao, and Jiangmiao Pang. Instructvla: Vision-language-action instruction tuning from understanding to manipulation, 2025. URL https://arxiv.org/abs/2507.17520.

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. URL https://arxiv.org/abs/2511.15669.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Ming-Yu Liu, Donglai Xiang, Gordon Wetzstein, and Tsung-Yi Lin. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. 2025. doi: 10.48550/ARXIV.2503.22020. URL https://arxiv.org/abs/2503.22020.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL https://arxiv.org/abs/2304.13705.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Ya-Qin Zhang, Jiangmiao Pang, Jingjing Liu, Tai Wang, and Xianyuan Zhan. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model, 2025. URL https://arxiv.org/abs/2510.10274.

Linqing Zhong, Yi Liu, Yifei Wei, Ziyu Xiong, Maoqing Yao, Si Liu, and Guanghui Ren. Acot-vla: Action chain-of-thought for vision-language-action models, 2026a. URL https://arxiv.org/abs/2601.11404.

Zhide Zhong, Junfeng Li, Junjie He, Haodong Yan, Xin Gong, Guanyi Zhao, Yingjie Cai, Jiantao Gao, Xu Yan, Bingbing Liu, Yingcong Chen, Liuqing Yang, and Haoang Li. Dualcot-vla: Visual-linguistic chain of thought via parallel reasoning for vision-language-action models, 2026b. URL https://arxiv.org/abs/2603.22280.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. Libero-pro: Towards robust and fair evaluation of vision-language-action models beyond memorization, 2025a. URL https://arxiv.org/abs/2510.03827.

Zhongyi Zhou, Yichen Zhu, Junjie Wen, Chaomin Shen, and Yi Xu. Chatvla-2: Vision-language-action model with open-world embodied reasoning from pretrained knowledge, 2025b. URL https://arxiv.org/abs/2505.21906.

A EXPERIMENTAL CONFIGURATION

Table A1 summarizes the reported default CR configuration. The settings describe CR rather than the independently released baseline policies. The sections below provide benchmark protocols and component-ablation definitions.

Table A1: Reported default H16 configuration for CR.

Symbol	Value	Setting
H	16	action horizon
C	4	internal action-block length
K	4	number of action blocks ($H = KC$)
N_τ	2	maximum thought slots
d_z	128	thought latent dimension
λ_{verify}	0.1	constant verification weight
γ_{EMA}	0.994	EMA decay
$\mathcal{S}_{\text{curr}}$	$[0, 2000]$	thought curriculum steps

Distributed WAE-MMD implementation. In multi-GPU, multi-node training, the WAE prior-matching loss should not be computed independently on local shards. Instead, we first aggregate the latent codes and Gaussian prior samples across all devices to form one global batch, and then compute a single IMQ-kernel MMD loss on that global batch. This detail matters because MMD is a batch-level discrepancy: if each device computes its own local MMD, the kernel only sees small per-device sub-batches and no longer matches the intended global latent distribution. This implementation detail applies to CR variants that retain the WAE objective.

B LIBERO-PRO EVALUATION DETAILS

Our LIBERO-PRO results follow the official static benchmark protocol released with LIBERO-PRO. We evaluate on four suites: `libero_spatial`, `libero_goal`, `libero_10`, and `libero_object`. Each suite contains 10 tasks. For every suite–perturbation pair, we run 20 trials per task, yielding 200 evaluation episodes for each cell reported in Table 1.

We report four official perturbation types: *object*, *position*, *semantic*, and *task*. Success rates are reported as percentages over the corresponding 200 episodes. The suite mean discussed in the main text is the arithmetic mean of these four perturbation success rates within the same suite block.

This protocol is intentionally more diagnostic than a single in-distribution success rate. Because each perturbation family isolates a different source of shift, the full LIBERO-PRO table reveals whether a policy mainly benefits from appearance robustness, instruction robustness, spatial retargeting, or task-level generalization.

