TacitVLA: Reasoning Vision-Language-Action Models Without Reasoning Latency

Anonymous authors
Paper under double-blind review

Abstract

Textual reasoning before acting has improved vision-language-action (VLA) models, but at the cost of latency: every action conditioned on the reasoning text must wait for autoregressive decoding. We show that actions do not need to wait for or condition on the generated reasoning to benefit from it, because the reasoning supervision already improves the backbone representation. We introduce *TacitVLA*, a reasoning VLA whose action expert never attends to the reasoning it generates for the current call. On long-horizon tasks where the reasoning text also serves as episodic memory, it can generate each entry during action execution rather than before acting. We evaluate it in a controlled study on four benchmarks spanning manipulation and autonomous driving, memory and non-memory tasks: RoboMME, RMBench, LIBERO, and PhysicalAI-AV. On all four, *TacitVLA* matches the performance of its explicit-reasoning counterpart at up to 3.4× lower latency, and matches the latency of the same architecture trained without reasoning at up to +17.1% success. In absolute terms, it surpasses the best previously published results on the RoboMME Counting suite (82.7% vs. 66.8%) and RMBench (77.5% vs. 56.5%), and reaches 72.9% on LIBERO-Plus and a 52.5% scene score on the public split of the PhysicalAI-AV closed-loop driving challenge.

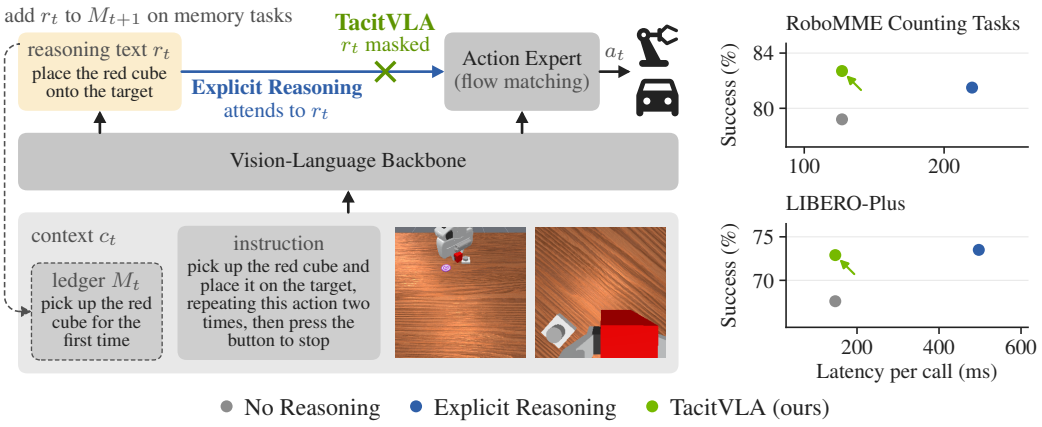


Figure 1: **TacitVLA retains the benefit of reasoning supervision without conditioning its actions on the generated text.** Left: how the policies act in one call. Given the context c_t, on memory tasks including the ledger M_t of past subtasks, the backbone generates the reasoning r_t and conditions the action expert that outputs the action chunk a_t. In Explicit Reasoning, the action expert attends to r_t, so every action chunk waits for it. TacitVLA trains on the same sequence with r_t masked from the action expert, bypassing the need to generate r_t on non-memory tasks. On memory tasks, r_t still becomes the next ledger entry (dashed loop), but is generated asynchronously during execution instead of making the action expert wait. No Reasoning is the same architecture trained without reasoning supervision. Right: TacitVLA matches Explicit Reasoning's success at 1.7× and 3.4× lower latency, and matches No Reasoning's latency at +3.5% and +5.3% success (Tables 3, 5, 15).

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 that generate a scene description, a plan, or the next subtask before acting have outperformed models that map observations directly to actions (Zawalski et al., 2024; Black et al., 2025a; Yin et al., 2026; NVIDIA et al., 2025b; Yang et al., 2026c). The generated text plays three roles in such a policy. (i) During training, it is a target, and the supervision shapes the backbone representation on which the action expert conditions. (ii) On long-horizon tasks, it can serve as episodic memory: each generated subtask is appended to the context, recording what the current observation no longer shows, such as how many button presses have been made. (iii) At deployment, it is an input: the action expert conditions on the reasoning text generated for the current call, and because the reasoning text is decoded autoregressively, the action must wait for that decoding to finish. For the reasoning VLAs we train on LIBERO and PhysicalAI-AV, generating the reasoning text accounts for 53% to 70% of the latency per call (Section 6.2).

To address the latency problem, dual-system designs run the vision-language model at a lower rate than a separate controller (Zhang et al., 2024; Huang et al., 2025; Figure AI, 2025), which either does not eliminate the wait or makes the controller act on what that model produced for an earlier observation. Emerging work trains reasoning VLAs on reasoning text and deploys them without generating it (Li et al., 2026a; Sun et al., 2026a), yet does not answer whether that comes at a cost to performance and does not consider tasks where the generated text must be retained as memory. We therefore ask whether role (iii) is truly necessary:

Must the action condition on the current reasoning, or is learning to generate it enough?

If the benefit of reasoning mainly comes through role (i), generating the reasoning text adds latency without improving performance. Where the reasoning text also serves as memory (ii), it must still be generated for later calls, but this can happen during action execution rather than before (Figure 1).

Our controlled study on four benchmarks shows that learning to generate the reasoning is enough. On each, we train three policies of the architecture recent reasoning VLAs adopt (Black et al., 2025a; NVIDIA et al., 2025b; Yang et al., 2026c; Lin et al., 2026a; Li et al., 2026a; Sun et al., 2026a): a vision-language backbone that generates the reasoning and feeds a flow-matching action expert. *Explicit Reasoning* deploys as embodied chain-of-thought does (Zawalski et al., 2024): its action expert attends to the reasoning it has just generated, making every action chunk wait. We introduce *TacitVLA*, which is trained on the same sequence and differs only in masking the action expert's attention to the reasoning tokens (Figure 2), so its actions do not wait for the reasoning text. *No Reasoning* is the same architecture trained without reasoning supervision, which matches *TacitVLA*'s latency and quantifies the benefit of that supervision. The benchmarks split by how the reasoning text is used: on the *memory tasks*, RoboMME and RMBench, each generated subtask becomes the next entry of a ledger that every later call conditions on. On the *non-memory tasks*, LIBERO and PhysicalAI-AV, the reasoning text serves the current action only and is discarded. Our contributions are:

- **Controlled study: an action expert that never conditions on the current reasoning matches one that does.** We train Explicit Reasoning and *TacitVLA* on the same sequences and vary only the action expert's attention to the reasoning tokens. We evaluate on four benchmarks spanning manipulation and autonomous driving, memory and non-memory tasks. On all four, *TacitVLA* matches the performance of Explicit Reasoning, confirming that learning to generate the reasoning is enough.
- **TacitVLA: Explicit Reasoning's success at No Reasoning's latency.** *TacitVLA*'s latency is up to 1.7× lower than Explicit Reasoning's on the memory tasks and 3.4× lower on the non-memory tasks (Section 6.2). While matching No Reasoning's latency, it improves success by up to +17.1% (Section 6.1). In absolute terms, it surpasses the best published results on RoboMME (48.1% vs. 44.5%; Dai et al., 2026) and RMBench (77.5% vs. 56.5%; Yang et al., 2026c), and reaches 72.9% on LIBERO-Plus and a 52.5% scene score on PhysicalAI-AV (Section 6.1).

2 RELATED WORK

Reasoning before acting and its cost. Embodied chain-of-thought trains a VLA to describe its scene, plan, and next subtask before acting and shows that the reasoning improves generalization (Zawalski et al., 2024). The pattern now runs through manipulation, driving, and hierarchical policies

(Lin et al., 2026a; Yang et al., 2026c; NVIDIA et al., 2025b; Black et al., 2025a; Gao et al., 2025). The cost is autoregressive generation on the action path: embodied chain-of-thought policies run at 0.3 to 0.5 Hz compared with 3 to 4 Hz for the same architecture without test-time reasoning on the same GPU (Chen et al., 2025b), and hybrid action decoding narrows the gap to 1.4× (Yin et al., 2026). Other methods that reduce this latency include the reuse of a potentially stale chain across calls (Duan et al., 2026), latent reasoning in place of tokens (Huang et al., 2026; Bai et al., 2026; Tan et al., 2026), and, for VLA inference in general, early exit, token caching, and state-space backbones (Yue et al., 2024; Xu et al., 2025; Liu et al., 2024). Instead of reducing the latency cost of reasoning, *TacitVLA* removes the need for reasoning generation before action prediction while keeping reasoning as a training target, eliminating the reasoning latency.

Reasoning as a training target. Several prior works train on reasoning text and deploy without generating it: implicit planning in VLA-OS (Gao et al., 2025), $\pi_{0.5}$ deployed without subtask inference (Black et al., 2025a), the reasoning pre-training and dropout recipes of ECoT-Lite (Chen et al., 2025b), ZR-0 (Li et al., 2026a), and ERVLA (Sun et al., 2026a). They establish that reasoning supervision can help a policy that generates no text at deployment. However, they either do not compare with an explicit-reasoning counterpart (Li et al., 2026a), or the comparison also varies how actions are decoded (Sun et al., 2026a) or the training mixture (Chen et al., 2025b), switches generation off on a checkpoint trained to condition on it (Black et al., 2025a), or uses a small policy trained from scratch (Gao et al., 2025). We hold the sequence, data, prompts, and optimizer fixed, vary the action expert's attention alone, extend the comparison to memory tasks, where the reasoning text must still be generated, and measure latency beside success. Fast-WAM (Yuan et al., 2026) asks the analogous question for future video and suggests that video prediction helps mainly by improving representations during training rather than by generating the future at test time. In language modeling, rationales used only as training targets improve answers (Hsieh et al., 2023), and chains of thought can be internalized (Deng et al., 2023; Hao et al., 2025; Kim et al., 2026).

Memory and asynchronous reasoning. Memory benchmarks (Dai et al., 2026; Chen et al., 2026) require information that the current observation no longer shows. Some policies keep it as text: WLA's subtask ledger, which its action waits for (Yang et al., 2026c), and OneTwoVLA's latest reasoning (Lin et al., 2026a). Memory can also be represented as latent tokens (Shi et al., 2026) or recent frames plus compressed gist tokens (Yang et al., 2026b). Hierarchical and dual-system designs run the vision-language model below the control rate, with a fast controller conditioning on what the slow model has produced (Zhang et al., 2024; Chen et al., 2025a; Duan et al., 2026). Real-time chunking keeps actions continuous while the next action chunk is computed (Black et al., 2025c). These designs change the schedule or the memory format. *TacitVLA* keeps one backbone feeding one action expert, holds the ledger protocol fixed, and moves the generation of each entry into the execution window to eliminate its latency at deployment.

3 Preliminaries

3.1 Reasoning VLAs

A reasoning VLA is trained to generate reasoning text before acting (Zawalski et al., 2024; Black et al., 2025a; NVIDIA et al., 2025b; Yang et al., 2026c). We study the form adopted by recent reasoning VLAs: a vision-language backbone generates the reasoning text and an action expert conditions on the backbone's representation (Black et al., 2025a; NVIDIA et al., 2025b; Yang et al., 2026c; Lin et al., 2026a; Li et al., 2026a; Sun et al., 2026a). A *call* of the policy maps the context to one action chunk, part or all of which is executed before the next call. The context c_t at call t holds the instruction and the observation: camera views, robot or vehicle state, and the history the policy keeps (Appendix A). On memory tasks, it also holds a ledger M_t of past subtasks. From it, the backbone generates the reasoning text r_t, which is the next subtask, a scene description and plan, or a chain of causation, depending on the benchmark (Appendices A and D).

