PONDERPOUNCE: A PRETRAINED MLLM AS AN EPISODE CONTEXT ENGINE FOR ROBOT CONTROL

Suhwan Choi[1,2]* **Jaeyoon Jung**[1]* **Sungkyung Kim**[1,2] **Yunsung Lee**[1]† **Youngjae Yu**[2]†

[1]MAUM.AI [2]Seoul National University

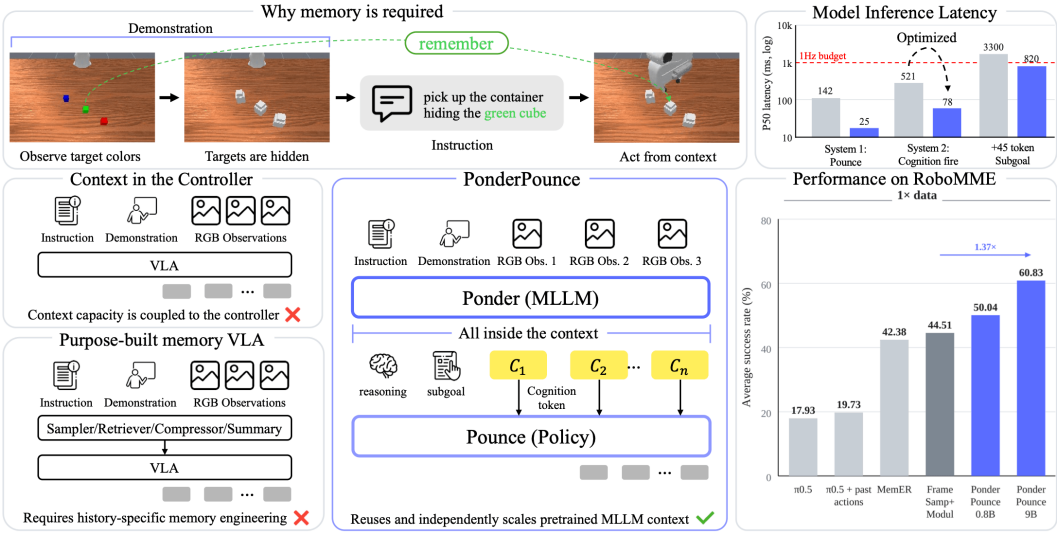

Figure 1: **Pretrained MLLM context as scalable robot memory.** Unlike designs processing context within the controller or through purpose-built memory, PONDERPOUNCE retains history in PONDER's context and asynchronously routes cognition to POUNCE. This separation keeps context processing off the action path and lets System 2 scale without changing the controller architecture.

ABSTRACT

Multimodal large language models (MLLMs) can integrate long visual histories, reason under partial observability, and infer behavior from a few examples. Yet vision-language-action (VLA) models generally inherit pretrained representations without using this contextual capacity as episode memory. Memory-dependent policies address this gap through purpose-built history mechanisms. PONDERPOUNCE instead reuses an MLLM's native causal context as robot memory. PONDER, a System 2 MLLM, accumulates episode observations, demonstrations, and prior cognition in its native causal context and can generate subgoal text and demonstration reasoning for internal use. POUNCE, a System 1 VLA, receives the current observation, instruction, and proprioception directly; through the PONDER–POUNCE interface, it asynchronously receives only the newest continuous cognition token and its age. Both are jointly trained end to end without a purpose-built memory module or separate bridge pretraining. Optimized serving achieves p50 latencies of 78 ms for cognition refresh and 25 ms for action-model invocation, supporting 20 Hz action playback. On RoboMME with base-scale training data, PONDERPOUNCE reaches 60.83% with 9B and 50.04% with 0.8B under the same POUNCE architecture and interface, versus 44.51% for FrameSamp+Modul and 17.93% for the current-observation $\pi_{0.5}$. With 9× data, it reaches 75.54% versus 57.88% for FrameSamp+Modul. On RoboCasa-DC, the same interface learns from action supervision alone and reaches 12.5% versus 11.6% for the strongest published demonstration-conditioned baseline, falling to 8.6% when cognition is replaced by a learned null state.

*Equal contribution.
†Co-corresponding authors.

1 INTRODUCTION

Robotic manipulation often depends on evidence absent from the current observation, such as an occluded object, an earlier event, the identity of a referent, or a demonstrated procedure (Dai et al., 2026; Son et al., 2026). Pretrained multimodal large language models (MLLMs) can integrate long visual histories and reason from a few examples (Team, 2026). VLA models inherit visual and language representations, but generally do not carry this contextual capacity into control as episode memory. Prior work instead routes context to control through architecture-specific mechanisms such as sampling, storage, retrieval, or compression (Dai et al., 2026; Fang et al., 2025; Torne et al., 2026; Sridhar et al., 2026; Shi et al., 2026), cross-attention or learned demonstration representations (Jain et al., 2024; Kim et al., 2025), and intermediate maps or plans (Huang et al., 2023; Son et al., 2026). We consider a different design point: using the native causal context of a pretrained MLLM as the episode context engine while a pretrained VLA remains responsible for control.

We study this interface through two forms of context-dependent control. Long-horizon memory requires acting on information observed earlier but absent from the current frame, whereas test-time demonstration conditioning requires inferring behavior from a provided example. On RoboMME (Dai et al., 2026), a current-observation $\pi_{0.5}$ reaches only 17.93%, and including past actions raises success to 19.73%, compared with 90.50% for humans. This gap motivates our central question: what representation and interface can make accumulated episode context actionable for the controller?

PONDERPOUNCE answers this question by routing accumulated episode context through a continuous cognition token in an asynchronous dual system. Following dual-process terminology (Kahneman, 2011), we refer to PONDER as System 2 and POUNCE as System 1. PONDER, a 9B MLLM, maintains append-only episode and demonstration context; POUNCE, an action model, conditions on the newest cognition and its age. Rather than adding a purpose-built memory mechanism, we adapt the contextual capacity of a pretrained MLLM and expose its readout to the controller. Decoupling the two systems in representation and frequency makes System 2 capacity an independent design axis, allowing a larger context model to remain off the fast action path without changing the POUNCE architecture. We jointly train both pretrained components without subsystem-specific pretraining or separate bridge pretraining. Subgoal text at annotated transitions and demonstration-reasoning targets ground PONDER's LM head during training but remain internal to System 2. Caching and fused inference support 20 Hz playback. Figure 1 summarizes the design.

On RoboMME, PONDERPOUNCE reaches 60.83% at the base data scale and 75.54% with 9× data, versus 44.51% and 57.88% for FrameSamp+Modul. Under the same controller architecture and interface, the 9B context engine exceeds its 0.8B counterpart by 10.79 percentage points, showing that control benefits from a larger pretrained context engine. The same interface reaches 12.5% on RoboCasa-DC versus 11.6% for the strongest published demonstration-conditioned baseline, while replacing cognition with a learned null state reduces success to 8.6%. These results support pretrained MLLM context as a strong general-purpose memory substrate, while task-level results show that it is not uniformly best across all memory demands.

Key contributions.

- **An independently scalable pretrained MLLM context engine.** We reuse an MLLM's native causal context to retain episode observations and demonstrations without adding a purpose-built memory store, retrieval policy, or history compressor. System 2 capacity can change without modifying the fast controller architecture.
- **An asynchronous continuous context-to-control interface.** A cognition token and its age connect two pretrained models that run on decoupled clocks and are jointly trained end to end. Transition-triggered subgoal generation and demonstration-reasoning supervision ground cognition through the LM head while text remains internal to System 2.
- **Empirical validation across memory and demonstration conditioning.** On RoboMME, PONDERPOUNCE achieves higher average success than the strongest non-oracle baseline at both data scales, and a 9B context engine improves over 0.8B under the same interface. Inference-time interventions show that control depends on the transmitted cognition, and the interface extends to cross-embodiment demonstration conditioning on RoboCasa-DC.

2 RELATED WORK

Purpose-built episode memory. Memory-augmented policies differ in both how they retain history and how it reaches action. FrameSamp+Modul (Dai et al., 2026) injects sampled-history tokens through layer-wise modulators; SAM2Act+ (Fang et al., 2025) attends to a segmentation-derived bank; and MemER (Sridhar et al., 2026) retrieves selected keyframes and emits textual subgoals. MEM (Torne et al., 2026) combines short-horizon video with recursive language memory. MemoryVLA (Shi et al., 2026) calls its per-observation VLM summary a cognitive token and combines it with perceptual tokens to retrieve and fuse entries from an external memory bank. RoboTTT (Jiang et al., 2026) instead compresses history into fast weights updated at deployment. Each adds a history-specific sampler, store, retrieval path, summary, or state update. PONDERPOUNCE asks whether a pretrained MLLM's native causal context can serve as episode memory without such a purpose-built component.

