RECOMPOSE AND REFINE LATENT REASONING FLOWS FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent reasoning enables vision-language-action (VLA) models to transform multimodal observations into task-relevant internal states before generating continuous robot actions. While existing methods learn to generate or refine such states for each policy query, they discard successful reasoning after execution and therefore reconstruct similar computation from scratch. We present *Reasoning and Flow Memory* (FLOWMEM), a unified VLA model that turns successful latent computation into reusable reasoning experience. Rather than appending a fixed retrieved context, FLOWMEM dynamically retrieves and recomposes compatible latent fragments as the embodied context evolves, forming a reasoning route that follows the temporal structure and progress of successful computation. The route is then refined using current visual and proprioceptive evidence before it conditions action generation. Experiments on RoboMME and LIBERO-Plus show that FLOWMEM attains 48.0% and 77.3% success—1.7 and 4.1 points above memory-free policies, respectively—demonstrating the value of reusing successful latent computation for closed-loop VLA control. Anonymous code is available online.

1 INTRODUCTION

Vision-language-action (VLA) models provide a unified interface between multimodal perception, language understanding, and continuous robot control. Recent work has moved beyond direct observation-to-action prediction by introducing intermediate reasoning before action generation. Explicit approaches verbalize embodied plans or predict future visual goals, while latent-reasoning VLAs internalize such computation into compact continuous states (Zawalski et al., 2024; Zhao et al., 2025; Huang et al., 2025). In particular, LaRA-VLA combines textual and visual prediction in latent reasoning slots, LaST$_0$ learns a spatio-temporal latent chain-of-thought from future physical states, and RD-VLA improves action prediction through recurrent latent computation (Bai et al., 2026; Liu et al., 2026; Tur et al., 2026). These advances establish latent reasoning as an effective intermediate representation between perception and action, avoiding the latency and representational bottlenecks of lengthy explicit reasoning traces.

However, latent reasoning remains largely transient. As illustrated in Fig. 1, a VLA constructs an intermediate trajectory from the current observation, uses it to generate actions, and discards the computation afterward. Even when the robot later encounters a familiar task stage, the successful reasoning that previously connected perception to control cannot be directly reused. Memory-augmented VLAs address a related but different limitation by preserving textual notes, perceptual-cognitive evidence, retrieved observations, or recurrent policy states (Haresh et al., 2026; Shi et al., 2026b; Cherepanov et al., 2026; Li et al., 2026b). Such memories enrich the historical context available to the policy, but they do not retain how latent reasoning itself evolved to-

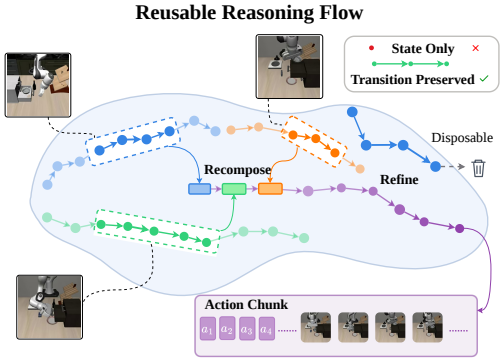

Figure 1: Comparison between transient latent reasoning and the proposed reusable reasoning-flow formulation.

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.

ward successful control. This leaves a central gap between *reasoning before acting* and *learning from how successful reasoning unfolded*.

Reusing latent reasoning is fundamentally different from retrieving a similar embedding. Its utility depends on task progress: a nearby state carries neither the direction of computation nor its unfinished portion. We therefore represent successful embodied reasoning as a *latent reasoning flow*, an ordered, progress-aware trajectory of latent computation.

This perspective raises a new question: *can a VLA construct its current reasoning process from previously successful latent computation, rather than regenerating it from scratch?* Figure 1 exposes the three requirements behind this question. First, retrieval must identify a compatible *transition fragment*, since a nearby latent point alone carries neither progress nor direction. Second, no single episode need cover the unfinished task, requiring complementary fragments to be recomposed without destroying their internal order. Third, the composed route is only prior computation: differences in object pose, layout, and robot state must be corrected by the current observation before the route can condition control.