The action expert generates an action chunk a_t (Zhao et al., 2023) by denoising (Chi et al., 2023), with flow matching in K Euler steps (Black et al., 2025b). The backbone and the action expert form a Mixture-of-Transformers (Liang et al., 2025): they keep separate weights but share one attention, in which the action positions attend to the backbone's keys and values layer by layer. Both train

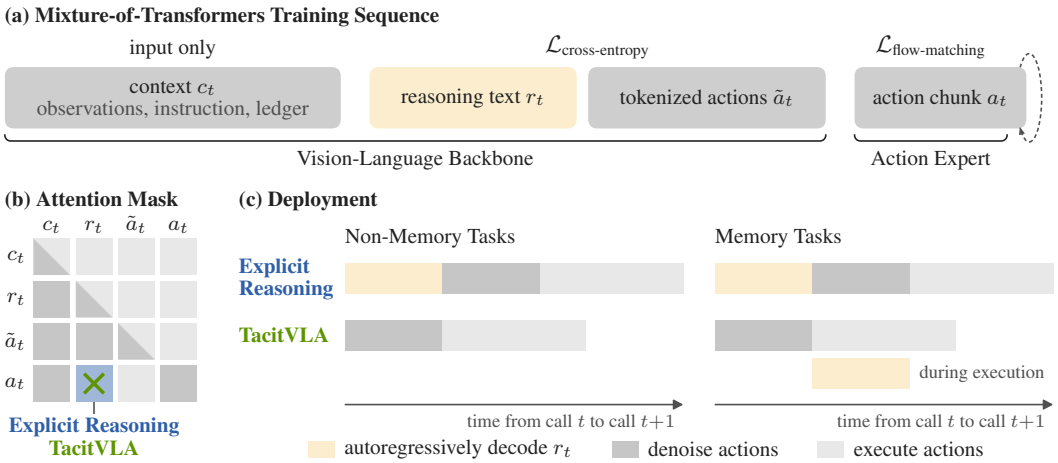


Figure 2: **Training and deployment of Explicit Reasoning and TacitVLA.** The two policies share the training sequence and differ in one attention-mask cell, which decouples TacitVLA's actions from text generation at deployment. (a) The backbone predicts the reasoning r_t and the tokenized actions $\tilde{a}_t$ (an auxiliary target) under cross-entropy, and the action expert denoises the action chunk a_t under flow matching. (b) Rows attend to columns. The backbone is causal. The action expert attends to c_t and its own action chunk, to r_t only in Explicit Reasoning, and never to $\tilde{a}_t$. The crossed cell is that difference. (c) On non-memory tasks, Explicit Reasoning needs to generate r_t before denoising and executing the action chunk, while TacitVLA does not. On memory tasks, both generate the ledger entry r_t: Explicit Reasoning before action execution, TacitVLA during it concurrently.

jointly on the same examples,

$$\mathcal{L} \;=\; \mathcal{L}_{\text{flow-matching}}(a_t) \;+\; \lambda\, \mathcal{L}_{\text{cross-entropy}}(r_t, \tilde{a}_t), \tag{1}$$

where the flow-matching loss is the action expert's denoising objective on the noised action chunk (Black et al., 2025b), and the cross-entropy is the backbone's next-token loss on the reasoning text and on $\tilde{a}_t$. Here, $\tilde{a}_t$ is the same action chunk in tokenized form, which the backbone predicts after r_t: FAST+ tokens (Pertsch et al., 2025), or trajectory tokens in driving (NVIDIA et al., 2025b). Training the backbone on tokenized actions alongside a flow-matching action expert that alone produces the deployed action follows $\pi_{0.5}$, and the tokenized actions speed up training (Pertsch et al., 2025; Black et al., 2025a).

At deployment, such a policy can generate r_t and then denoise the action from the backbone's representation over c_t and r_t, so every action waits for the reasoning text. This follows the deployment pattern of embodied chain-of-thought (Zawalski et al., 2024), and we call it *Explicit Reasoning*. The same architecture trained without reasoning supervision generates no text, and we call it *No Reasoning*.

3.2 Memory and Non-Memory Tasks

The reasoning text is used in two ways. On *memory tasks*, it serves the immediate action and the calls after it: the generated subtask r_t is appended to the ledger, so M_{t+1} is M_t followed by r_t, and every later call conditions on it. The ledger is a symbolic memory of what the current observation does not show, and an entry names the subtask the action chunk attempted. On RMBench, we follow the form of WLA's subtask ledger (Yang et al., 2026c), and on RoboMME, the entries are the benchmark's own simple subgoal annotations (Dai et al., 2026). On *non-memory tasks*, the reasoning text serves the immediate action only: r_t describes the current scene and the action to take, and later calls do not condition on it, as in the LIBERO annotations of DeepThinkVLA (Yin et al., 2026) and the chain of causation in driving (NVIDIA et al., 2025b).

3.3 Latency of a Call

Three stage times describe one call. T_{action} is the action path: visual encoding, prompt prefill, and denoising, the same three stages for every policy we train. T_{reason} is the cost of explicit reasoning: generating r_t and, where the action expert conditions on it, the backbone's pass over the generated

tokens. T_{exec} is the execution window: the physical time the committed actions take to execute before the next call is due, as in real-world deployment (Appendix C). The *latency* of a call is the time from the observation until the action chunk is ready. For No Reasoning, it is T_{action}, whereas for Explicit Reasoning, it is $T_{\text{action}} + T_{\text{reason}}$ on memory and non-memory tasks alike.

4 TacitVLA

4.1 Masking reasoning from the action expert

The backbone's input for one training example is the context c_t followed by the reasoning target r_t and the tokenized actions $\tilde{a}_t$ (Section 3.1), with the action chunk's placeholder positions appended for the action expert (Figure 2). To make a controlled comparison, Explicit Reasoning and *TacitVLA* share this sequence, the loss in Equation 1, the prompts, the data, and the optimizer. They differ only in the attention pattern of the action positions:

$$\text{Explicit:} \quad \pi\big(a_t \mid c_t, r_t\big), \qquad \text{the action expert attends to } c_t \text{ and } r_t; \tag{2}$$

$$\text{TacitVLA:} \quad \pi\big(a_t \mid c_t\big), \qquad \text{the action expert attends to } c_t \text{ only.} \tag{3}$$

Reasoning can thus reach the action through two pathways: the cross-entropy on the reasoning text trains the backbone whose representation the action expert conditions on, and the action expert can attend to r_t itself. Explicit Reasoning has both, *TacitVLA* only the first, and No Reasoning neither, so comparing *TacitVLA* with No Reasoning isolates the first pathway and comparing it with Explicit Reasoning isolates the second. At deployment, Explicit Reasoning's r_t is generated from c_t alone, so the second pathway adds computation, not information. On the tasks we study, *TacitVLA* matches Explicit Reasoning (Section 6.1), so the gain from reasoning does not need the extra computation. Both patterns are masks over one shared sequence, so the sequence shape is identical, and *TacitVLA* uses slightly less training compute per update because its action positions do not attend to r_t.

4.2 Deferring text generation

On non-memory tasks, nothing at deployment consumes r_t, so *TacitVLA* does not generate it. Its latency is T_{action}, the same as No Reasoning's. On memory tasks, *TacitVLA* computes the action first and generates the ledger entry afterwards. At call t, the action expert acts from c_t, which already contains M_t. While the robot executes this action chunk, the backbone generates r_t, which enters the ledger for call $t+1$ exactly as under Explicit Reasoning. Generation delays the next call only if it takes longer than the execution window, and then only by the difference. *TacitVLA*'s latency on memory tasks is therefore

$$T_{\text{action}} + \max(0,\, T_{\text{reason}} - T_{\text{exec}}). \tag{4}$$

Section 6.2 reports the measured T_{action} and T_{reason} for No Reasoning, Explicit Reasoning, and *TacitVLA* on all benchmarks.

5 Experimental setup

Benchmarks. We run the comparison between No Reasoning, Explicit Reasoning, and *TacitVLA* on four benchmarks, two memory and two non-memory (Table 1). On each, the three policies differ only in the attention mask, the reasoning supervision, and, on memory tasks, the ledger. On the memory tasks, No Reasoning thus measures the joint value of reasoning supervision and the ledger.

RoboMME. RoboMME is a 16-task memory benchmark (Dai et al., 2026). Its four counting tasks require counting repeated actions or events. Its authors find perceptual and symbolic memory complementary and suggest combining them, which we do in one backbone. Every policy carries the benchmark's perceptual memory, FrameSamp with its modulator, on all 16 tasks. The reasoning policies add the ledger on the counting tasks, trained on the benchmark's simple subgoal annotations (Appendix A). We train policies on the four counting tasks and on all 16 tasks.

RMBench. RMBench (Chen et al., 2026) tests memory over long manipulation episodes. We use the four of its nine tasks that WLA-0 reports, 100 cases each: Blocks Ranking Try, Battery Try, Press Button, and Cover Blocks.

Table 1: **The four benchmarks.** Each row is one architecture, trained three times on the same data: as No Reasoning, Explicit Reasoning, and TacitVLA. "Reasoning text" is what the backbone generates (Appendix A). Size is TacitVLA's total parameter count, action expert included; on the non-memory tasks, Explicit Reasoning needs 0.05B more to generate text (Table 7).

Benchmark	Task type	Reasoning text	Backbone	Size
RoboMME	memory	next subtask	PaliGemma ($\pi_{0.5}$)	3.7B
RMBench	memory	next subtask	RynnBrain-2B	2.6B
LIBERO (+Plus)	non-memory	scene description and plan	RynnBrain-2B	2.5B
PhysicalAI-AV	non-memory	chain of causation	Qwen3-VL-2B	3.9B

LIBERO. LIBERO is the standard manipulation benchmark of four 10-task suites (Liu et al., 2023), for which we use DeepThinkVLA's reasoning annotations (Yin et al., 2026). To test robustness out of distribution, we also evaluate LIBERO-Plus (Fei et al., 2026), which applies seven perturbation axes zero-shot over 10,030 cases.

PhysicalAI-AV. We train on a proprietary corpus of 3.8 million driving events with chain-of-causation annotations in the style of Alpamayo-R1 (NVIDIA et al., 2025b). The policy predicts a 6.4 s trajectory. We evaluate in open loop on a curated set of 1,434 held-out segments of the public PhysicalAI-AV release (NVIDIA, 2025) and in closed loop in AlpaSim (NVIDIA et al., 2025a), on the 441-scene public split of its driving challenge (NVIDIA, 2026; Appendix A).

Training and evaluation. On the three robot benchmarks, we train only on each benchmark's released demonstrations and compare with published results of other policies trained on them. Our policies are evaluated under each benchmark's official protocol and cases, and their success values are means $\pm$ standard deviation over three evaluation seeds. Closed-loop driving values are means over three rollouts per scene. All training runs on nodes of eight H100 GPUs, and all three policies run the same inference implementation, timed on one H100 (Appendix C). Initializations, architectures, training settings, prompts, decoding, the ledger protocol, and the source of every published row are in Appendix A.

6 Results

6.1 Performance

RoboMME. Trained on the Counting suite alone and evaluated on its four tasks (200 cases, Table 2), *TacitVLA* reaches 89.5 and Explicit Reasoning 86.7, compared with 79.0 for No Reasoning. *TacitVLA*'s 89.5 is above even the benchmark's human reference of 88.5 (Dai et al., 2026). In that reference, people only chose each high-level action and an oracle planner executed it without error, whereas *TacitVLA* must also execute every motion. Reasoning supervision adds 10.5 points for *TacitVLA* and 7.7 for Explicit Reasoning, so an entry generated during action execution retains the full gain of one generated before the action. *TacitVLA*'s ledger is also as accurate: the final ledger matches the instructed count within two points of Explicit Reasoning's on each counting task in Table 14.

Table 2: **RoboMME Counting.** Success (%) on the benchmark's four counting tasks (200 cases) of policies trained on them alone, and the benchmark's human reference (Dai et al., 2026). Shaded rows are ours, mean $\pm$ standard deviation over three evaluation seeds.

Policy	Success
Human (oracle planner)	88.5
No Reasoning	79.0 $\pm$ 1.0
Explicit Reasoning	86.7 $\pm$ 2.0
TacitVLA	**89.5** $\pm$ 1.3

Trained on all 16 tasks (Table 3), No Reasoning, Explicit Reasoning, and *TacitVLA* score 46.5, 45.8, and 48.1, respectively, compared with 44.5 for FrameSamp + Modulator (Dai et al., 2026), the benchmark's best published result. On the counting tasks, *TacitVLA*'s 82.7 is within 1.2 points of the oracle-subgoal rows (83.9 and 82.6; Dai et al., 2026), which receive the simulator's ground-truth subgoals, and above the trained symbolic systems (38.0 to 48.8), which run a second vision-language model to generate their subgoals. Additionally, we sweep the cross-entropy weight over three values in both training setups (Table 11). *TacitVLA* scores above No Reasoning in all six settings and is the best of the three policies in five, within 1.2 points of Explicit Reasoning in the sixth.

Table 3: **RoboMME.** Success (%) of policies trained on all 16 tasks, on each suite (200 cases) and on all tasks (800 cases). Shaded rows are ours, mean ± standard deviation over three evaluation seeds. Only the counting tasks have reasoning supervision. Published rows are from the RoboMME paper (Dai et al., 2026), except WeaveLA and AGM, which report their own (Appendix A).