C LIBERO-PRO ABLATION DETAILS

Table A2 reports the full LIBERO-PRO ablation breakdown averaged across all four suites. The variants remove one major component of Continuous Reasoning at a time, or freeze the EMA consumer after warmup, while keeping the same $\pi_{0.5}$ backbone and training budget.

Table A2: Full LIBERO-PRO ablation results averaged across the four suites. The full Continuous Reasoning model is lightly highlighted for readability.

Method	Object	Position	Semantic	Task	Overall
Full CR	84.6	**39.3**	94.9	**37.1**	**64.0**
w/o Gaussian latent space	86.0	30.2	95.5	24.9	59.2
w/o block-causal mask	**86.5**	32.5	96.0	29.2	61.1
w/o continuous thoughts	84.9	32.5	**96.5**	28.6	60.6
w/o self-verification	86.0	32.4	95.5	28.1	60.5
Frozen EMA consumer	86.0	31.3	96.0	29.8	60.8

For clarity, we summarize the precise implementation of each ablation below.

Ablation definitions. **w/o self-verification** removes the verification loss L_{verify} and does not train against the EMA consumer. The model still predicts continuous thoughts and still uses latent structuring, but those thoughts no longer receive auxiliary action supervision through the EMA consumer.

w/o Gaussian latent space removes the WAE objective and the Gaussian bottleneck. Instead of decoding thoughts from a structured latent code, we pass the model's original uncompressed continuous thought vectors directly to the action predictor and to the EMA consumer. This tests whether latent structuring itself matters beyond simply having a continuous intermediate representation.

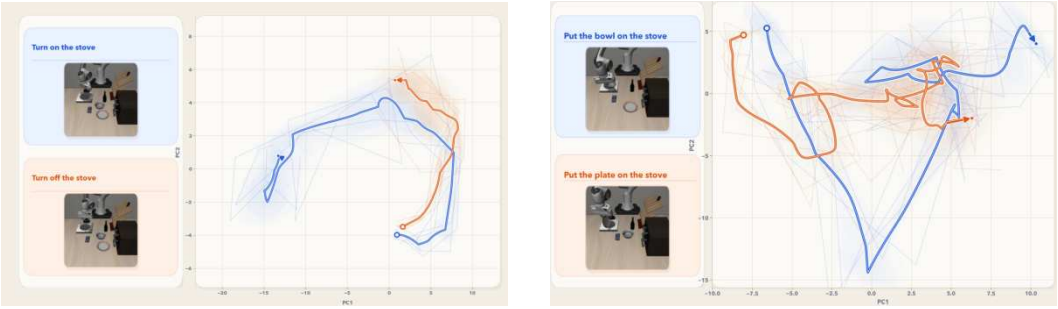
w/o block-causal mask keeps continuous thoughts, latent structuring, and self-verification, but removes the block-causal dependency across action blocks. In this variant, the flow-matching decoder treats the full action horizon as a single block rather than imposing the AR-like temporal dependency used by the full method.

w/o continuous thoughts is the action-only variant. We remove the continuous reasoning branch entirely and keep only the block-structured flow-matching action decoder. Because no thought interface is produced, this variant also has no latent-structuring objective and no self-verification objective.

Frozen EMA consumer keeps the full architecture and the verification loss but stops updating the EMA consumer after the initial warmup phase, so the student's codes are supervised by one fixed second reader for the rest of training. Its position and task success match the w/o self-verification variant within evaluation noise, indicating that the verification gradient is useful only while the consumer keeps tracking the student.

The same pattern as the main-text plot is visible here. Object and semantic perturbations remain relatively stable, but every ablation substantially reduces position and task success. The largest ablation effects are in spatial retargeting and adaptation to changed task structure.

Additional LIBERO paired-scene latent diagnostics. We additionally visualize paired LIBERO scenes that share the same initial state but differ only in instruction, to further probe how the latent reasoning interface reorganizes under changed goals. These plots give a more direct qualitative view of what the learned latent preserves and where it diverges.