Building on this formulation, we propose *Reasoning and Flow Memory* (FLOWMEM), a unified VLA model that transforms successful latent computation into reusable reasoning experience. FLOWMEM comprises four coordinated components. The *Embodied Context Encoder* represents the current observation, robot state, and instruction. The *Reasoning Flow Memory* organizes successful latent states into ordered and grounded experience rather than isolated memory vectors. Conditioned on the current embodied context, the *Recompose-and-Refine Reasoning Expert* retrieves anchor-aligned suffixes, constructs a compatible multi-source route, and refines it for the current scene. Finally, the *Memory-Adaptive Action Expert* injects the refined route through the VLA's native latent interface to produce executable action chunks. All four components communicate through a shared latent representation and form a single perception–reasoning–action model.

Recomposition determines which prior transitions support the unfinished task, while refinement determines how the resulting route should change under current embodied evidence. The route is compressed into fixed reasoning tokens and corrected by a fixed-depth residual block before action generation.

We evaluate FLOWMEM on RoboMME (Dai et al., 2026) and LIBERO-Plus (Fei et al., 2026). These benchmarks jointly assess memory-dependent manipulation across temporal, spatial, object, and procedural requirements, as well as the reuse of latent reasoning under systematic distribution shifts. Extensive experiments compare FLOWMEM with representative latent-reasoning and memory-augmented VLAs, while controlled memory interventions examine how its behavior changes with ordered flow recomposition and current-context refinement. Beyond task success, we further evaluate reasoning reuse and computational efficiency.

Our contributions are summarized as follows:

- We introduce latent reasoning flows as a reusable representation of successful embodied computation, capturing how internal reasoning evolves across task progress rather than reducing experience to isolated vectors or historical policy context.
- We propose FLOWMEM, a unified VLA model whose Recompose-and-Refine Reasoning Expert constructs task-specific reasoning from multiple experiences and refines it through current-context residual refinement before action generation.
- Empirical results on RoboMME and LIBERO-Plus indicate that treating successful latent computation as ordered reasoning experience can improve closed-loop VLA control, with gains of 1.7 and 4.1 points, respectively.

2 PRELIMINARIES

2.1 VISION-LANGUAGE-ACTION MODELS WITH LATENT REASONING

Given an RGB observation o_t, robot state s_t, and language instruction l, a vision-language-action (VLA) model predicts a chunk of H future actions $\mathbf{a}_{t:t+H-1} \in \mathbb{R}^{H \times d_a}$. Each action commonly specifies end-effector translation, rotation, and gripper control. Modern VLAs often generate continuous actions with diffusion or flow-matching objectives, while latent-reasoning VLAs additionally

introduce an intermediate state $\mathbf{Z}_t \in \mathbb{R}^{L\times d}$ between multimodal context and action generation (Liu et al., 2026; Bai et al., 2026; Tur et al., 2026):

$$\mathbf{Z}_t = R_\theta(o_t, s_t, l), \qquad \hat{\mathbf{a}}_{t:t+H-1} = A_\theta(o_t, s_t, l, \mathbf{Z}_t). \tag{1}$$

Here, R_θ denotes the native latent-reasoning module, A_θ the action expert, and $\mathbf{Z}_t$ the model's internal task-relevant computation, rather than an explicit textual plan, subgoal label, or action sequence. Existing methods generate or iteratively refine this state for the current policy query; the resulting computation is normally discarded after action prediction.

2.2 Problem Formulation

We consider language-conditioned robotic manipulation as a partially observed sequential decision problem. An episode is $\zeta = (l, \{o_t, s_t, \mathbf{a}_t\}_{t=1}^{T}, y)$, where $y \in \{0, 1\}$ denotes task success. Let $\mathcal{D}^+ = \{\zeta_i \mid y_i = 1\}$ contain successful training episodes and let $\mathcal{M} = \Phi(\mathcal{D}^+)$ be reusable memory constructed from their latent reasoning traces. Our goal is to learn a unified VLA policy

$$(\mathbf{Z}_t, \hat{\mathbf{a}}_{t:t+H-1}) = \Pi_\theta(o_t, s_t, l, \mathcal{M}) \tag{2}$$

where Π_θ denotes the memory-conditioned VLA policy that uses compatible prior computation to improve expected task success under the deployment distribution. Memory informs the current latent reasoning state, while every predicted action remains conditioned on the current observation and robot state.

At evaluation time, the policy may access only the current embodied context and memory committed before the evaluated episode. Memory-source and evaluation episodes are disjoint; future frames, rewards, success labels, simulator-privileged states, and retrieval from the currently executing episode are unavailable. Under these constraints, the problem is to reuse successful latent computation without sacrificing closed-loop adaptation to the present physical state.