Policy	Counting	Permanence	Reference	Imitation	All tasks
$\pi_{0.5}$ (Black et al., 2025a)	28.8	17.0	17.2	8.8	17.9
$\pi_{0.5}$ with past actions	29.1	22.8	15.9	11.2	19.7
SAM2Act+ (Fang et al., 2025)	35.3	26.0	16.8	7.3	21.4
SimpleSG + Qwen3-VL-4B	44.6	19.6	25.2	26.6	29.0
GroundSG + Qwen3-VL-4B	38.0	39.3	31.6	21.9	32.7
MemER (Sridhar et al., 2025)	48.8	**53.2**	**38.0**	29.5	42.4
FrameSamp + Expert	66.8	25.2	24.1	28.9	36.3
FrameSamp + Modulator	65.2	25.1	36.3	**51.4**	44.5
WeaveLA (Zhu et al., 2026)	31.0	19.5	12.5	9.5	18.1
$\pi_{0.5}$ + AGM (Gao et al., 2026)	56.0	–	–	–	–
No Reasoning	79.2 ± 1.0	23.8 ± 2.6	35.2 ± 2.0	48.0 ± 1.0	46.5 ± 0.9
Explicit Reasoning	81.5 ± 1.8	26.8 ± 0.6	27.7 ± 2.0	47.0 ± 2.6	45.8 ± 1.5
TacitVLA	**82.7** ± 3.8	23.8 ± 3.1	37.8 ± 2.9	48.0 ± 1.5	**48.1** ± 1.0

Table 4: **RMBench.** Success (%) on each task (100 cases) and the four-task mean. Shaded rows are ours, mean ± standard deviation over three evaluation seeds. Published rows are from the benchmark paper (Chen et al., 2026), EventVLA (Yang et al., 2026a), and the systems' own papers (Appendix A).

Policy	Blocks Ranking Try	Battery Try	Press Button	Cover Blocks	Four-task mean
DP (Chi et al., 2023)	10	10	0	0	5.0
ACT (Zhao et al., 2023)	0	19	0	0	4.8
$\pi_{0.5}$ (Black et al., 2025a)	6	16	0	0	5.5
X-VLA (Zheng et al., 2026)	1	26	0	2	7.3
WLA-0 (Yang et al., 2026c)	23	**45**	74	84	56.5
HiMem-WAM (Sun et al., 2026b)	24	28	8	19	19.8
QwenOFT	37	14	0	0	12.8
MemER (Sridhar et al., 2025)	0	27	0	6	8.3
MemoryVLA (QwenOFT; Shi et al., 2026)	53	33	0	69	38.8
EventVLA (Yang et al., 2026a)	81	35	3	97	54.0
No Reasoning	94.3 ± 2.5	42.0 ± 1.7	7.7 ± 2.1	97.7 ± 2.1	60.4 ± 1.5
Explicit Reasoning	94.3 ± 1.2	42.3 ± 1.5	73.3 ± 2.5	**98.0** ± 1.0	77.0 ± 1.9
TacitVLA	**94.7** ± 1.5	43.0 ± 2.6	**74.3** ± 2.5	**98.0** ± 1.0	**77.5** ± 1.1

RMBench. On RMBench (Table 4), the three policies are within a point of one another on Blocks Ranking Try, Battery Try, and Cover Blocks. On Press Button, whose count exists only in the ledger, Explicit Reasoning and *TacitVLA* score 73.3 and 74.3, respectively, compared with 7.7 for No Reasoning. Both keep a correct ledger in 99.7% of episodes (Table 14). The four-task means are 60.4, 77.0, and 77.5 for No Reasoning, Explicit Reasoning, and *TacitVLA*, respectively. All three are above the four-task mean of WLA-0 (56.5; Yang et al., 2026c).

LIBERO. On LIBERO-Plus, reasoning supervision lifts success from 67.6 for No Reasoning to 73.5 for Explicit Reasoning (Table 5), and *TacitVLA* scores 72.9, within a point of Explicit Reasoning and 5.3 points above No Reasoning. On the four standard suites, which are close to saturation, No Reasoning, Explicit Reasoning, and *TacitVLA* score 98.5, 98.1, and 98.6, respectively, within a point of one another and of WLA-0 (98.6; Yang et al., 2026c). By perturbation axis, the gain over No Reasoning is largest on changed robot initial states, 11.4 points for Explicit Reasoning and 10.2 for *TacitVLA* (Table 12).

Compared with published policies trained, like ours, on the LIBERO demonstrations alone (Table 5), the two reasoning policies hold the highest LIBERO-Plus totals. Policies of our backbone class trained without reasoning supervision score close to No Reasoning (67.6): Qwen3OFT, LoopVLA's baseline of Qwen3-VL-2B with a direct action MLP, scores 68.8 (Shen et al., 2026), and LIT's $\pi_{0.5}$ baseline, trained from the vision-language backbone, scores 69.0 (Lin et al., 2026b). No Reasoning

Table 5: **LIBERO and LIBERO-Plus.** Success (%) on LIBERO-Plus (10,030 cases) and the four standard suites (2,000 cases; per suite in Table 13). Shaded rows are ours, mean ± standard deviation over three evaluation seeds. Published rows are zero-shot LIBERO-Plus results of policies trained, like ours, on the LIBERO demonstrations alone, except the robot-action-pretrained MolmoAct (†). The $\pi_{0.5}$ and MolmoAct2 rows are LIT's baselines (Lin et al., 2026b), trained this way from the vision-language backbones. The LIBERO-Plus values of DreamVLA, MolmoAct, and VLA-Adapter are evaluations by Lin et al. (2026d), Dalal et al. (2026), and He et al. (2026), respectively.

Policy	Plus	Standard
DreamVLA (Zhang et al., 2025)	48.9	92.6
MolmoAct† (Lee et al., 2025)	60.8	86.6
VLA-Adapter (Wang et al., 2025)	59.1	97.3
MergeVLA (Fu et al., 2026)	72.4	96.7
LoopOFT$_\alpha$ (Shen et al., 2026)	65.8	95.3
Evo-Depth (Lin et al., 2026c)	69.6	95.4
WLA-0 (Yang et al., 2026c)	–	**98.6**
$\pi_{0.5}$ (Black et al., 2025a)	69.0	87.8
MolmoAct2 (Fang et al., 2026)	63.6	93.5
No Reasoning	67.6 ± 0.4	98.5 ± 0.3
Explicit Reasoning	**73.5** ± 0.3	98.1 ± 0.4
TacitVLA	72.9 ± 0.3	**98.6** ± 0.3

is thus a competent policy of its class, and reasoning supervision is what lifts it: *TacitVLA* generates no text here, so the lift reaches its action through the representation alone (Section 4.1).

PhysicalAI-AV. The three policies are trained on the driving data and evaluated on scenes disjoint from it. In open loop, both reasoning policies lower the displacement error below No Reasoning's 1.166 m, to 1.140 m for Explicit Reasoning and 1.116 m for *TacitVLA* (Table 6). In closed loop, *TacitVLA* reaches a scene score of 52.5% and Explicit Reasoning 52.3%, compared with 46.4% for No Reasoning. Both reasoning policies pass more scenes and cause fewer at-fault collisions. On the organizers' published reference rollouts for the same scenes and protocol (NVIDIA, 2026), Alpamayo-R1 (NVIDIA et al., 2025b), a 10B model with four cameras, scores 55.0%. VaVAM (Bartoccioni et al., 2025) scores 11.0% under the competition controller and 31.4% under a linear controller that is not available in the competition, and five unattributed submissions score 5.8% to 16.8%. *TacitVLA* comes within 2.5 points of Alpamayo-R1 with fewer than half its parameters and half its cameras. Appendix D shows, for every benchmark, a successful and a failed episode of both reasoning policies with the reasoning text they generate: in these examples, no failure mode is clearly specific to *TacitVLA*.

Table 6: **PhysicalAI-AV.** Open loop: minADE (m), the minimum average displacement error over six samples on the 1,434 held-out segments of the public PhysicalAI-AV release, computed from its released videos, mean ± standard deviation over evaluation seeds. Closed loop: AlpaSim's metrics on the 441-scene public split of its challenge with one sampled trajectory per call, mean ± standard deviation over three rollouts per scene. Infractions are at-fault collisions and off-road events. Shaded rows are ours. VaVAM (Bartoccioni et al., 2025): the organizers' reference runs (NVIDIA, 2026) under the competition's nonlinear controller, which our policies also use, and under a linear controller that is not available in the competition.

Policy	minADE ↓	Pass rate (%) ↑	Collision any (%) ↓	At-fault collision (%) ↓	Off-road (%) ↓	km per infraction ↑	Scene score (%) ↑
VaVAM (nonlinear)	–	15.0 ± 0.7	24.3 ± 0.6	18.0 ± 0.6	28.0 ± 1.1	0.14 ± 0.00	11.0 ± 0.7
VaVAM (linear)	–	42.1 ± 0.5	33.3 ± 0.3	16.4 ± 0.7	15.2 ± 0.8	0.35 ± 0.00	31.4 ± 0.6
No Reasoning	1.166 ± 0.010	50.1 ± 2.2	16.1 ± 1.3	6.9 ± 1.6	9.5 ± 0.5	0.87 ± 0.08	46.4 ± 2.2
Explicit Reasoning	1.140 ± 0.015	**56.4** ± 1.5	16.9 ± 0.7	**5.5** ± 0.5	**9.0** ± 0.1	**1.01** ± 0.03	52.3 ± 1.4
TacitVLA	**1.116** ± 0.011	55.8 ± 0.0	**15.6** ± 0.2	6.7 ± 0.3	10.0 ± 0.9	0.87 ± 0.08	**52.5** ± 0.3

6.2 Latency

We time every model stage with per-call timers during evaluation episodes (Appendix C). The three policies differ only in the stages they execute (Figure 3, Table 15). On the memory tasks, the deferred generation finished within the execution window in every call, so generation adds nothing to *TacitVLA*'s latency (Equation 4). The success rates of Section 6.1 hold unchanged under this schedule: *TacitVLA*'s action never conditions on r_t, and the entry enters the ledger at the same boundary under both policies. Deferring generation therefore changes only when the entry is generated.

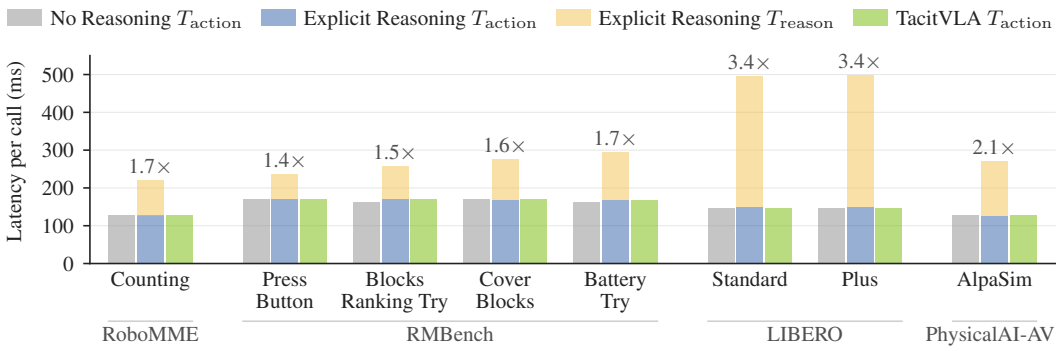


Figure 3: **TacitVLA matches the latency of No Reasoning on every benchmark, and Explicit Reasoning's latency is 1.4× to 3.4× higher.** Each bar is one policy's latency per call: T_{action} for No Reasoning and TacitVLA, and $T_{\text{action}} + T_{\text{reason}}$ for Explicit Reasoning, whose actions wait for text generation. The ratio above each group is Explicit Reasoning's latency over TacitVLA's. On the memory tasks, TacitVLA generates the ledger entry during execution, taking at most 12% of the execution window, so the next call never waits for generation. All values are means over model calls on one H100, with the per-stage breakdown in Table 15 (Appendix C).

On the RoboMME counting tasks, the action path takes $T_{\text{action}} = 127$ ms and generating the ledger entry takes $T_{\text{reason}} = 93$ ms under both reasoning policies. *TacitVLA* generates the entry during execution, in 12% of the 800 ms window (16 actions at 20 Hz), so its action is ready after 127 ms. Explicit Reasoning runs the two stages in series, and its action is ready after 220 ms, 1.7× later. On RMBench, generating the next subtask entry takes 66 to 120 ms, at most 1.1% of the 7.0 to 11.2 s window in which the motion planner executes each plan (Appendix C). *TacitVLA*'s action is ready after 169 to 171 ms and Explicit Reasoning's after 236 to 294 ms, 1.4× to 1.7× later.