Demonstrations as test-time context. One-shot imitation established that policies can infer tasks from a single example through attention or meta-learned adaptation (Duan et al., 2017; Yu et al., 2018). Recent systems use richer test-time context: ICRT (Fu et al., 2024) places sensorimotor demonstrations and rollouts in one autoregressive sequence; Vid2Robot (Jain et al., 2024) cross-attends to prompt-video features; UniSkill (Kim et al., 2025) extracts embodiment-agnostic skills; and ViVLA (Chen et al., 2025a) jointly models demonstration and robot actions. RoboCat (Bousmalis et al., 2023) studies adaptation from action-labeled experience, while SeeTraceAct (Son et al., 2026) grounds a demonstration-derived latent plan with future visual traces. These methods establish demonstrations as useful context, but process them through the controller, a task-specific representation, or parameter adaptation.

Routing context to control. Retaining context and exposing it to control are distinct choices. Text hierarchies emit subgoals, language motions, or coordinates (Shi et al., 2025; Belkhale et al., 2024; Chen et al., 2026b), while LCB (Shentu et al., 2024), FiS-VLA (Chen et al., 2025b), and Helix (Figure AI, 2025) transmit internal representations. Such continuous channels can still collapse toward instruction-level information (Cui et al., 2025). PONDERPOUNCE retains context in MLLM activations rather than an external bank (Shi et al., 2026), a controller sequence (Fu et al., 2024), or updated weights (Jiang et al., 2026), and asynchronously routes one cognition token and its age to the action model. In the main model, subgoal text and demonstration reasoning remain internal to System 2. Helix is the closest asynchronous analogue, but does not recurrently transmit episode or demonstration context. To our knowledge, prior work has not combined persistent MLLM context, decoupled clocks, and recurrent continuous routing. Appendix C provides the broader comparison.

3 PONDERPOUNCE

PONDERPOUNCE connects a pretrained MLLM (PONDER) to a pretrained action model (POUNCE) through recurrent continuous cognition tokens. At each sparse query, PONDER updates K carrier states from the task, episode history, optional demonstrations, and prior cognition. POUNCE runs independently and conditions each action chunk on the newest ready state. We train the assembled system end to end; where annotations exist, transition, subgoal-text, and demonstration-reasoning targets supervise PONDER, while subgoal text and demonstration reasoning remain inside System 2. Figure 2 summarizes the architecture and notation.

3.1 ARCHITECTURE

PONDER (System 2). At query t, PONDER appends observation O_t to an append-only causal context containing the instruction Ins, optional demonstrations $D_{1:n}$, internally generated subgoal text and demonstration reasoning, and earlier cognition carriers. Because history remains in this context, each query can attend to accumulated observations and demonstrations without an explicit action history. PONDER then predicts $T_t \in \{T_{\mathrm{Yes}}, T_{\mathrm{No}}\}$. If $T_t{=}T_{\mathrm{Yes}}$, it generates S_t before inserting the carrier positions. At the first execution transition of a demonstration episode, DR precedes S_t. If $T_t{=}T_{\mathrm{No}}$, the carriers immediately follow O_t. Their final hidden states form $\mathbf{C}_t \in \mathbb{R}^{K\times H}$ without decoding or pooling. Thus every query produces fresh cognition, while DR and S_t remain internal to System 2. Appendix A.2 specifies the token serialization and targets.

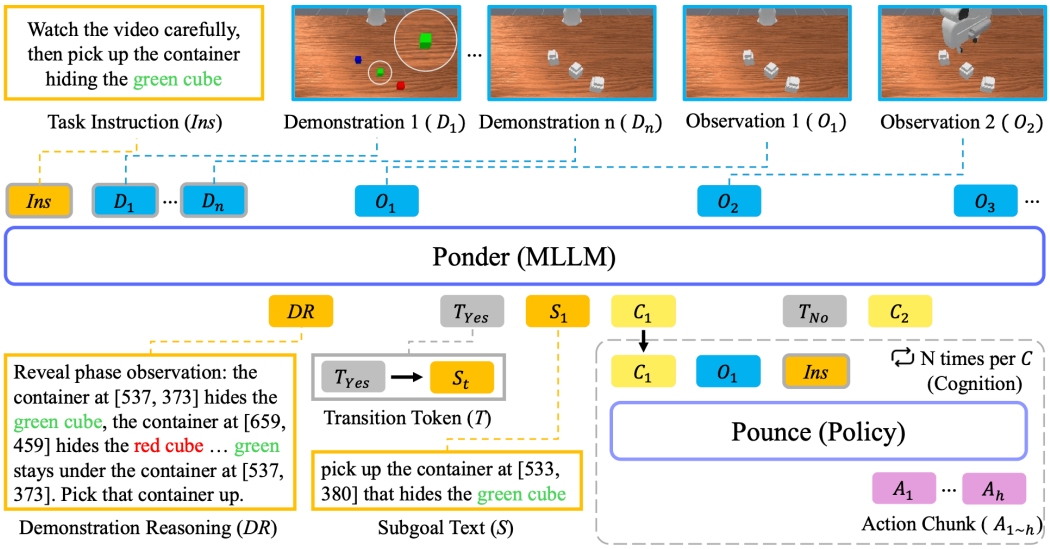

Figure 2: **PONDERPOUNCE architecture.** PONDER accumulates the instruction Ins, demonstrations $D_{1:n}$, and observations O_t. The transition token T_t gates optional internal text DR and S_t, while carrier states form cognition $\mathbf{C}_t$. POUNCE reuses the newest cognition with the current observation to predict $A_{1:h}$.

POUNCE (System 1). At invocation j, POUNCE receives Ins, the current observation O_j, proprioception, and a prefix formed from the newest ready cognition and its sinusoidally encoded age. If no cognition is ready, $\widetilde{\mathbf{C}}_j = \mathbf{C}_\varnothing$ for learned null cognition $\mathbf{C}_\varnothing \in \mathbb{R}^{K\times H}$ and $\Delta(j) = 0$; otherwise, $\widetilde{\mathbf{C}}_j = \mathbf{C}_{\mathrm{ref}(j)}$:

$$\mathbf{P}_j = [\,\mathbf{e}_{\Delta(j)},\ \mathbf{W}_c\widetilde{\mathbf{C}}_{j,1}, \ldots, \mathbf{W}_c\widetilde{\mathbf{C}}_{j,K}\,].$$

The action head predicts $A_{1:h}$, which is played back at 20 Hz. The System 2-to-System 1 interface transmits only cognition and age. In the subgoal-text reference, the latest subgoal text replaces the cognition prefix and updates POUNCE only at predicted transitions. Demonstration reasoning remains internal to PONDER.

3.2 TRAINING AND GROUNDING

We initialize both systems from pretrained checkpoints and jointly optimize all trainable components end to end without separate bridge pretraining; components frozen by the underlying action-model recipe remain frozen. We randomly initialize the carrier input embeddings, cognition projector, and age projection, and initialize learned null cognition to zero.

$$\mathcal{L} = w_1\mathcal{L}_{\mathrm{fm}} + w_2\mathcal{L}_{\mathrm{ground}}, \tag{1}$$

where $\mathcal{L}_{\mathrm{fm}}$ is action flow-matching MSE and $\mathcal{L}_{\mathrm{ground}}$ is token cross-entropy over T_t^*, S_t at annotated transitions, and, on demonstration episodes, first-execution DR. During training, the annotated transition target T_t^* and its text payload are teacher-forced before the carrier positions. At inference, PONDER predicts T_t and generates text only if $T_t{=}T_{\mathrm{Yes}}$. The flow-matching loss reaches PONDER only through $\mathbf{C}_t$, while grounding enters through the LM head. Neither DR nor S_t enters POUNCE in the default interface; $w_2{=}0$ without annotations. Motivated by reported co-training failures and latent shortcuts (Ye et al., 2026; Cui et al., 2025; Lian et al., 2026), we scale the action gradient entering PONDER relative to the LM-head path. Appendix A.3 gives the rationale.