3 The Proposed Model

3.1 Overview

Given an RGB observation o_t, robot state s_t, and language instruction l, a VLA policy predicts an action chunk $\hat{\mathbf{a}}_{t:t+H-1}$. We augment this mapping with a reasoning flow memory $\mathcal{M}$ that stores latent reasoning produced during successful interactions:

$$\hat{\mathbf{a}}_{t:t+H-1} = \Pi_\theta(o_t, s_t, l, \mathcal{M}). \tag{3}$$

where Π_θ denotes the memory-conditioned policy and $\mathcal{M}$ the reasoning flow memory. The central premise of FlowMem is that latent reasoning is reusable computation. Instead of retrieving past observations or replaying past actions, the model retrieves ordered latent fragments that describe how successful reasoning evolved, recomposes compatible fragments into a route for the current task, and refines the query-local route tokens using the current embodied context.

As illustrated in Fig. 2, FlowMem contains four components. The *Embodied Context Encoder* represents the current observation, state, and instruction. The *Reasoning Flow Memory* stores ordered latent fragments from successful source episodes together with compact retrieval and transition metadata. The *Recompose-and-Refine Reasoning Expert* performs anchor-aligned fragment retrieval, route tokenization, and current-context refinement. Finally, the *Memory-Adaptive Action Expert* injects the refined reasoning tokens through the native latent interface of the VLA and generates continuous actions.

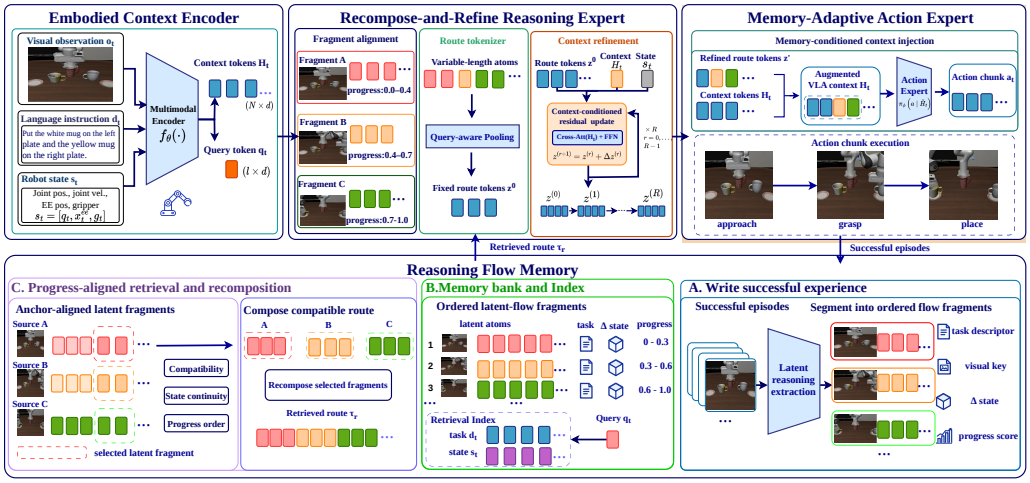

Figure 2: Overview of FLOWMEM. Successful latent reasoning is stored as ordered flow fragments, retrieved and recomposed according to the current embodied context, and refined before conditioning action generation.

3.2 EMBODIED CONTEXT ENCODER

The Embodied Context Encoder maps the current multimodal input to contextual tokens

$$\mathbf{H}_t = E_\theta(o_t, s_t, l) \in \mathbb{R}^{N_c \times d}. \tag{4}$$

where N_c is the number of contextual tokens and d is their latent width. These tokens are used both by the action policy and by the memory pathway. For retrieval, we derive a compact visual semantic descriptor $\mathbf{d}_t$ from the observable context and pair it with a normalized robot-state vector $\bar{\mathbf{s}}_t$. The resulting query is

$$\mathbf{q}_t = (\mathbf{d}_t, \bar{\mathbf{s}}_t, l). \tag{5}$$

Only information available at the current policy query is used online. Future observations, rewards, and success labels are used solely to construct or supervise memory offline.

3.3 REASONING FLOW MEMORY

For each successful source episode i, we record the latent reasoning states produced at successive policy queries,

$$\mathcal{Z}_i = [\mathbf{Z}_{i,1}, \dots, \mathbf{Z}_{i,T_i}]. \tag{6}$$

where T_i is the number of policy queries in source episode i. We organize each sequence into contiguous fragments, forming ordered memory units for subsequent retrieval, recomposition, and context refinement,

$$\mathcal{Z}_i = \mathcal{F}_i^1 \circ \mathcal{F}_i^2 \circ \cdots \circ \mathcal{F}_i^{K_i}, \tag{7}$$

where K_i is the number of fragments in episode i, $\circ$ denotes temporal concatenation, and each $\mathcal{F}_i^k$ contains one or more latent atoms from a local stage of the interaction. The partition follows the ordered interaction structure available during memory construction.