On the non-memory tasks, *TacitVLA* generates no text at deployment (Section 4.2), so its latency is No Reasoning's: 147 ms per call on LIBERO and LIBERO-Plus and 127 ms in closed-loop driving. Explicit Reasoning's action is ready after 495 ms on LIBERO and 269 ms in driving, 3.4× and 2.1× later, respectively. The difference is the generation of reasoning text the next call never uses. LIBERO's window is 250 ms (5 actions at 20 Hz), which *TacitVLA* meets and Explicit Reasoning does not.

7 Conclusion

On the four benchmarks we study, spanning manipulation and autonomous driving, memory and non-memory tasks, an action expert that never conditions on the current reasoning matches one that does: the benefit of reasoning reaches the action through the reasoning supervision and, on memory tasks, through the ledger of earlier entries in the action expert's context. *TacitVLA* thus delivers Explicit Reasoning's success at No Reasoning's latency. On the memory tasks, it generates the ledger entry within the execution window at every call. The design remains one backbone feeding one action expert, with no second model or scheduler added. Any reasoning VLA of this form can therefore be trained as *TacitVLA* by masking the reasoning tokens from the action expert, and deployed with generation omitted or, when the reasoning text serves as memory, run during action execution. Upon publication, we will release our code as open source, with weights that reproduce our success rates and latencies.

Limitations and future work. As for other reasoning VLAs, *TacitVLA*'s performance depends on the quality of the reasoning data. On non-memory tasks, it can still generate the reasoning text during action execution (Figure 2), which could serve interpretability in safety-critical settings such as autonomous driving, a use we leave to future work. Fast-WAM (Yuan et al., 2026) reaches an analogous conclusion for future video: a world-action model that retains video co-training but skips future prediction at test time remains competitive with imagine-then-execute variants. Combining the two suggests a backbone with three experts for reasoning, future video, and action, trained on all three targets, yet acting efficiently without first generating reasoning or future video. On memory tasks, the deferred entry adds no latency only while it fits in the execution window. It takes at most 12% of the window in our benchmarks, and longer entries or shorter windows would add the time by which generation exceeds the window (Equation 4).

AI USE STATEMENT

In this work, we used generative AI tools to implement methods and, at times, to interpret results: we wrote the code together with these tools under our direction, checked and tested all generated code, and reviewed every suggested result interpretation before incorporating it. To ensure the breadth of our literature search, we also used these tools to find related work, and we filtered the results ourselves after reading the papers. Finally, we used them to polish our own writing, such as by improving sentence structure, instead of drafting it directly. We have not used generative AI tools to generate synthetic datasets, to help develop theoretical models or conceptual frameworks, to propose or refine hypotheses, to design or provide feedback on research methodology or experiments, or to clean and reformat datasets. The idea, the choice of benchmarks and data, and the experimental design are our own. Formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, assisting with translation, and supporting qualitative and thematic data analysis are not applicable to this work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our experiments involve no human subjects. The robot benchmarks (LIBERO, RoboMME, RMBench) are public, with released demonstrations and reasoning annotations. The driving policies are evaluated on a held-out set of segments of the public PhysicalAI-AV release and on the public split of the AlpaSim challenge. *TacitVLA* lowers the latency of reasoning VLAs, which helps most in safety-critical settings such as on-vehicle deployment, where the action latency bounds how quickly the policy can react to a sudden event. As with any learned driving policy, deployment would require validation beyond the evaluations reported here.

REPRODUCIBILITY STATEMENT

Upon publication, we will release our code as open source, together with the checkpoints for every experiment and ablation in the paper and the manifest of the driving held-out set, so that every success rate and latency reported here can be reproduced within noise and hardware variation. The sole exception is the training of the driving policies, whose data remains proprietary. However, their trained checkpoints will be released, so their open-loop and closed-loop evaluations remain reproducible. Appendix A specifies every benchmark's setup: backbone initialization and architecture (Tables 7 and 8), training data, optimization (Table 9), the context, the prompts, the action and evaluation protocol (Table 10), the ledger semantics, and the deployment settings. Appendix B gives the ablations and full result tables behind Section 6.1. Appendix C gives the latency protocol, what is timed, and the per-stage times behind Figure 3. Appendix D shows rollouts with the generated reasoning and ledgers, Appendix B reports ledger correctness, and Appendix A reproduces the reasoning targets.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL Technical Report, 2025. URL https://arxiv.org/abs/2511.21631.

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. In *International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2602.01166.

Florent Bartoccioni, Elias Ramzi, Victor Besnier, Shashanka Venkataramanan, Tuan-Hung Vu, Yihong Xu, Loick Chambon, Spyros Gidaris, Serkan Odabas, David Hurych, Renaud Marlet,

Alexandre Boulch, Mickael Chen, Eloi Zablocki, Andrei Bursuc, Eduardo Valle, and Matthieu Cord. VaViM and VaVAM: Autonomous Driving through Video Generative Modeling, 2025. URL https://arxiv.org/abs/2502.15672.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert 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. In *Conference on Robot Learning (CoRL)*, 2025a. URL https://arxiv.org/abs/2504.16054.

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 *Robotics: Science and Systems (RSS)*, 2025b. URL https://arxiv.org/abs/2410.24164.

Kevin Black, Manuel Galliker, and Sergey Levine. Real-Time Execution of Action Chunking Flow Policies. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025c. URL https://arxiv.org/abs/2506.07339.

Hao Chen, Jiaming Liu, Chenyang Gu, Zhuoyang Liu, Renrui Zhang, Xiaoqi Li, Xiao He, Yandong Guo, Chi-Wing Fu, Shanghang Zhang, and Pheng-Ann Heng. Fast-in-Slow: A Dual-System VLA Model Unifying Fast Manipulation within Slow Reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025a. URL https://arxiv.org/abs/2506.01953.

Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, Zixuan Li, Yifan Hu, Yingsheng Zhang, Kaixuan Wang, Yue Chen, Hongcheng Wang, Junjie Wang, Tianhang Yang, Renjing Xu, Ruihai Wu, Yao Mu, Yaodong Yang, Hao Dong, and Ping Luo. RMBench: Memory-Dependent Robotic Manipulation Benchmark with Insights into Policy Design, 2026. URL https://arxiv.org/abs/2603.01229.

William Chen, Suneel Belkhale, Suvir Mirchandani, Karl Pertsch, Danny Driess, Oier Mees, and Sergey Levine. Training Strategies for Efficient Embodied Reasoning. In *Conference on Robot Learning (CoRL)*, 2025b. URL https://arxiv.org/abs/2505.08243.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Robotics: Science and Systems (RSS)*, 2023. URL https://arxiv.org/abs/2303.04137.

Yinpei Dai, Hongze Fu, Jayjun Lee, Yuejiang Liu, Haoran Zhang, Jianing Yang, Chelsea Finn, Nima Fazeli, and Joyce Chai. RoboMME: Benchmarking and Understanding Memory for Robotic Generalist Policies. In *International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2603.04639.

Dwip Dalal, Shivansh Patel, Chahit Jain, Jeonghwan Kim, Utkarsh Mishra, Alex Baratian, Hyeonjeong Ha, Heng Ji, Svetlana Lazebnik, and Unnat Jain. Generalizable VLA Finetuning via Representation Anchoring and Language-Action Alignment, 2026. URL https://arxiv.org/abs/2607.13429.

Ronghao Dang, Jiayan Guo, Bohan Hou, Sicong Leng, Kehan Li, Xin Li, Jiangpin Liu, Yunxuan Mao, Zhikai Wang, Yuqian Yuan, Minghao Zhu, Xiao Lin, Yang Bai, Qian Jiang, Yaxi Zhao, Minghua Zeng, Junlong Gao, Yuming Jiang, Jun Cen, Siteng Huang, Liuyi Wang, Wenqiao Zhang, Chengju Liu, Jianfei Yang, Shijian Lu, and Deli Zhao. RynnBrain: Open Embodied Foundation Models, 2026. URL https://arxiv.org/abs/2602.14979.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit Chain of Thought Reasoning via Knowledge Distillation, 2023. URL https://arxiv.org/abs/2311.01460.

Zhekai Duan, Yuan Zhang, Shikai Geng, Gaowen Liu, Joschka Boedecker, and Chris Xiaoxuan Lu. Fast ECoT: Efficient Embodied Chain-of-Thought via Thoughts Reuse. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026. URL https://arxiv.org/abs/2506.07639.

Haoquan Fang, Markus Grotz, Wilbert Pumacay, Yi Ru Wang, Dieter Fox, Ranjay Krishna, and Jiafei Duan. SAM2Act: Integrating Visual Foundation Model with A Memory Architecture for Robotic Manipulation. In *International Conference on Machine Learning (ICML)*, 2025. URL https://arxiv.org/abs/2501.18564.

Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, Shanli Xing, Jaemin Cho, Jae Sung Park, Ainaz Eftekhar, Peter Sushko, Karen Farley, Angad Wadhwa, Cole Harrison, Winson Han, Ying-Chun Lee, Eli VanderBilt, Rose Hendrix, Suveen Ellawela, Lucas Ngoo, Joyce Chai, Zhongzheng Ren, Ali Farhadi, Dieter Fox, and Ranjay Krishna. MolmoAct2: Action Reasoning Models for Real-world Deployment, 2026. URL https://arxiv.org/abs/2605.02881.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. 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)*, 2026. URL https://arxiv.org/abs/2510.13626.

Figure AI. Helix: A Vision-Language-Action Model for Generalist Humanoid Control. https://www.figure.ai/news/helix, 2025. Technical report, February 2025.

Yuxia Fu, Zhizhen Zhang, Yuqi Zhang, Zijian Wang, Zi Huang, and Yadan Luo. MergeVLA: Cross-Skill Model Merging Toward a Generalist Vision-Language-Action Agent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. URL https://arxiv.org/abs/2511.18810.

Chongkai Gao, Zixuan Liu, Zhenghao Chi, Junshan Huang, Xin Fei, Yiwen Hou, Yuxuan Zhang, Yudi Lin, Zhirui Fang, and Lin Shao. VLA-OS: Structuring and Dissecting Planning Representations and Paradigms in Vision-Language-Action Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2506.17561.

Hongbo Gao, Zeyu Ni, Xin Wen, Siyu Xu, and Ruifeng Li. AGM: Achievement-Grounded Memory for Closed-Loop Agents with Frozen VLA Policies, 2026. URL https://arxiv.org/abs/2608.29537.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training Large Language Models to Reason in a Continuous Latent Space. In *Conference on Language Modeling (COLM)*, 2025. URL https://arxiv.org/abs/2412.06769.

Zeyuan He, Bowen Yang, Zhirui Fang, Keru Zhou, Lei Jiang, Jingjing Qian, Fan Mo, Junchi Yan, Philip Torr, Xiu Li, Li Jiang, and Jialin Yu. ELAN4D: Embodiment-Centric 4D Supervision for Vision-Language-Action Models via Plug-and-Play Adaptation, 2026. URL https://arxiv.org/abs/2605.30484.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling Step-by-Step! Outperforming Larger Language Models with Less Training Data and Smaller Model Sizes. In *Findings of the Association for Computational Linguistics (ACL)*, 2023. URL https://arxiv.org/abs/2305.02301.

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 (NeurIPS)*, 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. URL https://arxiv.org/abs/2601.09708.

Yuhua Jiang, Junjie Lu, Xinyao Qin, Xiaoyu Chen, Kaixin Wang, Feifei Gao, and Li Zhao. VLA-GSE: Boosting Parameter-Efficient Fine-Tuning in VLA with Generalized and Specialized Experts, 2026. URL https://arxiv.org/abs/2605.06175.

Donggyun Kim, Jack Lu, Chanwoo Kim, Mengye Ren, and Seunghoon Hong. HyperThink: Text-to-Parameter Hypernetworks for Efficient Reasoning. In *Conference on Language Modeling (COLM)*, 2026. URL https://openreview.net/forum?id=4O1A1sbam6.

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.

Haoyang Li, Guanlin Li, Youhe Feng, Chen Zhao, Zhuoran Wang, Yang Li, Qizhe Wei, Shifeng Bao, Haitao Shen, Yihan Zhao, Tong Yang, and Jing Zhang. Training Vision-Language-Action Models with Dense Embodied Chain-of-Thought Supervision, 2026a. URL https://arxiv.org/abs/2606.30552.

Wei Li, Peijin Jia, Yuan Ma, Xuefeng Jiang, Titong Jiang, Sheng Sun, Yujian Li, Xin Wen, Han Hong, Zhikang Liu, Bailin Li, and Kun Zhan. FoMoVLA: Bridging Visual Foresight and Motion Guidance for Vision-Language-Action Models, 2026b. URL https://arxiv.org/abs/2607.14739.