3.3 ASYNCHRONOUS SCHEDULE

The model clocks advance independently. Let $\tau_t^{(2)}$ be the source-observation time of PONDER query t, d_t its delay, and $\tau_j^{(1)}$ the time of POUNCE invocation j. The scheduler selects

$$\mathrm{ref}(j) = \max\left\{t : \tau_t^{(2)} + d_t \le \tau_j^{(1)}\right\},$$

and uses null cognition with $\Delta(j) = 0$ if the set is empty. For a valid reference, $\Delta(j) = \tau_j^{(1)} - \tau_{\mathrm{ref}(j)}^{(2)}$. For RoboMME, training samples log-normal intervals around 100 ms for POUNCE and 1 s for PONDER, with a 300 ms compute delay, exposing each state at several ages. The evaluation schedule runs both models at 1 Hz with a fixed 300 ms delay and 20 Hz playback. Appendix A.1 gives the sampling details, while Section 5.2 varies age offline.

3.4 INFERENCE OPTIMIZATION

Append-only `StaticCache` sessions encode only new tokens, keeping cognition-only and 45-token fires at 93.7 and 871.1 ms p95 across 0.8K–14K context tokens. This native transformer KV cache avoids re-encoding causal context and is not a separate episode-memory store or retrieval mechanism. Fused Triton kernels (Ma et al., 2025) reduce POUNCE latency from 142 to 25 ms ($5.7\times$). Appendix D gives profiles and rate budgets; these optimizations support 1 Hz model clocks and 20 Hz playback.

4 EXPERIMENTS

4.1 SETUP

Benchmarks. We evaluate memory-dependent control on RoboMME (Dai et al., 2026) and demonstration-conditioned control on RoboCasa-DC (Son et al., 2026). In RoboMME, the robot must remember an object, location, count, order, or event that was visible earlier but is absent from the current observation. We use the full 16-task suite, organized into Counting, Permanence, Reference, and Imitation, and evaluate both the base ($1\times$) and $9\times$ training-data scales. RoboCasa-DC instead provides a demonstration video of the target task, which the policy combines with the current robot observation and language instruction. We use its Category-Balanced / Cross-Embodiment Demonstration setting across five held-out tasks.

Model configuration. Across both benchmarks, PONDER is initialized from Qwen3.5-9B (Team, 2026), uses one cognition carrier ($K{=}1$), and is trained jointly with POUNCE. On RoboMME, POUNCE is initialized from the 3.6B $\pi_{0.5}$ action model (Intelligence et al., 2025). Subgoal and demonstration-reasoning annotations additionally supervise PONDER. On RoboCasa-DC, POUNCE is initialized from the 3B GR00T N1.5 action model (Bjorck et al., 2025). Because RoboCasa-DC has no such annotations, training uses action supervision alone.

Baseline models. RoboMME comparisons include the current-observation $\pi_{0.5}$, $\pi_{0.5}$ + past actions, SAM2Act+ (Fang et al., 2025), MemER (Sridhar et al., 2026), and FrameSamp+Modul. Human performance and the SimpleSG+Oracle and GroundSG+Oracle results reported by RoboMME serve as references. For a data-matched $9\times$ comparison, we train FrameSamp+Modul with its original configuration and change only the training set to the expanded $9\times$ data. RoboCasa-DC comparisons include Vid2Robot (Jain et al., 2024), UniSkill (Kim et al., 2025), ViVLA (Chen et al., 2025a), and SeeTraceAct (Son et al., 2026).

Evaluation protocol. RoboMME evaluation uses 50 episodes per task, or 800 episodes per run. We report PONDERPOUNCE results as means over three runs on the same scene set. Each RoboCasa-DC run uses 50 episodes for each of five held-out tasks. We average PONDERPOUNCE and its cognition-disabled condition over five runs, while the no-demonstration control uses one run. Appendices B.1 and B.2 provide the full protocols.

4.2 MAIN RESULTS

Table 1 reports RoboMME success across the four task families. PONDERPOUNCE reaches 60.83% at the base data scale and 75.54% with $9\times$ data, exceeding FrameSamp+Modul by 16.32 and 17.66 pp, respectively. At the base data scale, the current-observation $\pi_{0.5}$ reaches 17.93%, and adding past actions raises success only to 19.73%. PONDERPOUNCE leads on Permanence and Reference at both scales, while FrameSamp+Modul remains stronger on Imitation and $9\times$ Counting.

This pattern is consistent with native context helping hidden-state tracking and reference resolution, while direct frame reuse remains useful for imitation-heavy tasks. The same architecture gains 14.71 pp with expanded training data. Figure 3 illustrates how episode context preserves a transient PickHighlight cue after its visual markers disappear.

Table 1: **RoboMME success rate (%) by memory design.** The memory column states how episode history is retained. PONDERPOUNCE reuses native MLLM context rather than adding a purpose-built episode-memory module. Each family averages four tasks (Appendix Table 7). Bold marks the best non-oracle result per scale. * marks results reported by RoboMME (Dai et al., 2026). † marks our FrameSamp+Modul result trained on 9× data.

Method	Episode memory	Counting	Permanence	Reference	Imitation	Average
1× data: no learned episode memory						
$\pi_{0.5}$ *	None	28.78	17.00	17.16	8.78	17.93
$\pi_{0.5}$ + past actions*	Action history	29.09	22.75	15.92	11.17	19.73
1× data: purpose-built memory systems						
SAM2Act+*	SAM2 bank + attention	35.33	26.00	16.83	7.33	21.37
MemER*	Keyframe selector/tracker	48.83	53.16	38.00	29.50	42.38
FrameSamp+Modul*	Frame tokens + modulator	65.22	25.11	36.33	**51.39**	44.51
1× data: pretrained MLLM context						
PONDERPOUNCE (ours)	Native MLLM context	**74.67**	**62.83**	**72.17**	33.67	**60.83**
9× data						
FrameSamp+Modul†	Frame tokens + modulator	**86.00**	24.50	58.00	**63.00**	57.88
PONDERPOUNCE (ours)	Native MLLM context	81.33	**80.17**	**92.67**	48.00	**75.54**
Oracle / human references						
Human*	Human memory	88.50	91.00	93.00	89.50	90.50
SimpleSG+Oracle*	Oracle subgoal	82.56	21.56	32.28	61.94	49.58
GroundSG+Oracle*	Grounded oracle subgoal	83.86	93.31	95.16	63.98	84.08

Table 2 reports results on RoboCasa-DC. PONDERPOUNCE reaches 12.5%, compared with 11.6% for SeeTraceAct, the strongest published baseline. Replacing cognition with the learned null state lowers success to 8.6%, while the single-run no-demonstration control reaches 9.0%. These results support interface transfer and show that control depends on cognition, but the low absolute success rates and unavailable baseline uncertainty preclude a broad superiority claim.

Table 2: **RoboCasa-DC success rate (%).** Five held-out tasks. PONDERPOUNCE reports five-run mean ± s.d. Published uncertainty is unavailable. ‡ marks one run.

Method	Average success (%)
Vid2Robot	8.8
UniSkill	11.2
ViVLA	8.0
SeeTraceAct	11.6
PONDERPOUNCE (ours)	12.5 ± 0.9
PONDERPOUNCE, cognition disabled	8.6 ± 0.4
PONDERPOUNCE, no demonstration	$9.0^{\ddagger}$

5 ANALYSIS OF THE COGNITION CHANNEL

The main results establish benchmark performance but do not identify which factors make the cognition channel useful or how POUNCE uses it over time. We therefore probe three questions: how supervision and channel type affect downstream control, whether cognition must be refreshed within a subgoal, and how System 2 scale and initialization affect performance. This matters because latent

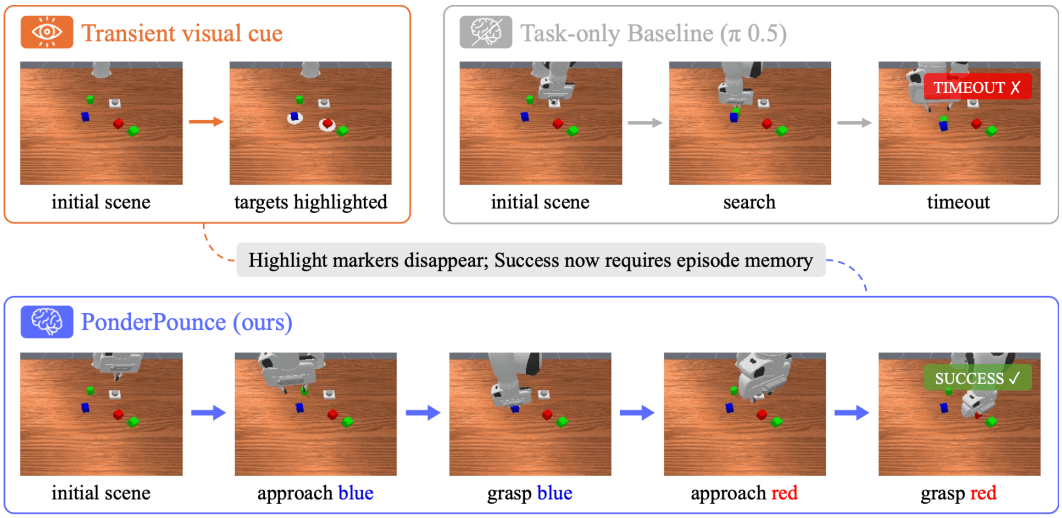

Figure 3: **Qualitative RoboMME PickHighlight.** The task is *"Press the button, pick all cubes that were highlighted, then stop."* Pressing the button briefly marks the blue and red targets, after which the markers disappear. The current-observation $\pi_{0.5}$ searches and times out, whereas PONDERPOUNCE retains the observed cue in its episode context and grasps both targets to complete the task.

channels can collapse to instruction-level summaries that are insensitive to visual changes (Cui et al., 2025).