Each fragment is stored as

$$m_i^k = (\mathcal{F}_i^k, \mathbf{d}_i^k, \bar{\mathbf{s}}_i^k, [p_{i,k}^{\text{start}}, p_{i,k}^{\text{end}}], \mathbf{c}_i^k, r_i^k), \tag{8}$$

where $\mathbf{d}_i^k$ and $\bar{\mathbf{s}}_i^k$ are the observable descriptor and normalized state at the fragment anchor, $p_{i,k}$ records relative progress, $\mathbf{c}_i^k$ summarizes available transition information, and r_i^k records source provenance and order. Thus, the memory bank contains latent payloads and the minimum metadata required to retrieve and recompose them.

3.4 RECOMPOSE-AND-REFINE REASONING EXPERT

The Recompose-and-Refine Reasoning Expert maps the current query and memory bank to a fixed-size latent reasoning state. Its three stages correspond to fragment alignment, route tokenization, and context refinement.

Fragment alignment. We first restrict retrieval to the task or instruction namespace associated with l and exclude the currently evaluated episode. For each eligible fragment, the retrieval score combines visual semantic similarity with normalized state compatibility:

$$S(\mathbf{q}_t, m_i^k) = \lambda_v \cos(\mathbf{d}_t, \mathbf{d}_i^k) - \frac{\lambda_s}{d_s} \left\| \frac{\bar{\mathbf{s}}_t - \bar{\mathbf{s}}_i^k}{\boldsymbol{\sigma}_s} \right\|_2^2, \tag{9}$$

where d_s is the state-vector dimension, $\boldsymbol{\sigma}_s$ is estimated from the indexed states, and $\lambda_v, \lambda_s \geq 0$. A high-scoring fragment identifies an anchor inside a successful source trajectory. Rather than retrieving only that point, we retain the ordered local suffix beginning at the anchor. This preserves the transition structure that follows a state compatible with the current scene.

The retained suffixes from several source episodes form the candidate set. We compose them with a deterministic constrained search that preserves within-source order, rejects repeated fragments, and requires consecutive fragments to be compatible in observable state, progress, and any available transition metadata. The selected route is

$$\mathcal{R}_t = (m_{i_1}^{k_1}, m_{i_2}^{k_2}, \ldots, m_{i_J}^{k_J}), \tag{10}$$

where J is the number of selected fragments and adjacent elements may originate from different successful episodes. Only latent reasoning fragments, rather than historical action segments, participate in route recomposition.

Route tokenization. The route $\mathcal{R}_t$ has a variable number of fragments and latent atoms. For each atom, we add embeddings of its robot state, progress interval, source identity, route position, and transition metadata to the latent payload. Let $\widetilde{\mathcal{R}}_t$ denote the resulting ordered token sequence. A bank of L learned queries attends to this sequence and produces a fixed-size representation:

$$\mathbf{Z}_t^0 = C_\theta(\widetilde{\mathcal{R}}_t) \in \mathbb{R}^{L \times d}. \tag{11}$$

where C_θ denotes the learned-query route-tokenization module. This operation retains source and temporal organization while converting routes of different lengths into the latent interface expected by the policy. When no compatible route is available, the policy falls back to the backbone's native latent-reasoning path.

Context refinement. Retrieved reasoning supplies a long-horizon computational prior, whose query-local route tokens are refined in light of current embodied evidence. We formulate this evidence-driven route-token refinement through a fixed-depth residual operator over the route tokens:

$$\mathbf{Z}_t' = \mathbf{Z}_t^0 + \Delta_\theta(\mathbf{Z}_t^0, \mathbf{H}_t, \bar{\mathbf{s}}_t). \tag{12}$$

Here, Δ_θ is a fixed-depth residual operator that derives query features from the retrieved route and state, reads the current context as key-value evidence, and produces a residual correction to the query-local route tokens.