Weixin Liang, Lili Yu, Liang Luo, Srini Iyer, Ning Dong, Chunting Zhou, Gargi Ghosh, Mike Lewis, Wen-tau Yih, Luke Zettlemoyer, and Xi Victoria Lin. Mixture-of-Transformers: A Sparse and Scalable Architecture for Multi-Modal Foundation Models. *Transactions on Machine Learning Research (TMLR)*, 2025. URL https://arxiv.org/abs/2411.04996.

Fanqi Lin, Ruiqian Nai, Yingdong Hu, Jiacheng You, Junming Zhao, and Yang Gao. OneTwoVLA: A Unified Vision-Language-Action Model with Adaptive Reasoning. In *International Conference on Learning Representations (ICLR)*, 2026a. URL https://arxiv.org/abs/2505.11917.

Jianman Lin, Shailesh Shailesh, Zhongyi Luo, and Jiafei Duan. Breaking the Vision-Action Shortcut: Latent Interface Training for Generalizable Robotics Foundation Models, 2026b. URL https://arxiv.org/abs/2609.12641.

Tao Lin, Yuxin Du, Jiting Liu, Nuobei Zhu, Yunhe Li, Yuqian Fu, Yinxinyu Chen, Hongyi Cai, Zewei Ye, Bing Cheng, Kai Ye, Yiran Mao, Yilei Zhong, MingKang Dong, Junchi Yan, Gen Li, and Bo Zhao. Evo-Depth: A Lightweight Depth-Enhanced Vision-Language-Action Model, 2026c. URL https://arxiv.org/abs/2605.14950.

Zefu Lin, Rongxu Cui, Junjia Xu, Xiaojuan Jin, Wenling Li, Lue Fan, and Zhaoxiang Zhang. World Pilot: Steering Vision-Language-Action Models with World-Action Priors, 2026d. URL https://arxiv.org/abs/2606.12403.

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 (NeurIPS), Datasets and Benchmarks Track*, 2023. URL https://arxiv.org/abs/2306.03310.

Jiaming Liu, Mengzhen Liu, Zhenyu Wang, Pengju An, Xiaoqi Li, Kaichen Zhou, Senqiao Yang, Renrui Zhang, Yandong Guo, and Shanghang Zhang. RoboMamba: Efficient Vision-Language-Action Model for Robotic Reasoning and Manipulation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2406.04339.

NVIDIA. PhysicalAI Autonomous Vehicles Dataset. Hugging Face dataset, 2025. URL https://huggingface.co/datasets/nvidia/PhysicalAI-Autonomous-Vehicles.

NVIDIA. AlpaSim E2E Closed Loop Challenge 2026. https://huggingface.co/spaces/nvidia/AlpasimE2EClosedLoopChallenge2026, 2026. Challenge guide, protocol, and the curated NuRec splits in the `e2e_challenge` branch of https://github.com/NVlabs/alpasim.

NVIDIA, Yulong Cao, Riccardo de Lutio, Sanja Fidler, Guillermo Garcia Cobo, Zan Gojcic, Maximilian Igl, Boris Ivanovic, Peter Karkus, Janick Martinez Esturo, Marco Pavone, Aaron Smith, Ellie Tanimura, Michal Tyszkiewicz, Michael Watson, Qi Wu, and Le Zhang. AlpaSim: A Modular, Lightweight, and Data-Driven Research Simulator for Autonomous Driving, 2025a. URL https://github.com/NVlabs/alpasim.

NVIDIA, Yan Wang, Wenjie Luo, Junjie Bai, Yulong Cao, Tong Che, Ke Chen, Yuxiao Chen, Jenna Diamond, Yifan Ding, Wenhao Ding, Liang Feng, Greg Heinrich, Jack Huang, Peter Karkus, Boyi Li, Pinyi Li, Tsung-Yi Lin, Dongran Liu, Ming-Yu Liu, Langechuan Liu, Zhijian Liu, Jason Lu, Yunxiang Mao, Pavlo Molchanov, Lindsey Pavao, Zhenghao Peng, Mike Ranzinger, Ed Schmerling, Shida Shen, Yunfei Shi, Sarah Tariq, Ran Tian, Tilman Wekel, Xinshuo Weng, Tianjun Xiao, Eric Yang, Xiaodong Yang, Yurong You, Xiaohui Zeng, Wenyuan Zhang, Boris Ivanovic, and Marco Pavone. Alpamayo-R1: Bridging Reasoning and Action Prediction for Generalizable Autonomous Driving in the Long Tail, 2025b. URL https://arxiv.org/abs/2511.00088.

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 *Robotics: Science and Systems (RSS)*, 2025. URL https://arxiv.org/abs/2501.09747.

Boyang Shen, Kaixiang Yang, Hao Wang, Qiuyu Yu, Qiang Xie, Qiang Li, and Zhiwei Wang. LoopVLA: Learning Sufficiency in Recurrent Refinement for Vision-Language-Action Models, 2026. URL https://arxiv.org/abs/2605.09948.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. MemoryVLA: Perceptual-Cognitive Memory in Vision-Language-Action Models for Robotic Manipulation. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2508.19236.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. MemER: Scaling Up Memory for Robot Control via Experience Retrieval, 2025. URL https://arxiv.org/abs/2510.20328.

Nan Sun, Yuan Zhang, Yongkun Yang, Wentao Zhao, Peiyan Li, Jun Guo, Wenxuan Song, Pengxiang Ding, Runze Suo, Yifei Su, Xin Xiao, Xinghang Li, and Huaping Liu. Revisiting Embodied Chain-of-Thought for Generalizable Robot Manipulation, 2026a. URL https://arxiv.org/abs/2606.03784.

Xiaoquan Sun, Ruijian Zhang, Chen Cao, Yihan Sun, Jiahui Chen, Zetian Xu, Bo Chen, Haijier Chen, Zhen Yang, Jiarun Zhu, Yijun Hong, JingZhe Xu, Jingrui Pang, Mingqi Yuan, and Jiayu Chen. HiMem-WAM: Hierarchical Memory-Gated World Action Models for Robotic Manipulation, 2026b. URL https://arxiv.org/abs/2606.10363.

Shuhan Tan, Kashyap Chitta, Yuxiao Chen, Ran Tian, Yurong You, Yan Wang, Wenjie Luo, Yulong Cao, Philipp Krähenbühl, Marco Pavone, and Boris Ivanovic. Latent Chain-of-Thought World Modeling for End-to-End Autonomous Driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. URL https://arxiv.org/abs/2512.10226.

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.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. VLA-Cache: Efficient Vision-Language-Action Manipulation via Adaptive Token Caching. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2502.02175.

Ganlin Yang, Zhangzheng Tu, Yuqiang Yang, Sitong Mao, Junyi Dong, Tianxing Chen, Jiaqi Peng, Jing Xiong, Jiafei Cao, Jifeng Dai, Wengang Zhou, Yao Mu, and Tai Wang. EventVLA: Event-Driven Visual Evidence Memory for Long-Horizon Vision-Language-Action Policies, 2026a. URL https://arxiv.org/abs/2606.20092.

Sizhe Yang, Juncheng Mu, Tianming Wei, Chenhao Lu, Xiaofan Li, Linning Xu, Zhengrong Xue, Zhecheng Yuan, Dahua Lin, Jiangmiao Pang, and Huazhe Xu. MemoryWAM: Efficient World Action Modeling with Persistent Memory, 2026b. URL https://arxiv.org/abs/2606.20562.

Yi Yang, Zhihong Liu, Siqi Kou, Yiyang Chen, Yanzhe Hu, Jianbo Zhou, Boyuan Zhao, Zhijie Wei, Xiao Xia, Xueqi Li, Pengfei Liu, and Zhijie Deng. World-Language-Action Model for Unified World Modeling, Language Reasoning, and Action Synthesis, 2026c. URL https://arxiv.org/abs/2606.05979.

Cheng Yin, Yankai Lin, Wang Xu, Sikyuen Tam, Xiangrui Zeng, Zhiyuan Liu, and Zhouping Yin. DeepThinkVLA: Enhancing Reasoning Capability of Vision-Language-Action Models. In *Conference on Language Modeling (COLM)*, 2026. URL https://arxiv.org/abs/2511.15669.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do World Action Models Need Test-time Future Imagination?, 2026. URL https://arxiv.org/abs/2603.16666.

Yang Yue, Yulin Wang, Bingyi Kang, Yizeng Han, Shenzhi Wang, Shiji Song, Jiashi Feng, and Gao Huang. DeeR-VLA: Dynamic Inference of Multimodal Large Language Models for Efficient Robot Execution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2411.02359.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic Control via Embodied Chain-of-Thought Reasoning. In *Conference on Robot Learning (CoRL)*, 2024. URL https://arxiv.org/abs/2407.08693.

Jianke Zhang, Yanjiang Guo, Xiaoyu Chen, Yen-Jen Wang, Yucheng Hu, Chengming Shi, and Jianyu Chen. HiRT: Enhancing Robotic Control with Hierarchical Robot Transformers. In *Conference on Robot Learning (CoRL)*, 2024. URL https://arxiv.org/abs/2410.05273.

Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, He Wang, Zhizheng Zhang, Li Yi, Wenjun Zeng, and Xin Jin. DreamVLA: A Vision-Language-Action Model Dreamed with Comprehensive World Knowledge. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2507.04447.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Robotics: Science and Systems (RSS)*, 2023. 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, Tai Wang, Ya-Qin Zhang, Jingjing Liu, and Xianyuan Zhan. X-VLA: Soft-Prompted Transformer as Scalable Cross-Embodiment Vision-Language-Action Model. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2510.10274.

Shoujing Zhu, Zhenyang Liu, Fungmiu Wang, Jiafeng Wang, Bo Yue, Guiliang Liu, Simo Wu, Xiangyang Xue, and Taiping Zeng. WeaveLA: Event Driven Cross-Subtask Latent Memory Weaving for Repetitive Robot Manipulation, 2026. URL https://arxiv.org/abs/2606.17463.

A Implementation and evaluation protocol

Initialization and architecture. RoboMME initializes the backbone and action expert from the $\pi_{0.5}$ base weights (Black et al., 2025a) and adds RoboMME's frame-sampling memory (Dai et al., 2026). RMBench and LIBERO initialize the backbone from the released RynnBrain-2B (Dang et al., 2026), and driving from Qwen3-VL-2B-Instruct (Bai et al., 2025), enlarged with trajectory tokens, each with a randomly initialized action expert. Table 7 counts the parameters of each policy and Table 8 lists the architectures. On every benchmark, the action expert is a second transformer stack with the backbone's attention geometry and no vocabulary: at layer ℓ, its positions attend to the backbone's

layer-ℓ keys and values together with their own, so the two stacks form one attention over one sequence while keeping separate weights (Section 3.1). The action positions continue the prefix's position IDs and attend to one another without a causal mask. Each action position is one action of the chunk. On RoboMME, the flow time enters through adaptive RMSNorm as in $\pi_{0.5}$, and the frame-sampling memory adds a memory attention over sampled frames and a memory-conditioned modulation in every layer of both stacks. On RMBench and LIBERO, the action expert embeds the noised action with the flow time and appends one clean state token. The driving action expert encodes each waypoint with Fourier features and outputs acceleration and curvature, integrated by a unicycle model.

Table 7: **Parameters of the policies (billions).** The tied output head is counted once. The components sum to Explicit Reasoning's total. On RoboMME, the total also includes the 0.315B frame-sampling memory, whose history frames pass through the same frozen SigLIP as the current image. A policy that generates no text (No Reasoning, and TacitVLA on the non-memory tasks) uses only the keys and values of the last language layer and not the rest of that layer or the final norm.

	Components				Policy		
Benchmark	Vision tower	Text embedding	Language layers	Action expert	No Reasoning	Explicit Reasoning	TacitVLA
RoboMME	0.415	0.527	1.982	0.430	3.547	3.669	3.669
RMBench	0.407	0.311	1.409	0.446	2.527	2.573	2.573
LIBERO	0.407	0.311	1.409	0.446	2.527	2.573	2.527
PhysicalAI-AV	0.407	0.313	1.409	1.774	3.857	3.903	3.857

Training data. RoboMME counting uses the 400 demonstrations of BinFill, PickXtimes, SwingXtimes, and StopCube, and the 16-task policy uses all 1,600, with subtask text on the four counting tasks only. On RMBench, each task has its own policy, trained on the benchmark's 50 released demonstrations. LIBERO trains on the 1,817 demonstrations that replay successfully in the simulator, of which 1,616 carry DeepThinkVLA reasoning annotations (Yin et al., 2026). Driving trains on 3.8 million reasoning-labeled events, each with a 1.6 s history and a 6.4 s future window.