5.1 SUPERVISION AND THE COGNITION INTERFACE

We first examine LM-head supervision. Removing only the demonstration-reasoning targets while retaining joint training, the transition gate, and subgoal text lowers average RoboMME success from 60.83% to 48.21%. Removing all LM-head grounding lowers it to 27.96% (Table 3). Because the LM head and cognition carriers share the same transformer trunk, these gains may reflect richer cognition or improved gate and subgoal generation. The ablations do not distinguish these mechanisms.

We next compare the jointly trained continuous interface with a separately adapted subgoal-text reference that replaces cognition with the latest subgoal text and updates POUNCE only at predicted transitions. Demonstration reasoning remains internal to PONDER. The reference reaches 59.96% versus 60.83% for continuous cognition, a 0.87 pp gap smaller than the observed 2.63 pp spread across base-scale evaluation runs (Appendix B.1). Continuous cognition is therefore accuracy-competitive rather than superior.

Table 3: **Supervision and context-to-control variants.** RoboMME success rate (%) at 1× data. The three continuous-cognition rows are jointly trained and form matched supervision ablations. The subgoal-text reference is separately adapted, and FrameSamp+Modul is an external memory reference. Bold marks the best value in each column.

Variant	Training	LM-head targets	Counting	Permanence	Reference	Imitation	Average
Continuous cognition (ours)	joint	transition + subgoal text + demo reasoning	**74.67**	**62.83**	72.17	33.67	**60.83**
w/o demo reasoning	joint	transition + subgoal text	70.50	60.00	36.67	25.67	48.21
w/o LM-head grounding	joint	none	49.34	27.17	21.50	13.84	27.96
Subgoal-text reference	separate	transition + subgoal text + demo reasoning	62.00	61.67	**73.34**	42.83	59.96
FrameSamp+Modul	–	–	65.22	25.11	36.33	**51.39**	44.51

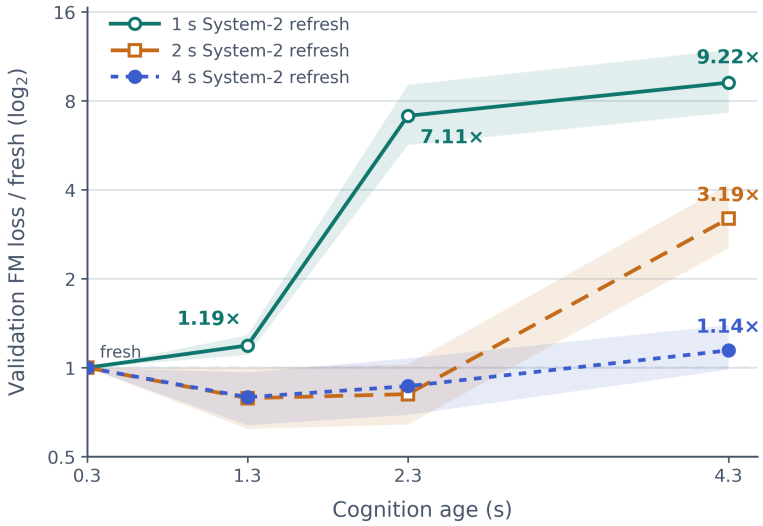

Figure 4: **Measured staleness under slow-refresh training.** The curves are checkpoints trained with 1, 2, or 4 s System 2 refresh intervals. Bands show 95% task-stratified bootstrap CIs. The $\log_2$ axis shows validation flow-matching loss normalized by each checkpoint's own 0.3 s condition. Lower is better.

For the continuous checkpoint, cognition-only fires provide the per-query refresh required in Section 5.2 at 78 ms p50. Matching that cadence with a forced 45-token subgoal-text decode would take 0.82 s per fire (Section 3.4). Grounding is not universally required for cognition to affect control. RoboCasa-DC provides no subgoal or demonstration-reasoning annotations, yet replacing cognition with the learned null state lowers success from 12.5% to 8.6%.

5.2 Cognition Refresh and Staleness

If cognition only encodes the current subgoal, holding it between predicted transitions should preserve performance. We keep running PONDER at every query, but POUNCE receives only the first cognition and subsequent transition-time states. The last transmitted state is reused otherwise, while the reported age remains fixed at the standard 300 ms. This intervention reduces success from 60.83% to 1.83% (Table 4), showing that the reported checkpoint depends on within-subgoal refreshes. However, held cognition becomes older than its age input indicates, and predicted transitions may be missed. The result therefore does not rule out sparse delivery when training matches the interface. The separately trained subgoal-text reference updates POUNCE only at predicted transitions and reaches 59.96% (Table 3).

Table 4: **The reported checkpoint requires within-subgoal refreshes.** Same-checkpoint RoboMME intervention at 1× data. "Held" reuses the last transition-time state while all other inputs remain fixed.

Refresh	S1 receives	Counting	Permanence	Reference	Imitation	Average
Every query	latest $\mathbf{C}_t$	74.67	62.83	72.17	33.67	60.83
Transitions only	held transition $\mathbf{C}$	0.17	0.17	0.67	6.33	1.83

We then measure sensitivity to stale cognition using a teacher-forced offline diagnostic (Figure 4). On identical held-out ticks, we replace $\mathbf{C}_t$ with $\mathbf{C}_{t-k}$ for $k \in \{0, 1, 2, 4\}$ and report its true age of $0.3 + k$ seconds. All inputs except cognition and its reported age remain fixed, as do the targets and flow draws. The same 757 ticks from 80 task-balanced episodes are used for every condition.

For the 1 s-refresh checkpoint, normalized loss rises from $1.00\times$ at 0.3 s to $7.11\times$ at 2.3 s and $9.22\times$ at 4.3 s. At 4.3 s, the checkpoints trained with 2 and 4 s refresh intervals reduce this value to $3.19\times$ and $1.14\times$. Robustness to stale cognition trades off against fresh-condition fit. Absolute loss at 0.3 s rises from 0.117 for 1 s training to 0.213 and 0.232 for 2 and 4 s training. Slow-refresh training also reduces the number of System 2 queries and grounding updates, so this experiment does not isolate refresh interval from supervision frequency. Cognition and its reported age change together, so the diagnostic also does not isolate their contributions. It remains an offline loss diagnostic rather than a closed-loop success measurement.

5.3 Does Pretrained Context Capacity Transfer to Control?

Purpose-built memory systems introduce history-specific mechanisms whose sampling, storage, retrieval, or compression behavior must be designed or learned for robot trajectories. PONDER-POUNCE instead starts from an MLLM already pretrained to integrate multimodal context and adapts a continuous readout of its native causal context to control. This view predicts that increasing System 2 capacity while holding the controller architecture and interface constant should improve control.

We test this prediction by replacing pretrained Qwen3.5-9B with Qwen3.5-0.8B while retaining the same POUNCE architecture and context-to-control interface. The pretrained 9B model reaches 60.83%, 10.79 pp above the pretrained 0.8B model at 50.04% (Table 5). The result shows that a larger pretrained context engine can improve control through the same interface while remaining off the fast action path.

Pretrained initialization is also operationally important. Randomly initializing the 9B MLLM makes training unstable and yields 0.00% success. Because this run does not converge, it does not isolate the benefit of pretraining under matched optimization, but it shows that the current end-to-end recipe depends on a pretrained context model.

Table 5: **Pretrained context capacity transfers to control.** Average RoboMME success rate (%) at $1\times$ data. The pretrained rows report three-run means. The POUNCE architecture and interface are unchanged across model sizes. Randomly initialized 9B training was unstable and reached zero success.