Turn on vs. turn off the stove **Place different objects onto the stove**

Figure A1: Additional LIBERO paired-scene latent visualizations under matched initial states. Left: the two instructions initially produce closely aligned latent trajectories because both begin with the same approach-to-stove motion; the *turn off* rollout then terminates early because the stove already starts in the off state. Right: when placing different objects onto the stove, the latent state is similar near the initial and final phases, where the start and placement geometry are analogous, but diverges in the middle while the policy reaches to different object locations.

D Additional Results on CALVIN

We report CALVIN ABC→D as a complementary long-horizon benchmark. While it is not the primary benchmark used to support our main empirical claims, it provides a useful check that Contin-

uous Reasoning remains competitive on standard long-horizon manipulation evaluation. We report only average sequence length here.

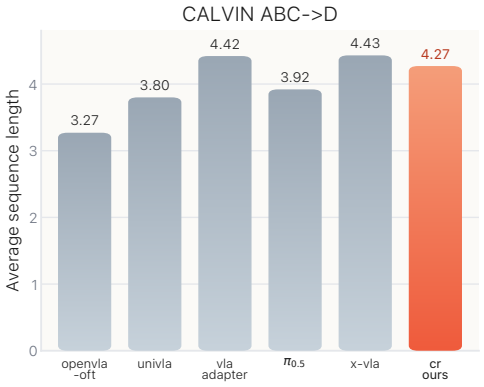


Figure A2: CALVIN ABC→D average sequence length.

As shown in Figure A2, Continuous Reasoning remains close to the strongest baselines on CALVIN ABC→D, indicating that the robustness gains observed in the main text are not purchased by sacrificing standard long-horizon performance.

E Real-Robot Evaluation Details

Shared evaluation setup. All reported real-robot baselines are fine-tuned for 150k steps. We exclude OpenVLA-OFT from both TX-G2 and HSR because our RTX 5070 evaluation setup runs out of memory under the deployed real-robot inference pipeline.

TX-G2 protocol. TX-G2 (an AgiBot G2-compatible variant) is a bimanual platform equipped with three cameras. Each task is evaluated with 10 episodes: 8 use in-distribution object placements that remain close to the training distribution without exactly duplicating dataset states, and 2 use substantially shifted out-of-distribution placements. For fairness, all compared methods are evaluated on the same 10 initial states and object placements for each task. Objects may appear on either the left or right side of the workspace, so the policy must also decide which arm to use.

TX-G2 dataset composition. The TX-G2 finetuning dataset contains 1198 source trajectories across four long-horizon task families. Table A3 summarizes the trajectory count for each family, and the short descriptions below give the canonical subtask sequence that appears most often in each case.

TX-G2 tasks. **Cutlery Transfer** requires moving a spoon and a fork from one bowl to another. The utensils are thin and difficult to grasp, and lifting the source bowl counts as failure.

Bowl Stacking presents three bowls with different colors and asks the robot to stack the instructed target bowls.

Clothes Sorting places two pairs of socks together with a handkerchief on the table and requires the instructed items to be placed into a basket.

Dish Racking uses two colored plates and requires placing the instructed plate onto a dish rack with limited clearance, making orientation correction particularly important.

Table A3: Composition of the TX-G2 finetuning dataset.

Task family	Trajectories
Bowl Stacking	129
Clothes Sorting	514
Cutlery Transfer	206
Dish Racking	349

Canonical TX-G2 subtask sequences. **Bowl Stacking.** A typical sequence is: pick up the yellow bowl from the desk, stack the yellow bowl on the grey bowl, pick up the light-blue bowl from the desk, and stack the light-blue bowl on the yellow bowl.

Clothes Sorting. A typical sequence is: pick up the green socks from the desk, place the green socks into the basket, pick up the handkerchief from the desk, place the handkerchief into the basket, pick up the yellow socks from the desk, and place the yellow socks into the basket.