Optimization. Table 9 lists batch sizes, learning rates, and schedules. All recipes use AdamW. RMBench and LIBERO decay the learning rate by cosine to 10% of its peak, driving to 1%, and RoboMME holds it constant after warmup. The cross-entropy weight λ is 0.1 on LIBERO and driving and 1.0 on RMBench, and on RoboMME each policy is trained at $\lambda \in \{0.1, 0.01, 0.001\}$ and reported at its best (Table 11). The same λ scales the tokenized-action term of Equation 1 in every policy, and in No Reasoning it scales that term alone. The gradient from the action expert into the backbone is enabled after 2k updates on RMBench and LIBERO, after the 4k warmup in driving, and from the start on RoboMME. *TacitVLA*'s updates cost slightly less than Explicit Reasoning's, whose action positions also attend to r_t, and No Reasoning's sequences carry no reasoning text, so its updates are the cheapest.

Context. On RoboMME, the context c_t is the front and wrist views, the benchmark's frame-sampling memory of up to 32 history frames, the robot state, the instruction, and, on the counting tasks, the ledger. On RMBench, it is, in order, the instruction, the episode's first head frame, the ledger of prior subtasks, the head frame from 32 actions earlier, the current head, left, and right frames, the robot state, and the cue for the next subtask. On LIBERO, it is the agent and wrist images, the agent image from eight steps earlier, the robot state, and the instruction. In driving, it is four frames from each of two front cameras at 10 Hz, 16 past ego states, and a fixed reasoning prompt. No Reasoning keeps the same observation on every benchmark, including RMBench's first and lagged frames, and drops only the ledger.

Prompts. Reasoning examples and direct examples use separate prompts, and the reasoning prompt adds a fixed cue that requests the reasoning. On LIBERO, the annotated demonstrations train under the reasoning prompt and the 201 unannotated ones under the direct prompt, for action supervision only. RoboMME and RMBench examples carry the ledger prompt, and driving examples the reasoning prompt. Both reasoning policies are deployed with the prompt they trained on, including *TacitVLA* on the non-memory tasks, where it generates no text.

Table 8: **Architecture of the policies.** In the output head row, d is the action dimension (14 on RMBench, 7 on LIBERO). The RoboMME policies use the $\pi_{0.5}$ architecture with RoboMME's frame-sampling memory. The RMBench, LIBERO, and driving backbones share the Qwen3-VL-2B architecture and differ in vocabulary.

Component	Setting	RoboMME	RMBench, LIBERO	PhysicalAI-AV
Vision encoder	layers	27 (SigLIP)	24	24
	width	1,152	1,024	1,024
Language model	layers	18 (Gemma-2B)	28	28
	width	2,048	2,048	2,048
	heads (query / key-value)	8 / 1	16 / 8	16 / 8
	feed-forward	16,384, GeGLU	6,144, SwiGLU	6,144, SwiGLU
Action expert	layers	18	28	28
	width	1,024	1,024	2,048
	heads (query / key-value)	8 / 1	16 / 8	16 / 8
	feed-forward	4,096, GeGLU	3,072	8,256
	coupling	shared attention with Gemma, layer by layer	attends to backbone keys and values, layer by layer	as RMBench and LIBERO
	time conditioning	adaptive RMSNorm (scale, shift, gate)	sinusoidal, fused into each action	Fourier features per waypoint
Action interface	actions per chunk	20	32 (RMBench), 10 (LIBERO)	64
	action dimension	32 (8 active)	14, 7	2: acceleration, curvature
	state	as $\pi_{0.5}$	one token, 14-D or 8-D	16 past ego states in the prefix
	output head	linear $1{,}024 \to 32$	MLP $1{,}024 \to 1{,}024 \to d$	linear $2{,}048 \to 2$
Flow matching	sampler, solver, steps	beta, Euler, 10	beta, Euler, 32	beta, Euler, 10

Table 9: **Training settings.** Learning rates are for the vision tower, the language model, and the action expert, respectively, and RoboMME uses one rate for all three.

Setting	Batch	Peak learning rate	Warmup	Updates	Noise draws
RoboMME counting	64	5×10^{-5}	5k	60k	1
RoboMME all tasks	64	5×10^{-5}	10k	80k	1
RMBench	512	5×10^{-6} / 5×10^{-5} / 2×10^{-4}	2k	80k	16
LIBERO	512	5×10^{-6} / 5×10^{-5} / 2×10^{-4}	2k	40k	16
PhysicalAI-AV	256	1×10^{-5} / 1×10^{-4} / 4×10^{-4}	4k	100k	1

Reasoning targets. On LIBERO, for the instruction "turn on the stove and put the moka pot on it", DeepThinkVLA's released per-frame annotation at the first frame reads:

> This is the initial frame. The stove (right-front) is off. The moka pot (center-front) and a pan (left-front) are on the counter. The stove knob (right-front, on stove) is accessible. To begin the global task, the stove must be turned on first. Turn on the stove.

The prompt is "Task: *instruction*", followed on reasoning examples by "Current reasoning:". On RMBench, each call's prompt carries the ledger so far under "Previous subtask:" and asks for "Current subtask:". In driving, one training annotation reads, verbatim: "Maintain lane and search

for a usable gap to the left to prepare for a lane change because a car is overtaking alongside in the left lane while a slower SUV is ahead in the current lane." The prompt asks the model to "output the chain-of-thought reasoning of the driving process, then output the future trajectory".

Table 10: **Observation, action, and execution.** Solver steps are uniform Euler steps. Closed-loop driving replans at every 100 ms control step of the simulator, so one waypoint is executed per call.

Benchmark	Observation	Predicted chunk	Executed	Solver steps
RoboMME	front and wrist views, up to 32 sampled history frames	20 actions (32-D, 8 active)	16	10
RMBench	first frame, head frame 32 actions earlier, current head/left/right	32 targets (14-D)	32	32
LIBERO	agent and wrist images, agent image 8 steps earlier	10 actions (7-D)	5	32
PhysicalAI-AV	4 frames from two front cameras, 16 past ego states	64 waypoints / 6.4 s	1	10

Evaluation protocol. Table 10 lists observations, action chunks, and solver steps. RoboMME is evaluated as the benchmark specifies: its official test episodes (50 per task), horizon, chunking, and the three seeds its evaluation script names. RMBench evaluates the benchmark's own 100 cases per task, disjoint from the 50 demonstrations and identical for every policy and evaluation seed. LIBERO evaluates 50 initial states per task (2,000 cases) under the DeepThinkVLA step caps, and LIBERO-Plus its 10,030 published variants under the same caps, with the perturbations applied as the benchmark's released wrapper applies them. Reasoning is decoded greedily on the three robot benchmarks, up to 128 tokens (RoboMME) or 256 (RMBench, LIBERO), and sampled in driving at temperature 0.6 and top-p 0.98. Open-loop driving reports the minimum over six sampled trajectories of the average displacement error on the 1,434 held-out segments, averaged over action-sampling seeds. Closed loop follows the protocol of the AlpaSim E2E Closed Loop Challenge on its 441-scene public split (NVIDIA, 2026): 20 s rollouts at 10 Hz under the challenge's controller and scoring, three rollouts per scene, with a failed rollout counted as a failure. The driver samples one trajectory per call and executes it, the protocol of the challenge's reference driver. The route is not a model input.

Ledger semantics. On RoboMME, one subtask sentence, the benchmark's simple subgoal without the image coordinates of its grounded variant, is generated per call. An entry enters the ledger only after its 16-action chunk has executed, so the first entry appears at the second call. Adjacent duplicates are removed, and the ledger resets per episode and is never truncated. Explicit Reasoning generates its entry before the action it conditions on and *TacitVLA* after computing the action. Both append it at the same boundary, so every later call conditions on a ledger built under one protocol. In both policies, an entry names the subtask being undertaken and carries no outcome: a grasp that fails still leaves its entry, and the count recovers when the next call generates the subtask again. On RMBench, the next subtask is decoded greedily for each 32-action plan.

Deployment. On the memory tasks, the deferred generation continues from the action path's key-value cache after the action is computed, and if it has not finished when the next observation arrives, the next call waits for it (Equation 4). In training, the ledger prefix is the annotated subtask sequence, and at deployment it is generated, a shift that Explicit Reasoning faces as well. On the non-memory tasks, *TacitVLA* can still generate the reasoning text after the action when it is wanted for inspection.

Baselines. Published rows follow each benchmark's protocol and are quoted as printed, except means over our task sets that a source does not print, which we compute from its per-task values. RoboMME rows are the RoboMME paper's (Dai et al., 2026), except WeaveLA and AGM, which report their own. RMBench rows include QwenOFT and the MemER and MemoryVLA rows from EventVLA's reproductions (Yang et al., 2026a). LIBERO-Plus rows are zero-shot policies trained on the LIBERO demonstrations alone, plus the robot-action-pretrained MolmoAct for reference; the $\pi_{0.5}$ and MolmoAct2 rows are LIT's baselines (Lin et al., 2026b), and the LIBERO-Plus values of MolmoAct, DreamVLA, and VLA-Adapter come from Dalal et al. (2026), Lin et al. (2026d), and He et al. (2026), respectively. Systems with robot-action pretraining fall outside this matched

comparison, such as DeepThinkVLA (79.0; Yin et al., 2026) on LIBERO-Plus, as do systems with significantly more parameters than ours, such as VLA-GSE (81.2; Jiang et al., 2026) and FoMoVLA (80.5; Li et al., 2026b) on LIBERO-Plus and MemoryWAM (81.5; Yang et al., 2026b) on RMBench.

B Ablations and full result tables

Tables 11 to 14 hold the RoboMME weight sweep, the per-axis and per-suite LIBERO tables behind Section 6, and the ledger correctness counts on the memory tasks.

Table 11: **RoboMME, the cross-entropy weight sweep.** Success (%) of the three policies at $\lambda \in \{0.1, 0.01, 0.001\}$, mean $\pm$ standard deviation over three evaluation seeds: the Counting-suite policies on the four counting tasks (200 cases) and the 16-task policies on all 16 tasks (800 cases).

	Trained on the Counting suite			Trained on all suites		
Policy	λ=0.1	0.01	0.001	λ=0.1	0.01	0.001
No Reasoning	77.3 $\pm$ 2.5	79.0 $\pm$ 1.0	74.5 $\pm$ 2.8	43.6 $\pm$ 0.2	46.5 $\pm$ 0.9	44.0 $\pm$ 1.6
Explicit Reasoning	**86.7** $\pm$ 2.0	84.7 $\pm$ 1.6	73.0 $\pm$ 2.0	43.5 $\pm$ 1.2	45.8 $\pm$ 1.5	43.8 $\pm$ 1.3
TacitVLA	85.5 $\pm$ 3.3	**89.5** $\pm$ 1.3	**82.5** $\pm$ 2.5	**44.0** $\pm$ 0.9	**47.6** $\pm$ 1.4	**48.1** $\pm$ 1.0

Table 12: **LIBERO-Plus by perturbation axis.** Success (%) on each of the seven perturbation axes and on the full 10,030-case suite, mean $\pm$ standard deviation over three evaluation seeds.

Policy Cases	Robot init. 1,550	Sensor 1,601	Camera 1,599	Language 1,537	Objects 1,525	Background 1,076	Light 1,142	All 10,030
No Reasoning	59.7 $\pm$ 1.1	51.7 $\pm$ 1.1	41.0 $\pm$ 1.1	76.6 $\pm$ 0.9	73.2 $\pm$ 1.0	92.9 $\pm$ 0.7	94.0 $\pm$ 0.6	67.6 $\pm$ 0.4
Explicit Reasoning	**71.1** $\pm$ 0.8	51.1 $\pm$ 0.9	**51.5** $\pm$ 0.9	**83.7** $\pm$ 0.6	**78.3** $\pm$ 0.7	97.2 $\pm$ 0.3	**96.8** $\pm$ 0.4	**73.5** $\pm$ 0.3
TacitVLA	69.9 $\pm$ 0.8	**60.0** $\pm$ 0.8	42.2 $\pm$ 0.8	83.0 $\pm$ 0.6	77.1 $\pm$ 0.7	**98.0** $\pm$ 0.3	95.4 $\pm$ 0.4	72.9 $\pm$ 0.3

Table 13: **LIBERO, every policy and suite.** Success (%) on the four standard suites (50 trials per task) and on the 10,030-case LIBERO-Plus suite, mean $\pm$ standard deviation over three evaluation seeds.