PONDER (System 2)	Initialization	Average
Qwen3.5 9B	pretrained	**60.83**
Qwen3.5 0.8B	pretrained	50.04
Qwen3.5 9B	random	0.00

6 Conclusion

PONDERPOUNCE treats memory as pretrained contextual capacity rather than a new purpose-built robotics mechanism. It reuses native MLLM context and routes one asynchronous cognition token to a VLA without adding a purpose-built episode-memory store or retrieval path. It reaches 60.83%/75.54% on RoboMME versus FrameSamp+Modul's 44.51%/57.88%, while remaining weaker on Imitation. With the same controller architecture and interface, increasing the pretrained context engine from 0.8B to 9B improves success by 10.79 pp, showing that the fast controller benefits from greater System 2 capacity. Continuous cognition remains competitive with the separately trained transition-only subgoal-text interface. Grounding and demonstration-reasoning supervision improve control, and the reported checkpoint depends on timely within-subgoal refreshes. The interface also transfers to RoboCasa-DC, where cognition affects control without LM-head grounding. Together, these findings support pretrained MLLM context as a strong but not universal or compute-free memory substrate for robot control.

7 LIMITATIONS

RoboMME provides simulator-derived gates, subgoals, and demonstration-reasoning targets whose production cost is not measured, while published baselines do not receive the same reasoning supervision. The comparisons therefore reflect both architecture and supervision. Our evaluation is limited to two simulated benchmarks and one cognition carrier per query ($K{=}1$). RoboCasa-DC further covers five challenging held-out tasks with modest absolute success, providing initial evidence of interface transfer rather than broad cross-embodiment generalization.

Pairing a 9B context model with a 3–3.6B controller incurs additional training and inference cost. The latency profiles characterize batch-1 calls rather than concurrent throughput, and the append-only context is limited to 16K tokens. Our interventions establish that cognition contributes to control, while its representation and use by POUNCE remain open. In particular, the held-state intervention does not evaluate a policy trained for sparse transmission, the staleness diagnostic uses teacher-forced loss, and unstable optimization of the randomly initialized model prevents a matched-convergence comparison of pretraining.

8 FUTURE WORK

Future work should compare architectures under matched supervision and annotation cost. Labels generated or propagated from vision-language models may reduce dependence on simulator annotations, but their effect should be tested across label quality and production cost (Xiao et al., 2022; Zhao et al., 2025; Feng et al., 2026). Counterfactual pairs that preserve the current scene while changing relevant history, richer context-type breakdowns, additional embodiments, and real-robot experiments would better isolate memory use and establish generalization beyond the present simulated benchmarks.

System-level studies should compare energy and concurrent throughput under matched compute and evaluate smaller, distilled, or quantized context models. Closed-loop refresh-period sweeps, policies trained explicitly for sparse cognition delivery, representation probes, cognition-width sweeps, and broader scale and pretraining controls would clarify what the channel encodes and how POUNCE uses it. Scaling soft-token learning may also require longer, temporally coherent vision–action data in which behavior depends on memory and extended intent, with desktop and game interaction providing useful complementary settings (Choi et al., 2026; Magne et al., 2026).

ACKNOWLEDGMENTS

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2026-25522885, Development of a World Foundation Model for Training and Deployment of Physical AI Systems).

REFERENCES

Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quon Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. Rt-h: Action hierarchies using language. *arXiv preprint arXiv:2403.01823*, 2024.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

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

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X Lee, Maria Bauzá, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. Robocat: A self-improving generalist agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first international conference on machine learning*, 2024.

Qingwen Bu, Hongyang Li, Li Chen, Jisong Cai, Jia Zeng, Heming Cui, Maoqing Yao, and Yu Qiao. Towards synergistic, generalized, and efficient dual-system for robotic manipulation. *arXiv preprint arXiv:2410.08001*, 2024.

Guangyan Chen, Meiling Wang, Qi Shao, Zichen Zhou, Weixin Mao, Te Cui, Minzhao Zhu, Yinan Deng, Luojie Yang, Zhanqi Zhang, et al. See once, then act: Vision-language-action model with task learning from one-shot video demonstrations. *arXiv preprint arXiv:2512.07582*, 2025a.

Hao Chen, Jiaming Liu, Chenyang Gu, Zhuoyang Liu, Renrui Zhang, Xiaoqi Li, Xiao He, Yandong Guo, Chi-Wing Fu, Shanghang Zhang, et al. Fast-in-slow: A dual-system foundation model unifying fast manipulation within slow reasoning. *arXiv preprint arXiv:2506.01953*, 2025b.

Tongqing Chen, Hang Wu, Jiasen Wang, Xiaotao Li, and Lu Fang. Streamvla: Breaking the reason-act cycle via completion-state gating. *arXiv preprint arXiv:2602.01100*, 2026a.

William Chen, Jagdeep Singh Bhatia, Catherine Glossop, Nikhil Mathihalli, Ria Doshi, Andy Tang, Danny Driess, Karl Pertsch, and Sergey Levine. Steerable vision-language-action policies for embodied reasoning and hierarchical control. *arXiv preprint arXiv:2602.13193*, 2026b.

Yi Chen, Yuying Ge, Yizhuo Li, Yixiao Ge, Mingyu Ding, Ying Shan, and Xihui Liu. Moto: Latent motion token as the bridging language for robot manipulation. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2025c.

Suhwan Choi, Jaeyoon Jung, Haebin Seong, Minchan Kim, Minyeong Kim, Yongjun Cho, Yoonshik Kim, Yu Park, Youngjae Yu, and Yunsung Lee. D2e: Scaling vision-action pretraining on desktop data for transfer to embodied ai. In *International Conference on Learning Representations*, volume 2026, pp. 46207–46236, 2026.

Can Cui, Pengxiang Ding, Wenxuan Song, Shuanghao Bai, Xinyang Tong, Zirui Ge, Runze Suo, Wanqi Zhou, Yang Liu, Bofang Jia, et al. Openhelix: A short survey, empirical analysis, and open-source dual-system vla model for robotic manipulation. *arXiv preprint arXiv:2505.03912*, 2025.

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. *arXiv preprint arXiv:2603.04639*, 2026.

Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *Advances in neural information processing systems*, 30, 2017.

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. *arXiv preprint arXiv:2501.18564*, 2025.

Youhe Feng, Hansen Shi, Haoyang Li, Xinlei Guo, Yang Wang, Chengyang Zhang, Jinkai Zhang, Xiaohan Zhang, Jie Tang, and Jing Zhang. Procvlm: Learning procedure-grounded progress rewards for robotic manipulation. *arXiv preprint arXiv:2605.08774*, 2026.

Figure AI. Helix: A vision-language-action model for generalist humanoid control, 2025. URL `https://www.figure.ai/news/helix`. Blog post.

Letian Fu, Huang Huang, Gaurav Datta, Lawrence Yunliang Chen, William Chung-Ho Panitch, Fangchen Liu, Hui Li, and Ken Goldberg. In-context imitation learning via next-token prediction. *arXiv preprint arXiv:2408.15980*, 2024.

Chenyang Gu, Jiaming Liu, Hao Chen, Runzhong Huang, Qingpo Wuwu, Zhuoyang Liu, Xiaoqi Li, Ying Li, Renrui Zhang, Peng Jia, et al. Manualvla: A unified vla model for chain-of-thought manual generation and robotic manipulation. *arXiv preprint arXiv:2512.02013*, 2025.

ByungOk Han, Jaehong Kim, and Jinhyeok Jang. A dual process vla: Efficient robotic manipulation leveraging vlm. *arXiv preprint arXiv:2410.15549*, 2024.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Jiaheng Hu, Mohit Shridhar, Caden Lu, Dhruv Shah, Hao-Tien Lewis Chiang, Jie Tan, and Annie Xie. What matters in orchestrating robot policies: A systematic study of hierarchical vla agents. *arXiv preprint arXiv:2606.10267*, 2026.

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. *arXiv preprint arXiv:2307.05973*, 2023.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Vidhi Jain, Maria Attarian, Nikhil J Joshi, Ayzaan Wahid, Danny Driess, Quan Vuong, Pannag R Sanketi, Pierre Sermanet, Stefan Welker, Christine Chan, et al. Vid2robot: End-to-end video-conditioned policy learning with cross-attention transformers. *arXiv preprint arXiv:2403.12943*, 2024.

Yunfan Jiang, Yevgen Chebotar, Ruijie Zheng, Fengyuan Hu, Yunhao Ge, Jimmy Wu, Tianyuan Dai, Scott Reed, Li Fei-Fei, Yuke Zhu, et al. Robottt: Context scaling for robot policies. *arXiv preprint arXiv:2607.15275*, 2026.

Daniel Kahneman. Thinking, fast and slow. *Farrar, Straus and Giroux*, 2011.