Cutlery Transfer. A typical sequence is: pick up the light-blue spoon from the grey bowl, place the spoon in the yellow bowl, pick up the pink fork from the grey bowl, and place the fork in the yellow bowl.

Dish Racking. A typical sequence is: pick up the yellow dish from the desk, place the yellow dish in the wooden dish rack, pick up the green dish from the desk, and place the green dish in the wooden dish rack.



Figure A3: Representative TX-G2 trajectory snapshots for the four long-horizon tasks described above. Each panel shows a rollout of the corresponding task family used in our real-robot evaluation.

Additional TX-G2 latent diagnostic under dynamic object injection. We also visualize a more adversarial TX-G2 setup in which the workspace initially contains no relevant object and a human successively throws different objects into view. Only green socks, yellow socks, a blue handkerchief, and the basket appear in the training data; at test time, we additionally inject distractors such as green fruit. The resulting latent trajectories show how the interface responds as distractors and target objects enter the scene.

Ordered TX-G2 execution under human intervention. We examine a qualitative clothes-sorting rollout in which the workspace initially contains only the basket. The instruction requires picking and placing green socks, then the handkerchief, and finally yellow socks. During execution, a nearby human moves the basket and introduces additional objects. CR completes the requested sequence as the scene changes. Figure A5 shows the full eight-frame summary, including the added objects and a visible human hand; the snapshots do not resolve every individual intervention. This example supplements the quantitative evaluation with a view of ordered execution under ongoing scene changes.

HSR tasks. HSR emphasizes long-horizon mobile manipulation with locomotion. Each HSR task is evaluated with 6 episodes per method. The scored subtasks are pick and place: Bottle–Box, Two Bottles, Box Rearrangement, and Mug Rectangle contain 2, 4, 2, and 2 scored subtasks per episode, respectively. Navigation is required for execution but is not scored separately. Thus, the subtask denominators are 12, 24, 12, and 12; prescribed subtasks not completed after an earlier failure remain in the denominator.

HSR dataset composition. The HSR finetuning dataset contains 1205 source trajectories across four long-horizon mobile-manipulation task families. Table A4 summarizes the trajectory count for each family, and the short descriptions below give the canonical subtask sequence that appears most often in each case.

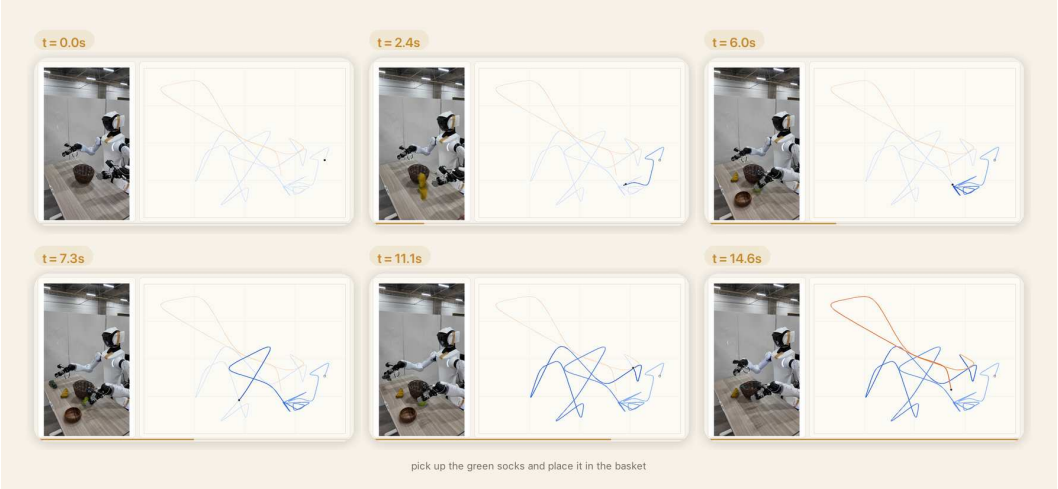


Figure A4: TX-G2 latent visualization under dynamic object injection for the clothes-sorting task. Before the target green socks appear, the continuous reasoning latent remains clustered near the same state around t=6.0 despite incoming distractor objects. Once the green socks enter the scene, the latent sharply transitions into a different region and then proceeds through distinct pickup and placement phases. In the plot, blue markers denote the pickup stage and brown markers denote the placement stage, illustrating how the latent trajectory changes between the two action phases.