Policy	Spatial	Object	Goal	Long	Avg.	Plus
No Reasoning	**99.2** $\pm$ 0.4	99.6 $\pm$ 0.3	**98.8** $\pm$ 0.5	96.4 $\pm$ 0.9	98.5 $\pm$ 0.3	67.6 $\pm$ 0.4
Explicit Reasoning	98.4 $\pm$ 0.7	99.8 $\pm$ 0.3	97.8 $\pm$ 0.9	96.4 $\pm$ 1.1	98.1 $\pm$ 0.4	**73.5** $\pm$ 0.3
TacitVLA	**99.2** $\pm$ 0.5	**100.0** $\pm$ 0.0	98.0 $\pm$ 0.7	**97.2** $\pm$ 0.8	**98.6** $\pm$ 0.3	72.9 $\pm$ 0.3

Ledger correctness. On the memory tasks, later calls rely on the ledger to count what has been done, so deferring generation is free only if the entries stay correct. Table 14 counts the episodes whose final ledger matches the instructed count: the number of pick-and-place, swing, or bin-fill completions on the RoboMME counting tasks (StopCube, which counts the cube's passes rather than the robot's completions, is not in the table), and the presses per button plus one confirm entry on Press Button. *TacitVLA* and Explicit Reasoning are within two points of each other on every task, and every *TacitVLA* failure on Press Button ends on the step budget with the physical press count off target while its ledger is correct.

Table 14: **Correct ledgers (%).** Episodes whose final generated ledger matches the instructed count, for the Counting-suite policies of Table 2 and the Press Button policies of Table 4.

Policy	BinFill	PickXtimes	SwingXtimes	Press Button
Explicit Reasoning	90.0	100.0	100.0	99.7
TacitVLA	92.0	100.0	100.0	99.7

C Latency accounting

What is timed. Each stage of a call is timed by host wall-clock timers with device synchronization at each stage boundary, on one NVIDIA H100 80 GB GPU in bf16, one policy process at a time.

Table 15: **Stage times and the latency per call (ms).** The left columns give the execution window T_{exec} and each policy's action path T_{action} and reasoning cost T_{reason} (Section 3.3), and the right columns give the latency per call: T_{action} for No Reasoning and TacitVLA, and $T_{\text{action}} + T_{\text{reason}}$ for Explicit Reasoning, whose action waits for the generation. TacitVLA generates during T_{exec} on the memory tasks and not at all on the non-memory tasks. All values are means of per-call stage timers on one H100. RMBench's window is the motion planner's mean time to execute a plan.

		T_{action}			T_{reason}		Latency		
Benchmark	T_{exec}	No Reasoning	TacitVLA	Explicit Reasoning	TacitVLA	Explicit Reasoning	No Reasoning	TacitVLA	Explicit Reasoning
RoboMME, counting	800	127	127	127	93	93	127	127	220
RMBench, Press Button	6,972	170	171	170	66	66	170	171	236
RMBench, Blocks Ranking Try	8,116	162	171	171	81	87	162	171	257
RMBench, Cover Blocks	9,359	170	170	169	101	106	170	170	275
RMBench, Battery Try	11,181	162	169	168	120	126	162	169	294
LIBERO	250	147	147	149	–	345	147	147	495
LIBERO-Plus	250	147	147	150	–	347	147	147	497
PhysicalAI-AV, closed loop	100	127	127	126	–	143	127	127	269

The visual encoder, the prompt prefill, and each denoising step run as captured CUDA graphs, and text generation replays one graph per token over a static key-value cache. T_{action} is visual encoding, prompt prefill, and denoising, the same three stages for every policy. The prompt prefill sits in T_{action} for each policy (under Explicit Reasoning it runs as the first step of generation). T_{reason} is the autoregressive generation of r_t plus, for Explicit Reasoning, the backbone's pass over the generated tokens that its action expert conditions on (5 to 8 ms), which runs after generation and before denoising. On the memory tasks, *TacitVLA*'s action and its deferred generation share one prefill, so that prefill is on its action path and its T_{action} is a few milliseconds above No Reasoning's.

Execution windows. We use the simulators' own control rates. LIBERO executes 5 actions per call at 20 Hz (250 ms), RoboMME 16 at 20 Hz (800 ms), and closed-loop driving replans at every 100 ms control step. RMBench has no control rate: an action is a joint-position target, and the simulator hands each of the 32 targets of a plan to a motion planner that moves the arm under its joint velocity and acceleration limits, stepping the physics at 4 ms until the arm arrives. A plan's duration is set by the commanded motion. We count those physics steps per plan and report the mean over the 100 cases of each task (218 to 349 ms per target). No deferred generation exceeded its window in any call (the longest RoboMME entry took 180 ms, the longest RMBench entry 225 ms), so *TacitVLA*'s latency is T_{action} in every cell.

Reading the table. On the non-memory tasks, *TacitVLA* and No Reasoning run the same stages, and their latencies agree to the millisecond. On the memory tasks, both reasoning policies generate entries of similar cost (RMBench T_{reason} 66 to 120 compared with 66 to 126 ms, the latter including the pass over the entry), and they differ only in where that generation sits (Section 6.2). The RoboMME row is the 16-task policy on its four counting tasks. On its other 12 tasks, which keep no ledger, no policy generates text and the three latencies coincide (143 to 144 ms).

D QUALITATIVE RESULTS

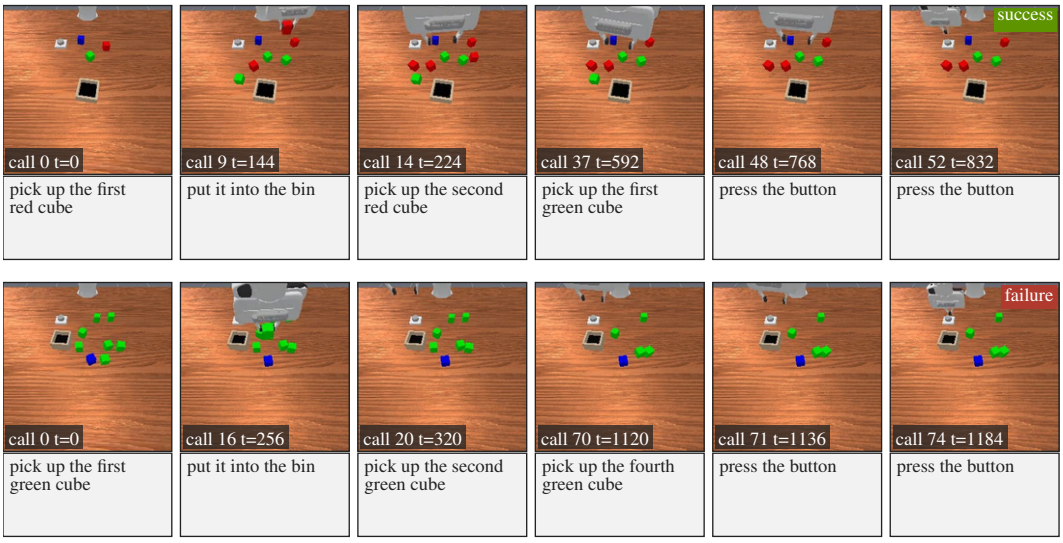
Figures 4 to 8 show the reasoning text the policies generate and the episodes it accompanies, for RoboMME, RMBench, LIBERO, LIBERO-Plus, and driving: a successful episode and a failed one, each rolled out by *TacitVLA* and by Explicit Reasoning from the same initial state, with the reasoning text generated at each shown call under its frame. The two upper rows are *TacitVLA* on the successful and then the failed episode, and the two lower rows are Explicit Reasoning on the same two episodes. Frames are stamped with the call index and the simulator step before that call. The box under a frame holds the text the backbone generated at that call, verbatim. On the memory tasks, the text is the ledger entry, and the frames are the calls that generate a new entry, plus the episode's last call; on LIBERO and driving, the text is the reasoning for that call and the frames span the episode. On the memory tasks, the text is generated during execution of the action chunk and enters the ledger for the

next call. On LIBERO and driving, where *TacitVLA* generates no text at deployment, we generated it for this appendix after computing each action, as on the memory tasks, so the actions are unchanged.

In these examples, no failure mode is clearly specific to *TacitVLA*. On every failure shown, Explicit Reasoning fails the same episode with the same kind of execution error: cubes that do not end in the bin, a confirm press that never registers, a grasp carried without a placement, a lateral departure from the lane corridor.

Task instruction (successful episode): put three red cubes and one green cube into the bin, then press the button to stop
Task instruction (failed episode): put four green cubes into the bin, then press the button to stop

TacitVLA



Explicit Reasoning

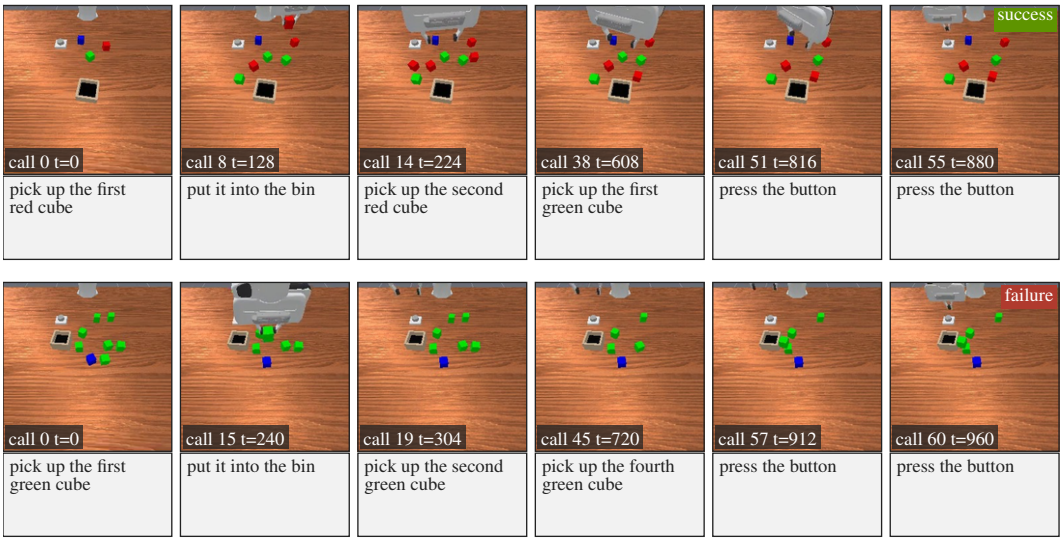
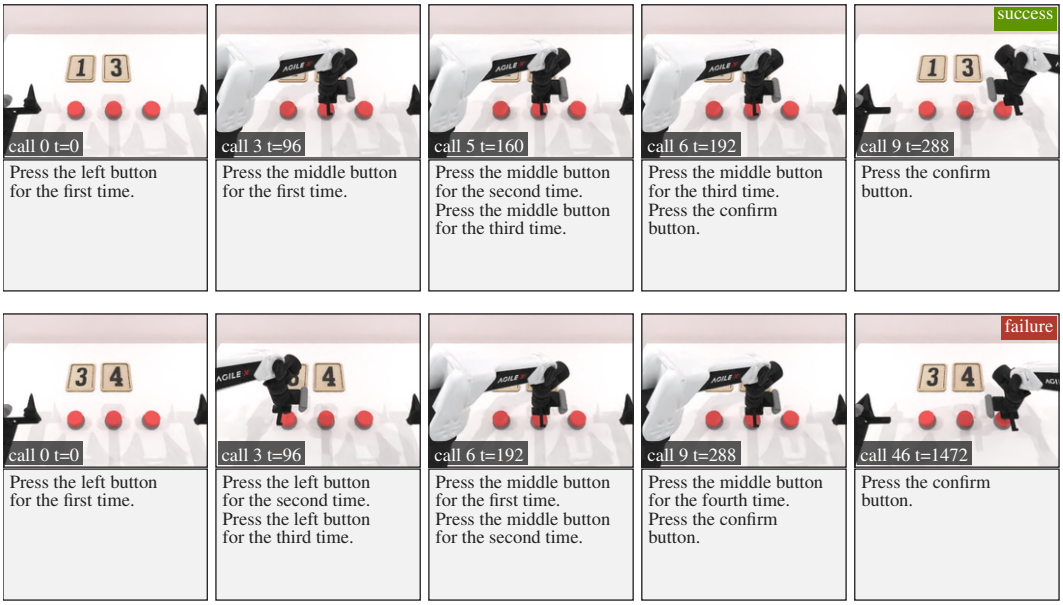


Figure 4: **RoboMME BinFill.** The front camera at the calls that generate a new ledger entry and at the episode's last call. Counting-suite policies of Table 2.

Task instruction: Observe the two numbers on the table. Press the left button the number of times corresponding to the number on the left, and press the middle button the number of times corresponding to the number on the right. Then press the right button once to confirm.