Hanjung Kim, Jaehyun Kang, Hyolim Kang, Meedeum Cho, Seon Joo Kim, and Youngwoon Lee. Uniskill: Imitating human videos via cross-embodiment skill representations. *arXiv preprint arXiv:2505.08787*, 2025.

Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Caelan Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, et al. Hamster: Hierarchical action models for open-world robot manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 24040–24068, 2025.

Shijie Lian, Bin Yu, Xiaopeng Lin, Laurence T Yang, Zhaolong Shen, Changti Wu, Yuzhuo Miao, Cong Huang, and Kai Chen. Langforce: Bayesian decomposition of vision language action models via latent action queries. *arXiv preprint arXiv:2601.15197*, 2026.

Yudong Liu, Yuan Li, Zijia Tang, Yuxi Zheng, Yueqian Lin, Qinsi Wang, Yi Li, Shuangjun Liu, Shuai Zhang, Taotao Jing, et al. Latent bridge: Feature delta prediction for efficient dual-system vision-language-action model inference. *arXiv preprint arXiv:2605.02739*, 2026a.

Zhuoyang Liu, Jiaming Liu, Hao Chen, Jiale Yu, Ziyu Guo, Chengkai Hou, Chenyang Gu, Xiangju Mi, Renrui Zhang, Kun Wu, et al. Last $_{\{0\}}$: Latent spatio-temporal chain-of-thought for robotic vision-language-action model. *arXiv preprint arXiv:2601.05248*, 2026b.

Yunchao Ma, Yizhuang Zhou, Yunhuan Yang, Tiancai Wang, and Haoqiang Fan. Running vlas at real-time speed. *arXiv preprint arXiv:2510.26742*, 2025.

Loïc Magne, Anas Awadalla, Guanzhi Wang, Yinzhen Xu, Joshua Belofsky, Fengyuan Hu, Joohwan Kim, Ludwig Schmidt, Georgia Gkioxari, Jan Kautz, et al. Nitrogen: An open foundation model for generalist gaming agents. *arXiv preprint arXiv:2601.02427*, 2026.

Yide Shentu, Philipp Wu, Aravind Rajeswaran, and Pieter Abbeel. From llms to actions: Latent codes as bridges in hierarchical robot control. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8539–8546. IEEE, 2024.

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*, volume 2026, pp. 18567–18602, 2026.

Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*, 2025.

Jaehyeon Son, Junhyun Kim, Kyle Kam, Jeremiah Coholich, Seok Joon Kim, Jinhoo Kim, Chris Dongjoo Kim, Jaemin Cho, Dieter Fox, and Zsolt Kira. Seetraceact: Visibility-aware latent planning from cross-embodiment demonstration videos. *arXiv preprint arXiv:2606.02745*, 2026.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Scaling up memory for robotic control via experience retrieval. In *International Conference on Learning Representations*, volume 2026, pp. 97142–97166, 2026.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, et al. Mem: Multi-scale embodied memory for vision language action models. *arXiv preprint arXiv:2603.03596*, 2026.

Yifei Wei, Linqing Zhong, Yi Liu, Yuxiang Lu, Xindong He, Maoqing Yao, and Guanghui Ren. Libra-vla: Achieving learning equilibrium via asynchronous coarse-to-fine dual-system. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 39799–39815, 2026.

Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025.

Ted Xiao, Harris Chan, Pierre Sermanet, Ayzaan Wahid, Anthony Brohan, Karol Hausman, Sergey Levine, and Jonathan Tompson. Robotic skill acquisition via instruction augmentation with vision-language models. *arXiv preprint arXiv:2211.11736*, 2022.

Yi Yang, Jiaxuan Sun, Siqi Kou, Yihan Wang, and Zhijie Deng. Lohovla: A unified vision-language-action model for long-horizon embodied tasks. *arXiv preprint arXiv:2506.00411*, 2025.

Jinhui Ye, Fangjing Wang, Ning Gao, Junqiu Yu, Yangkun Zhu, Bin Wang, Jinyu Zhang, Weiyang Jin, Yanwei Fu, Feng Zheng, et al. St4vla: Spatially guided training for vision-language-action models. *arXiv preprint arXiv:2602.10109*, 2026.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. In *International Conference on Learning Representations*, volume 2025, pp. 28213–28239, 2025.

Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557*, 2018.

Jianke Zhang, Yanjiang Guo, Xiaoyu Chen, Yen-Jen Wang, Yucheng Hu, Chengming Shi, and Jianyu Chen. Hirt: Enhancing robotic control with hierarchical robot transformers. *arXiv preprint arXiv:2410.05273*, 2024.

Yinuo Zhao, Jiale Yuan, Zhiyuan Xu, Xiaoshuai Hao, Xinyi Zhang, Kun Wu, Zhengping Che, Chi Harold Liu, and Jian Tang. Training-free generation of temporally consistent rewards from vlms. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8133–8143. IEEE, 2025.

Minjie Zhu, Yichen Zhu, Jinming Li, Junjie Wen, Zhiyuan Xu, Zhengping Che, Chaomin Shen, Yaxin Peng, Dong Liu, Feifei Feng, et al. Language-conditioned robotic manipulation with fast and slow thinking. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4333–4339. IEEE, 2024.

Teqiang Zou, Hongliang Zeng, Yuxuan Nong, Yifan Li, Kehui Liu, Haotian Yang, Xinyang Ling, Xin Li, and Lianyang Ma. Asynchronous fast-slow vision-language-action policies for whole-body robotic manipulation. *arXiv preprint arXiv:2512.20188*, 2025.

A Additional Method Details

A.1 Schedule Sampling

Training uses the latest-ready pairing rule from Section 3.3. Intervals are log-normal with arithmetic mean m and log-space standard deviation σ, clipped at three log-space standard deviations. For RoboMME, POUNCE uses $(m, \sigma)=(100\,\mathrm{ms}, 0.05)$, PONDER uses $(1\,\mathrm{s}, 0.05)$, and compute delay uses $(300\,\mathrm{ms}, 0.3)$. Ready times preserve query order. Evaluation instead fixes both model clocks at 1 Hz and delay at 300 ms.

A.2 Subgoal and Demonstration-Reasoning Supervision

Let $\mathcal{T}^*$ be the set of annotated transition queries. The teacher-forced transition target in Figure 2 is

$$T_t^* = \begin{cases} T_{\mathrm{Yes}}, & t \in \mathcal{T}^*, \\ T_{\mathrm{No}}, & t \notin \mathcal{T}^*. \end{cases}$$

We map T_{Yes} to <|fim_prefix|> and T_{No} to <|fim_pad|>. Training teacher-forces T_t^* and its payload. An ordinary transition serializes $[\,T_{\mathrm{Yes}}, S_t\,]$, while a non-transition uses T_{No} as the LM target at the first carrier position. At inference, greedy decoding treats only exact T_{Yes} as a transition, and T_{No} terminates payload generation. A learned carrier embedding replaces the ordinary token embedding, so the first carrier is instantiated once in either branch. Subgoal tokens receive cross-entropy, but POUNCE receives only the resulting cognition. Grounded coordinates convert from 256-pixel $\langle y, x\rangle$ to normalized $[x, y] \in [0, 1000]^2$.

Demonstration reasoning. At the first execution transition of demonstration episodes, PONDER emits $[\,T_{\mathrm{Yes}}, \mathrm{DR}, T_{\mathrm{Yes}}, S_t\,]$. The first transition token opens DR, and the second separates it from S_t. Cross-entropy supervises the block once per episode. Programmatic DR targets describe demonstration facts and the implied next step, such as button order or arc direction. At inference, PONDER receives no oracle text: it generates and retains the block internally, passing only $\mathbf{C}_t$ to POUNCE.

The following target is verbatim from a 9× RouteStick episode:

```
<|fim_prefix|> Demo route across the button row (arcs around
obstacle posts in between):  target_0 at <77, 103> → target_1 at
<71, 74> → target_2 at <77, 103>.  Arc directions in order:
clockwise, clockwise.  Wrong arc side drives the tip into the post
--- fail.  Next:  target_0 at <77, 103> → target_1 at <71, 74>, on
the right, circling clockwise around the post between.
<|fim_prefix|> move to the nearest right target by circling around
the stick clockwise
```

A.3 Gradient Scaling

Multiple POUNCE invocations can reference one cognition state, so their action-loss gradients accumulate before reaching PONDER and can dominate LM-head grounding. We therefore scale the action gradient entering PONDER by 0.5.

B BENCHMARK AND EVALUATION DETAILS

B.1 ROBOMME

Task suite and episode representation. RoboMME contains 16 tasks across Counting, Permanence, Reference, and Imitation, with four tasks per family. Episodes pair an observation-only demonstration with an action-labeled execution. Each trajectory contains front and wrist 256×256 RGB, 8D state and action, and pre-action state–image–action alignment at 20 Hz.