Table A4: Composition of the HSR finetuning dataset.

Task family	Trajectories
Mug Rectangle	399
Coffee Bottle → Box	295
Box Rearrangement	195
Coffee Bottles → Table	316

Coffee Bottle → Box requires HSR to move from a start position to a coffee bottle, grasp it, navigate to the area in front of the designated box, and insert the bottle into that box. This task is difficult because the clearance above the target box is small, so locomotion error makes precise placement substantially harder.

Coffee Bottles → Table requires moving two coffee bottles from a shelf to the table and placing them near the mugs. This task combines locomotion with repeated pick-and-place under clutter, and requires stable transport from an elevated shelf to a lower placement surface.

Box Rearrangement requires moving the box labeled 2 beside the box labeled 1. Execution involves moving to box 2, grasping box 2, navigating to the area in front of box 1, and placing box 2 beside box 1; only grasping and placing are scored as subtasks. The box is large and relatively heavy for the gripper, making stable transport and final placement both challenging.

Mug Rectangle requires moving mugs so that the final layout forms a rectangular arrangement. The policy must reason about which mug should move and where the missing rectangle corner lies before executing the relocation.

Canonical HSR subtask sequences. **Mug Rectangle.** A typical sequence is: pick up the mug that is not at a rectangle corner, and place it at the missing rectangle corner.

Coffee Bottle → Box. A typical sequence is: pick up the coffee bottle on the right, and place it into the box labeled 1.

Box Rearrangement. A typical sequence is: pick up the box labeled 2, and place it beside the box labeled 1.



Figure A5: Ordered clothes sorting as the workspace changes. Read left to right, then top to bottom. The workspace initially contains only the basket (1.8 s); target and distractor objects appear in subsequent frames, with a human hand visible at the lower edge at 53.1 s. CR handles the instructed items in order: green socks (22.6 s), the handkerchief (61.3–68.1 s), and yellow socks (102.0–108.1 s).

Coffee Bottles → Table. A typical sequence is: pick up the right-hand bottle from the shelves, place it next to any mug on the table, pick up the left-hand bottle from the shelves, and place it next to a mug that does not already have a bottle.

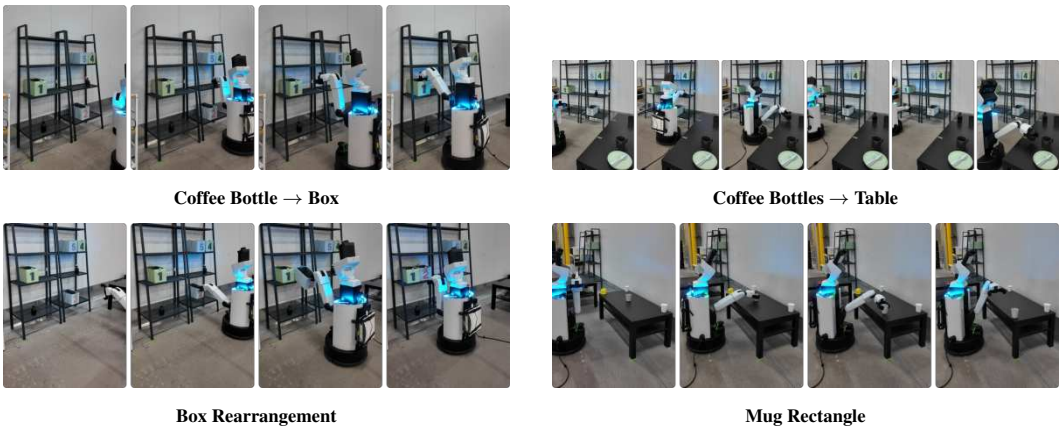


Figure A6: Representative HSR trajectory snapshots for the four mobile-manipulation tasks described above. Each panel shows a rollout of the corresponding task family used in our real-robot evaluation.