TacitVLA



Explicit Reasoning

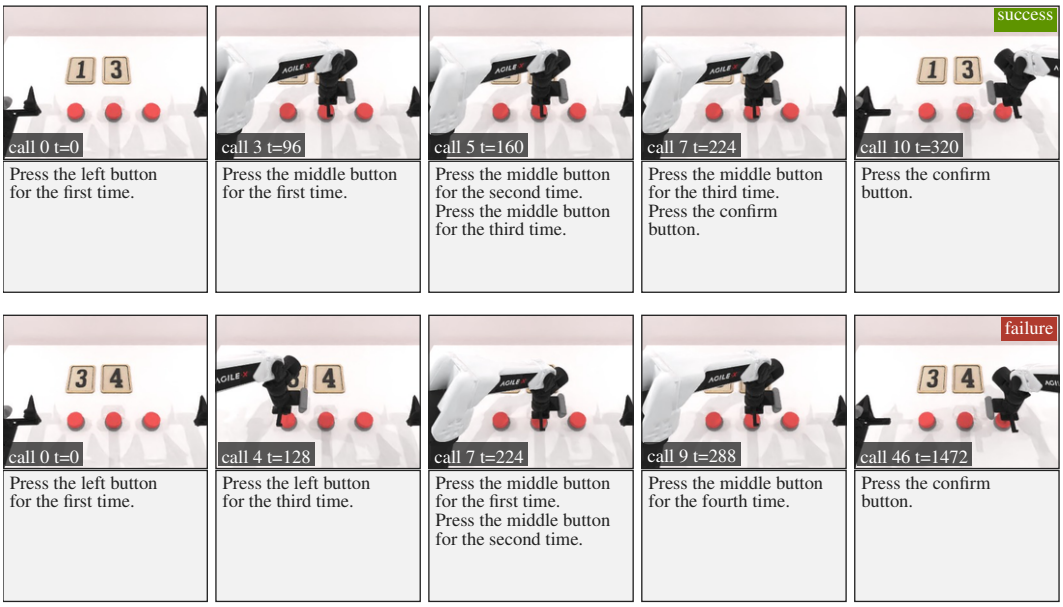
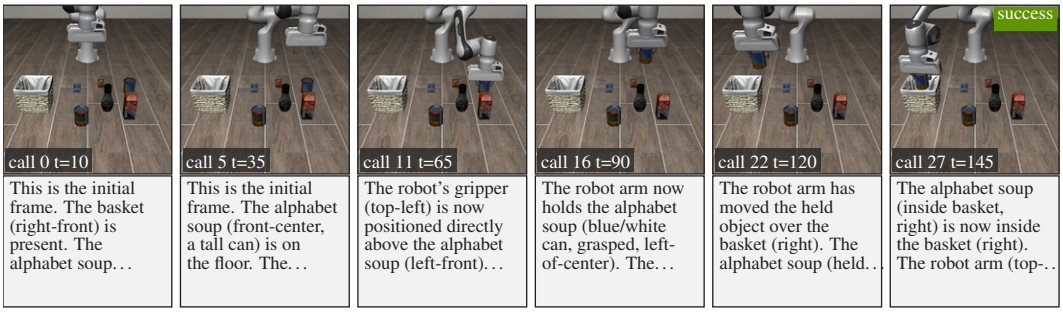


Figure 5: **RMBench Press Button.** The head camera at the calls that generate a new ledger entry and at the episode's last call. A call can add two entries when a press completes within the action chunk. Policies of Table 4.

Task instruction (successful episode): pick up the alphabet soup and place it in the basket
Task instruction (failed episode): put both the alphabet soup and the tomato sauce in the basket

TacitVLA



Explicit Reasoning

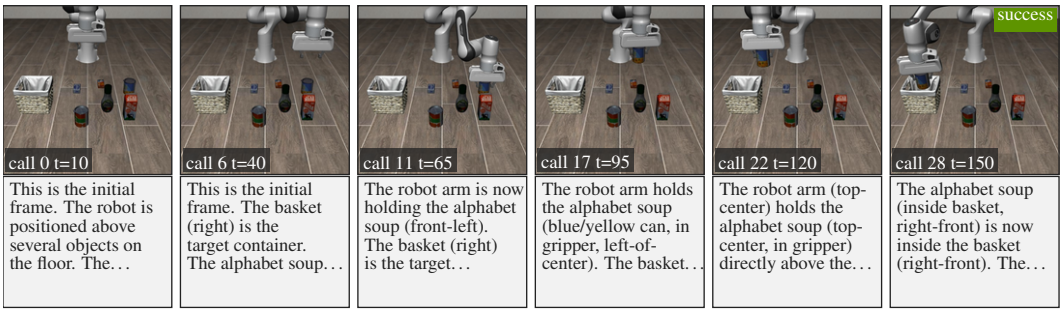
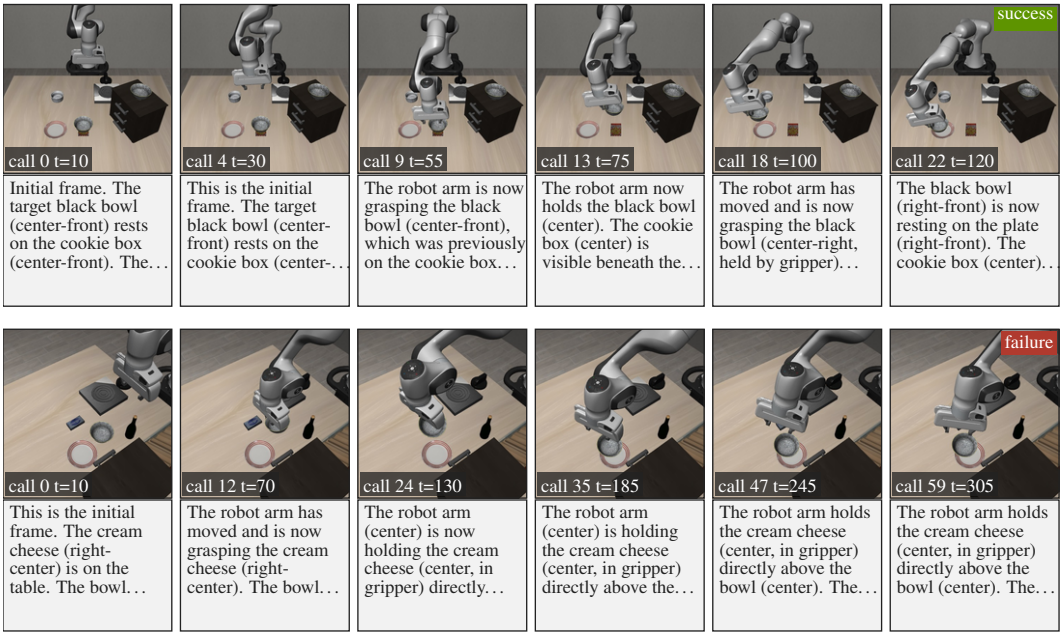


Figure 6: **LIBERO.** The agent camera at six calls, with the opening of the reasoning generated at each. Policies of Table 5.

Task instruction (successful episode): pick up the black bowl on the cookie box and place it on the plate
Task instruction (failed episode): put the cream cheese in the bowl

TacitVLA



Explicit Reasoning

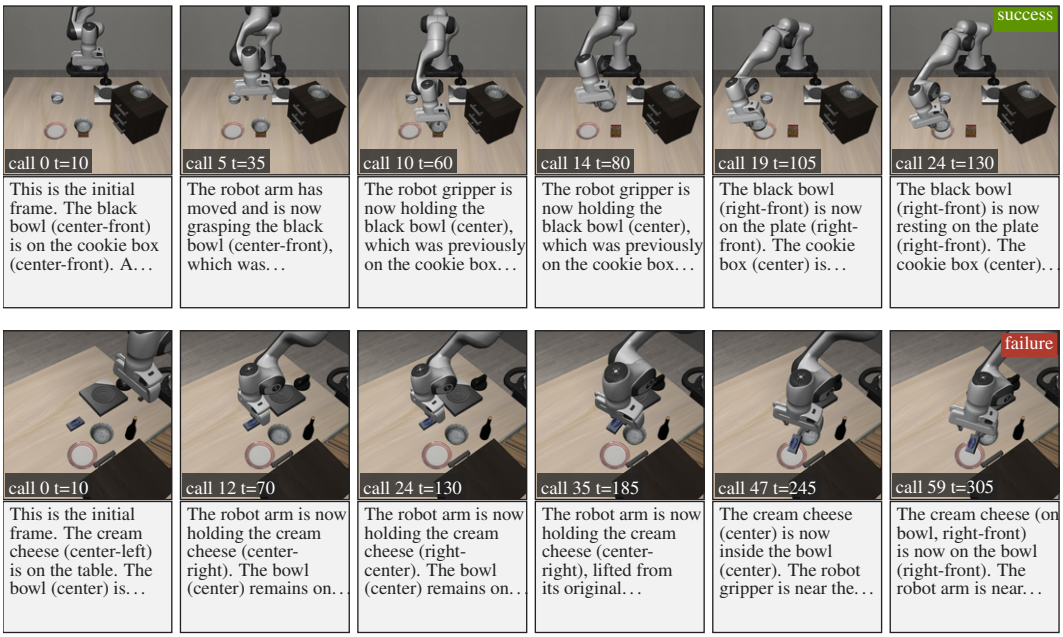
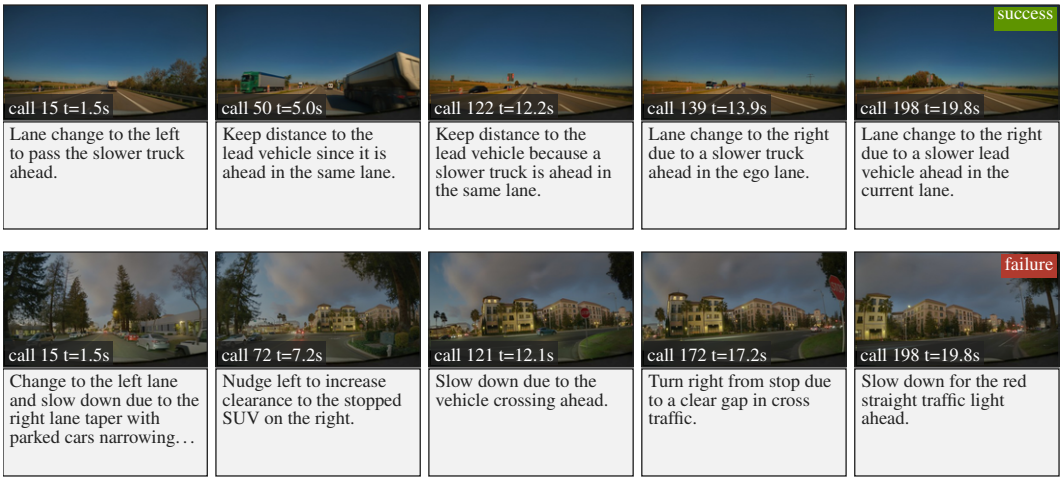


Figure 7: **LIBERO-Plus, Camera Viewpoints.** The agent camera on two tasks under perturbed viewpoints. Policies of Table 5.

Task instruction: output the chain-of-thought reasoning of the driving process, then output the future trajectory

TacitVLA



Explicit Reasoning

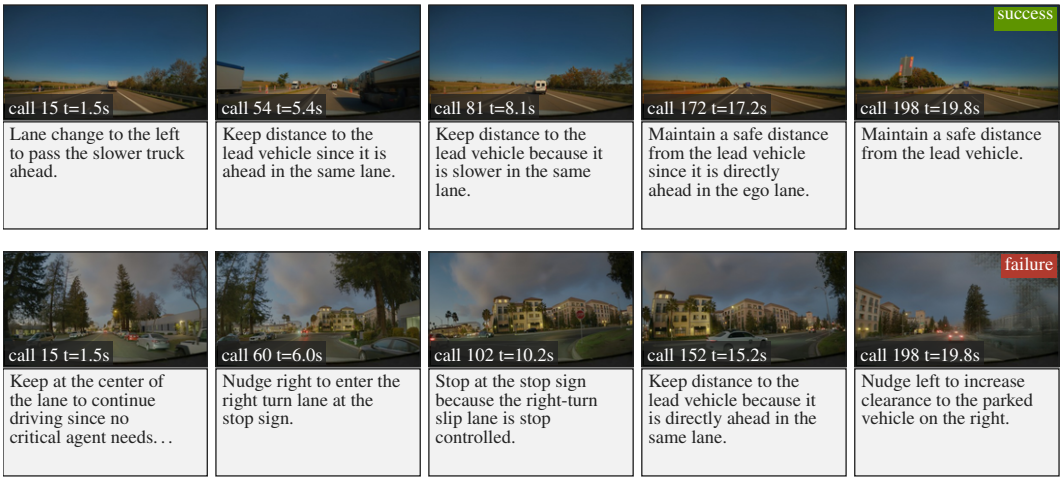


Figure 8: **PhysicalAI-AV closed loop.** The front camera at five calls of two public-split scenes, with the chain of causation generated at each call. Policies of Table 6.