Dataset scales. The base dataset contains 1,587 episodes, approximately 100 per task. The $9\times$ dataset is a fresh oracle collection of 14,400 episodes, with 900 solvable episodes per task, rather than duplicated base data.

Reasoning annotations. The reasoning-annotated variant shares trajectories and videos with the action dataset and adds deterministic simulator-template labels rather than human or model annotations. Demonstration reasoning describes observable facts from the demonstration and is attached once at the first execution transition. Episodes without a demonstration contain no demonstration-reasoning target.

Reasoning templates. Table 6 summarizes the four template families. Each uses concrete episode values and image-space $\langle y, x \rangle$ coordinates.

Table 6: **RoboMME reasoning-template schemas.** Braces denote episode-specific values. The actual strings also include the current grounded subgoal.

Family	Context extracted	Representative template schema
Counting	target colors/counts, completed cycles or passes, target coordinates	"Target: {count}; completed: {done}; remaining: {left}. Next, move {object} to {target $\langle y, x \rangle$}."
Permanence	reveal map, target color, ordered container swaps, button gates	"Reveal: container {A} hides {color}. Track swaps {A↔B, . . .}; final target is at {$\langle y, x \rangle$}."
Reference	highlighted set, demonstrated referent, repeat count, demonstrated button/placement order	"Demo selected {referent/order}; after {swaps or ordinal resolution}, choose {current target $\langle y, x \rangle$}."
Imitation	manipulation manner, grasp end and insertion side, button path, route and arc directions	"Demo sequence: {targets with coordinates}; modes: {directions/arcs/grasp side}. Reproduce exactly; next is {step}."

Training configuration. The RoboMME models are trained end to end for 4,320 steps on eight B200 GPUs with seed 42, gradient clipping at 1.0, and global batch 32. The learning rate is 2×10^{-5} with 100 warmup steps followed by a constant schedule, and AdamW uses β_1=0.9, β_2=0.95, ϵ=10^{-8}, and weight decay 10^{-10}. PONDER starts from Qwen3.5-9B and POUNCE from $\pi_{0.5}$. Components frozen by the underlying $\pi_{0.5}$ recipe, including the PaliGemma backbone, remain frozen. The policy predicts 20-step chunks of the 8-dimensional action. Demonstration video is sampled at 1 Hz, and the model jointly learns the action objective, transition gate, grounded subgoal, and first-execution reasoning target with one cognition carrier per query. The action and grounding losses use w_1=1.0 and w_2=0.1. Cognition dropout is disabled. The base and $9\times$ runs use the same recipe and differ only in training data.

Data-matched baseline. For the † FrameSamp+Modul result in Table 1, we retain its original configuration and change only the training data to the $9\times$ collection.

Evaluation protocol. We evaluate 50 episodes per task (800 total) for at most 1,300 environment steps. Unless noted, PONDERPOUNCE reports three-run means. Overall scores are 62.25%, 59.62%, and 60.62% at 1×, and 75.75%, 74.62%, and 76.25% at 9×. The overall spread is 2.63 pp at 1× and 1.63 pp at 9×, while individual tasks vary by up to 16 pp. This variation motivates averaging. Main PONDERPOUNCE results use grounding, while cited baselines retain their published protocols.

Both systems run at 1 Hz; POUNCE chunks play at 20 Hz. Control starts only after the first cognition is ready, so evaluation never acts on $\mathbf{C}_{\varnothing}$. Synchronized clocks and a fixed 300 ms delay give each invocation the newest cognition at age 300 ms. Thus evaluation does not exercise stale-state reuse or the null path used during training.

Per-task results. Table 7 reports the 16 task-level values underlying the family averages in Table 1. The method groups and source markers follow the main table.

Table 7: **RoboMME per-task success rate (%).** Success rate on all 16 tasks, grouped by the benchmark's four memory families. Task-name abbreviations follow RoboMME. Bold marks the best non-oracle result in each column within each data scale. * marks results reported by RoboMME (Dai et al., 2026); † marks our FrameSamp+Modul result trained on the 9× data.

Method	Counting				Permanence				Reference				Imitation				Average
	Bin Fill	Pick Xtimes	Swing Xtimes	Stop Cube	Video Umsk	Button Umsk	Video UmskS	Button UmskS	Pick HighL	Video Repick	Video PlcBtn	Video PlcOrd	Move Cube	Insert Peg	Pattern Lock	Route Stick	
Oracle / human references																	
Human*	96.00	100.00	80.00	78.00	90.00	92.00	92.00	90.00	92.00	92.00	98.00	90.00	90.00	98.00	84.00	86.00	90.50
SimpleSG+Oracle*	85.78	99.78	100.00	44.67	33.11	22.00	15.56	15.56	44.00	27.78	31.33	26.00	87.33	10.00	95.33	55.11	49.58
GroundSG+Oracle*	85.78	100.00	100.00	49.67	98.78	95.00	99.22	80.22	83.33	97.33	100.00	100.00	87.78	15.56	97.00	55.56	84.08
1× training data																	
$\pi_{0.5}$ *	30.00	42.89	35.56	6.67	20.44	22.22	18.67	6.67	11.33	0.44	31.11	25.78	26.00	1.56	2.89	4.67	17.93
$\pi_{0.5}$ + past actions*	26.67	58.33	26.67	4.67	30.67	23.67	20.67	16.00	12.33	8.67	24.00	18.67	34.00	1.00	4.00	5.67	19.73
SAM2Act+*	40.00	76.00	25.33	0.00	27.33	32.00	18.00	26.67	17.33	5.33	24.67	20.00	29.33	0.00	0.00	0.00	21.37
MemER*	**56.67**	79.33	59.33	0.00	81.33	72.00	**38.00**	21.33	70.67	25.33	30.00	26.00	**82.67**	6.67	16.67	12.00	42.38
FrameSamp+Modul*	39.56	87.33	**92.00**	42.00	32.67	25.11	24.44	18.22	22.89	30.44	60.00	32.00	77.78	**7.56**	**53.56**	**66.67**	44.51
PONDERPOUNCE (ours)	50.67	**93.33**	72.00	**82.67**	**94.67**	**97.33**	25.33	**34.00**	**85.33**	**40.67**	**83.33**	**79.33**	81.33	6.00	12.67	34.67	**60.83**
9× training data																	
FrameSamp+Modul†	**74.00**	96.00	**98.00**	**76.00**	28.00	22.00	20.00	28.00	32.00	42.00	80.00	78.00	92.00	8.00	**66.00**	**86.00**	57.88
PONDERPOUNCE (ours)	**74.00**	**98.00**	93.33	60.00	**96.67**	**100.00**	**73.33**	**50.67**	**92.67**	**88.67**	**92.00**	**97.33**	**93.33**	**18.67**	25.33	54.67	**75.54**

B.2 ROBOCASA-DC

Episode representation. RoboCasa-DC pairs each mobile Franka (PandaOmron) execution with a human-teleoperated Fourier GR-1 demonstration of the same task. Following SeeTraceAct (Son et al., 2026), PandaOmron executions provide three RGB views, 53D proprioception, and 12D relative actions, while demonstrations provide one task-selected GR-1 view.

Task split. We train on 19 tasks and reserve five non-overlapping category-balanced tasks, each with a same-family training counterpart (Table 8). There is no within-task validation split. Roughly 100 paired episodes per task yield 1,900 training and 500 held-out pairs.

Table 8: **RoboCasa-DC task split.** Nineteen seen tasks and five category-balanced unseen tasks.

Family	Seen (train)	Unseen (held-out)
Pick-and-place	PnPCabToCounter, PnPCounterToCab, PnPCounterToSink, PnPCounterToStove, PnPMicrowaveToCounter, PnPSinkToCounter, PnPStoveToCounter	PnPCounterToMicrowave
Doors	CloseDoubleDoor, CloseSingleDoor, OpenSingleDoor	OpenDoubleDoor
Drawer	OpenDrawer	CloseDrawer
Coffee	CoffeePressButton, CoffeeSetupMug	CoffeeServeMug
Faucet / knobs	TurnOnSinkFaucet, TurnSinkSpout, TurnOnMicrowave, TurnOffMicrowave, TurnOnStove, TurnOffStove	TurnOffSinkFaucet

Model and training configuration. RoboCasa-DC has no subgoal or demonstration-reasoning annotations, so we disable LM-head grounding and set $w_2{=}0$. PONDER uses one cognition car-

rier per query, conditioned on the task, a 1 Hz GR-1 demonstration, and sampled PandaOmron observations. Cognition dropout substitutes the null state on 15% of training ticks. POUNCE is GR00T N1.5-3B with an Eagle 2B backbone and a 16-step flow-matching head. Cognition projects to Eagle's 2,048D width with weights initialized from a zero-mean normal distribution with standard deviation 10^{-3} and zero bias. Training invokes System 1 every 2 s and System 2 every 4 s with a 300 ms delay model.