Full real-robot tables. In the main-text real-robot table, TX-G2 shorthand headers *Cutl.*, *Bowl*, *Cloth.*, and *Dish* denote *Cutlery Transfer*, *Bowl Stacking*, *Clothes Sorting*, and *Dish Racking*, respec-

tively. HSR shorthand headers *Bott.-1*, *Bott.-2*, *Box*, and *Mug* denote *Coffee Bottle → Box*, *Coffee Bottles → Table*, *Box Rearrangement*, and *Mug Rectangle*, respectively.

Table A5 reports the full TX-G2 metrics, including both mean subtask success and E2E completion. Table A6 provides the corresponding HSR table with the same metrics.

Table A5: Full TX-G2 real-robot results on four long-horizon tasks. Subtask denotes mean subtask success (%); E2E measures full-task completion (%). Entries are point estimates. Nominal E2E and subtask intervals are reported in Tables A7 and A8.

Method	Cutlery Transfer		Bowl Stacking		Clothes Sorting		Dish Racking	
	Subtask	E2E	Subtask	E2E	Subtask	E2E	Subtask	E2E
X-VLA	0.0	0.0	7.5	0.0	5.0	0.0	5.0	0.0
VLA-Adapter	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$\pi_{0.5}$	5.0	0.0	47.5	0.0	83.3	50.0	60.0	20.0
CR (Ours)	**22.5**	**0.0**	**70.0**	**40.0**	**95.0**	**80.0**	**87.5**	**60.0**

VLA-Adapter deployment checks on TX-G2. We did not find evidence of a large inference-time integration error for VLA-Adapter. Its preprocessing path, proprioceptive normalization, and bundle-level input/output contract all passed our validation checks. However, in a 30-sample dataset-replay sanity test, the predicted actions were only marginally better than a simple mean-action baseline. These checks do not establish the cause of the 0% closed-loop success. We retain the result as an outcome of the evaluated adaptation and deployment configuration, rather than as a general claim about VLA-Adapter.

Table A6: Full HSR real-robot results on four long-horizon mobile-manipulation tasks. Subtask denotes mean subtask success (%); E2E measures full-task completion (%). Entries are point estimates. Nominal E2E and subtask intervals appear in Tables A9 and A10.

Method	Bottle Box		Two Bottles		Box Rearr.		Mug Rect.	
	Subtask	E2E	Subtask	E2E	Subtask	E2E	Subtask	E2E
X-VLA	8.3	0.0	4.2	0.0	0.0	0.0	25.0	16.7
VLA-Adapter	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$\pi_{0.5}$	**83.3**	**66.7**	45.8	0.0	41.7	**16.7**	66.7	33.3
CR (Ours)	**83.3**	**66.7**	**83.3**	**66.7**	**58.3**	**16.7**	**75.0**	**50.0**

TX-G2 E2E confidence intervals. Table A7 reports nominal 95% Wilson score intervals from the E2E success counts in Table A5 and ten trials per task, treating trials as independent draws with a common success probability. Because each task mixes eight near-distribution and two shifted placements, the intervals do not separately quantify either stratum, training-seed variability, or paired method differences.

Table A7: TX-G2 E2E successes/trials and nominal 95% Wilson intervals (%). Zero observed successes does not imply zero uncertainty.

Method	Cutlery	Bowl	Clothes	Dish
X-VLA	0/10 [0.0, 27.8]	0/10 [0.0, 27.8]	0/10 [0.0, 27.8]	0/10 [0.0, 27.8]
VLA-Adapter	0/10 [0.0, 27.8]	0/10 [0.0, 27.8]	0/10 [0.0, 27.8]	0/10 [0.0, 27.8]
$\pi_{0.5}$	0/10 [0.0, 27.8]	0/10 [0.0, 27.8]	5/10 [23.7, 76.3]	2/10 [5.7, 51.0]
CR (Ours)	0/10 [0.0, 27.8]	4/10 [16.8, 68.7]	8/10 [49.0, 94.3]	6/10 [31.3, 83.2]

TX-G2 subtask confidence intervals. Subtask success divides completed subtasks by all prescribed subtask opportunities, including those not attempted after an earlier failure; with ten episodes and 4, 4, 6, and 4 subtasks per episode for Cutlery, Bowl, Clothes, and Dish, the denominators are 40, 40, 60, and 40. Table A8 reports nominal 95% Wilson intervals under an independent Bernoulli model for these opportunities. This approximation ignores within-episode dependence and differ-

ences in subtask difficulty and may understate uncertainty when outcomes are positively correlated within episodes.

Table A8: TX-G2 completed/prescribed subtasks and nominal 95% Wilson intervals (%), assuming independent subtask outcomes.

Method	Cutlery	Bowl	Clothes	Dish
X-VLA	0/40 [0.0, 8.8]	3/40 [2.6, 19.9]	3/60 [1.7, 13.7]	2/40 [1.4, 16.5]
VLA-Adapter	0/40 [0.0, 8.8]	0/40 [0.0, 8.8]	0/60 [0.0, 6.0]	0/40 [0.0, 8.8]
$\pi_{0.5}$	2/40 [1.4, 16.5]	19/40 [32.9, 62.5]	50/60 [72.0, 90.7]	24/40 [44.6, 73.7]
CR (Ours)	9/40 [12.3, 37.5]	28/40 [54.6, 81.9]	57/60 [86.3, 98.3]	35/40 [73.9, 94.5]

HSR confidence intervals. Tables A9 and A10 report nominal 95% Wilson intervals from the aggregate counts in Table A6: six episodes per method and task for E2E, and 12, 24, 12, and 12 prescribed subtask opportunities for Bottle–Box, Two Bottles, Box Rearrangement, and Mug Rectangle, including opportunities not attempted after an earlier failure. The same independent-outcome approximation and caveats as for TX-G2 apply; neither set of intervals quantifies training-seed variability or paired method differences.

Table A9: HSR E2E successes/episodes. Nominal 95% Wilson intervals (%).

Method	Bottle–Box	Two Bottles	Box Rearr.	Mug Rect.
X-VLA	0/6 [0.0, 39.0]	0/6 [0.0, 39.0]	0/6 [0.0, 39.0]	1/6 [3.0, 56.4]
VLA-Adapter	0/6 [0.0, 39.0]	0/6 [0.0, 39.0]	0/6 [0.0, 39.0]	0/6 [0.0, 39.0]
$\pi_{0.5}$	4/6 [30.0, 90.3]	0/6 [0.0, 39.0]	1/6 [3.0, 56.4]	2/6 [9.7, 70.0]
CR (Ours)	4/6 [30.0, 90.3]	4/6 [30.0, 90.3]	1/6 [3.0, 56.4]	3/6 [18.8, 81.2]

Table A10: HSR completed/prescribed subtasks; independent-subtask approximation. Nominal 95% Wilson intervals (%).

Method	Bottle–Box	Two Bottles	Box Rearr.	Mug Rect.
X-VLA	1/12 [1.5, 35.4]	1/24 [0.7, 20.2]	0/12 [0.0, 24.2]	3/12 [8.9, 53.2]
VLA-Adapter	0/12 [0.0, 24.2]	0/24 [0.0, 13.8]	0/12 [0.0, 24.2]	0/12 [0.0, 24.2]
$\pi_{0.5}$	10/12 [55.2, 95.3]	11/24 [27.9, 64.9]	5/12 [19.3, 68.0]	8/12 [39.1, 86.2]
CR (Ours)	10/12 [55.2, 95.3]	20/24 [64.1, 93.3]	7/12 [32.0, 80.7]	9/12 [46.8, 91.1]