We train end to end for 4,000 steps on eight B200 GPUs with clipping at 1.0 and global batch 32. The learning rate is 1×10^{-5} with 100 warmup steps then constant. AdamW uses (β_1, β_2)=$(0.9, 0.95)$, ϵ=10^{-8}, and weight decay 10^{-10}. Only flow matching supervises the model (w_1=1, w_2=0), and Qwen3.5-9B is fully fine-tuned without LoRA.

Evaluation protocol. We follow the Category-Balanced / Cross-Embodiment protocol on five held-out tasks. Each run restores 50 predefined PandaOmron scenes per task and pairs each with the same-index GR-1 demonstration, filtering unusable pairs before rollout. Episodes run for at most 1,000 steps. We average equally across tasks and report five-run mean and standard deviation. Cognition disabled changes only the transmitted state to the learned null value.

C SELECTED CONTEXT-TO-CONTROL ARCHITECTURES

Table 9 compares contextual-model scale, control channel, training regime, episode/demo context, and channel analysis for representative methods.

Adjacent context mechanisms. Coconut (Hao et al., 2024) recurs hidden states within a language model, latent-action and world-model approaches learn action or motion tokens from video (Ye et al., 2025; Chen et al., 2025c; Bruce et al., 2024), and RoboTTT (Jiang et al., 2026) compresses history into fast weights updated at deployment. These mechanisms are related to recurrent or latent context processing but do not expose a decoded or continuous activation-level interface from a contextual module to an action controller, so they are not included in Table 9.

Table 9: **Selected context-to-control architectures.** Backbones and scales follow each method's original paper rather than benchmark-specific reproductions. We compare contextual and action backbones, scale, training regime, episode/demo context use, and channel analysis. S2/S1: System 2 contextual component/System 1 action component; P/C: perceptual/cognitive; MoT: mixture-of-transformers; hetero-freq: heterogeneous-frequency training; TF: transformer; DiT: diffusion transformer; C2F: coarse-to-fine; HL/LL: high-/low-level; e2e: end-to-end. "—" denotes a mechanism the system does not have or a detail its paper does not report. §: sourced from a company blog post rather than a peer-reviewed paper.

	S2 Backbone	S2 Scale (B)	S1 Backbone	Joint Train	Context Use	Channel Analysis
Frozen or partially frozen S2						
DP-VLA (Han et al., 2024)	OpenVLA	7	scratch TF	frozen	×	—
HiRT (Zhang et al., 2024)	InstructBLIP	7	scratch RT-1	LoRA	×	—
LCB (Shentu et al., 2024)	LLaVA	7	low-level policy	e2e tune	×	—
Latent Bridge (Liu et al., 2026a)	GR00T-N1.6/$\pi_{0.5}$	3	host head	post-hoc	×	—
RFST (Zhu et al., 2024)	VLM (LoRA)	7	scratch policy	LoRA	×	—
Joint training, separate models						
π_0 (Black et al., 2024)	PaliGemma	3	flow-match head	sync	×	—
GR00T N1 (Bjorck et al., 2025)	Eagle VLM	2	DiT	sync	×	—
OpenHelix (Cui et al., 2025)	LLaVA	7	TF policy	co-train	×	semantic
RoboDual (Bu et al., 2024)	OpenVLA	7	diffusion expert	specialist	×	—
DexVLA (Wen et al., 2025)	Qwen2-VL	2	ScaledDP 1B	curric.	×	—
ST4VLA (Ye et al., 2026)	Qwen2.5-VL	3	DiT expert	2-stage	×	—
DuoCore-FS (Zou et al., 2025)	VLM	3	action expert	co-train	×	—
StreamVLA (Chen et al., 2026a)	shared	3	flow-match	gated	×	—
Libra-VLA (Wei et al., 2026)	InternVL2.5	2	action refiner	C2F	×	—
Helix§ (Figure AI, 2025)	open VLM	7	80M enc-dec	e2e	×	—
MemoryVLA (Shi et al., 2026)	VLM	7	diffusion expert	e2e	✓	P/C token ablations
Joint training, shared / coupled parameters						

	S2 Backbone	S2 Scale (B)	S1 Backbone	Joint Train	Context Use	Channel Analysis
FiS (Chen et al., 2025b)	shared VLM	7	shared	co-train	×	—
LaST$_0$ (Liu et al., 2026b)	Janus-Pro (MoT)	1.5	MoT expert	hetero-freq	×	—
Text-hierarchy (discrete subgoal channel)						
Hi Robot (Shi et al., 2025)	PaliGemma	3	VLA policy	2-tier	×	—
RT-H (Belkhale et al., 2024)	PaLI-X	55	action TF	joint	×	—
HAMSTER (Li et al., 2025)	VILA-1.5	13	action expert	2-tier	×	—
Steerable Pol. (Chen et al., 2026b)	Prismatic	7	VLA	2-tier	×	—
LoHoVLA (Yang et al., 2025)	PaliGemma (shared)	3	shared	joint	×	—
ManualVLA (Gu et al., 2025)	Janus-Pro (MoT)	1.5	MoT actor	MoT	×	—
MemER (Sridhar et al., 2026)	Qwen2.5-VL	7	$\pi_{0.5}$	2-tier	✓	—
MEM (Torne et al., 2026)	$\pi_{0.6}$ HL	—	$\pi_{0.6}$ LL	2-tier	✓	—
What-Matters (Hu et al., 2026)	Gemini 2.5	—	GROD family	2-tier	×	planner-context ablations
PONDERPOUNCE (ours)	**Qwen3.5**	**9**	$\pi_{0.5}$ 3.6B / GR00T-N1.5 3B	e2e	✓	**discrete / continuous**

D SERVING MEASUREMENTS

We profile both systems on one H100 and additionally profile POUNCE on one A100 (bf16, batch 1) after warmup, synchronizing CUDA around each call. POUNCE p50 covers 50 calls; PONDER p50 and p95 cover the sweeps below.

Table 10: **Target vs. sustainable output rate per clock** (single H100, bf16; per-call p50 in parentheses). Unoptimized: eager POUNCE, per-fire context re-encoding for PONDER. Optimized: fused kernels for POUNCE; session-resident `StaticCache` and `torch.compile` for cache-only language-model appends in PONDER. Bold: optimized rates meeting the target; †: below target.

Clock	Target	Sustainable rate (per-call p50)	
		Unoptimized	Optimized
Action playback from chunks	20 Hz	20 Hz (not model-bound)	
System 1: POUNCE invocation	1 Hz	7.1 Hz (142 ms)	**40 Hz** (25 ms)
System 2: cognition fire	1 Hz	1.9 Hz (521 ms)	**12.8 Hz** (78 ms)
+45-token subgoal	1 Hz	0.31 Hz† (3.3 s)	**1.2 Hz** (0.82 s)

POUNCE. The production shape uses three 224 px views, a 60-token prompt, 10 flow-matching steps, a 20-step chunk, and a $K{=}8$ prefix (an upper bound on the trained $K{=}1$). Fused kernels are $1.8\times$ faster than the best `torch.compile` mode on H100 and $1.4\times$ on A100, with 4×10^{-3} bf16 action MAE versus eager. Removing cognition changes latency by less than 0.2 ms.

PONDER. The workload uses 10 demonstration frames, two 256 px cameras, and a 16K cache. Context grows from 0.8K to 14K tokens across 100 cognition-only and 50 forced 45-token subgoal fires per configuration. Cached latency stays nearly flat from the early to late sweep, whereas re-encoding grows with context.

Table 11: **POUNCE invocation latency (ms, p50) by serving path.** Bold marks the fastest path.

Serving path	H100	A100
HF eager	141.7	235.5
`torch.compile`, reduce-overhead	43.9	66.6
`torch.compile`, max-autotune	44.1	68.1
Fused Triton kernels (ours)	**24.9**	**48.5**

Table 12: **PONDER per-fire latency (ms) by serving path.** Bold marks the fastest path.

	Cognition-only fire		45-token subgoal fire	
Serving path	p50	p95	p50	p95
Re-encode context	521	1,004	3,269	7,024
Session cache (ours)	**77.9**	**93.7**	**820.2**	**871.1**