3.5 Memory-Adaptive Action Expert

The refined tokens $\mathbf{Z}_t'$ are mapped into the VLA backbone's native latent-conditioning interface and combined with the current context. We denote this interface operation by

$$\widetilde{\mathbf{H}}_t = I_\theta(\mathbf{H}_t, \mathbf{Z}_t'). \tag{13}$$

where I_θ denotes the native latent-conditioning interface. The action expert then predicts a conditional velocity field from $\mathbf{a}^\tau$ at flow time τ, where $\mathbf{a}^\tau = \tau \mathbf{a}^0 + (1-\tau)\mathbf{a}^1$,

$$\mathbf{v}_\theta = A_\theta(\mathbf{a}^\tau, \tau, \widetilde{\mathbf{H}}_t), \tag{14}$$

which is integrated to obtain the action chunk $\hat{\mathbf{a}}_{t:t+H-1}$. The expert consumes the current embodied context and the refined reusable reasoning state.

3.6 Learning Objectives

We train the memory pathway to reconstruct the frozen native latent reasoning target associated with the current embodied context. Let $\mathbf{Z}_t^*$ denote the target reasoning tokens and $\mathbf{Z}'_t$ the memory-conditioned prediction. Each reasoning token is a d-dimensional continuous vector in the VLA's latent reasoning space: $\mathbf{Z}_t^*$ is the frozen reference sequence and $\mathbf{Z}'_t$ is its memory-conditioned prediction. The latent alignment objective is

$$\mathcal{L}_{\text{latent}} = \frac{1}{L} \sum_{j=1}^{L} \left(1 - \cos(\mathbf{Z}'_{t,j}, \text{sg}(\mathbf{Z}^*_{t,j}))\right), \tag{15}$$

where j indexes the L reasoning tokens and $\text{sg}(\cdot)$ stops gradients through the target. The original action-learning objective of the VLA is retained; for flow-based action experts it is the conditional flow-matching loss

$$\mathcal{L}_{\text{act}} = \mathbb{E}_{\tau,\mathbf{a}^0,\mathbf{a}^1} \left\| \mathbf{v}_\theta(\mathbf{a}^\tau, \tau, \widetilde{\mathbf{H}}_t) - (\mathbf{a}^0 - \mathbf{a}^1) \right\|_2^2. \tag{16}$$

where $\mathbf{a}^0$ is sampled noise and $\mathbf{a}^1$ is the target action chunk. The complete objective combines latent alignment with the original action-learning objective:

$$\mathcal{L} = \mathcal{L}_{\text{act}} + \mathcal{L}_{\text{latent}}. \tag{17}$$

This objective directly supervises reusable latent reasoning and executable actions.

4 Experiments

4.1 Experimental Setup

Benchmarks and protocol. We evaluate FlowMem on RoboMME (Dai et al., 2026) and LIBERO-Plus (Fei et al., 2026). RoboMME evaluates temporal, spatial, object, and procedural memory through Counting, Permanence, Reference, and Imitation. We follow its full-16 protocol, evaluating 50 fixed episodes for each of the 16 tasks and reporting the task-macro success rate. Training, memory-source, and evaluation episodes are disjoint.

For LIBERO-Plus, every policy is trained only on the corresponding standard LIBERO suite (Liu et al., 2023) and is then evaluated zero-shot under the seven perturbation domains: camera viewpoint, robot initialization, language, lighting, background, observation noise, and layout. The main result reports success separately for the four original LIBERO suites, while the appendix breaks each method down by perturbation domain. The perturbed instances are used exclusively for zero-shot evaluation.

Baselines. On RoboMME, we consider no-memory policies, past-action conditioning, episodic memory, and the symbolic, perceptual, and recurrent MME-VLA variants (Fang et al., 2025; Sridhar et al., 2026; Dai et al., 2026). Together, these baselines contrast memory derived from action history, symbolic experience, visual context, and recurrent state. On LIBERO-Plus, we compare with suite-specific OpenVLA, UniVLA, OpenVLA-OFT (Kim et al., 2025), MemoryVLA (Shi et al., 2026b), MergeVLA (Fu et al., 2026), and VLA-Adapter (Wang et al., 2026).

Our controlled no-memory baseline, LaST_0 (Liu et al., 2026), reasons only from the current embodied input and neither constructs nor queries reasoning memory. All controlled models use the same benchmark-specific split and evaluation manifest.

4.2 Main Results

Table 1 evaluates whether reusable latent reasoning improves closed-loop performance on memory-dependent manipulation. Table 2 tests whether the same mechanism remains useful under zero-shot perturbations. Detailed RoboMME task results are deferred to Appendix A.

Table 1: **Main results on RoboMME.** Success rate (%) under the official Full-16 protocol. Higher is better. Best and second-best are bolded and underlined.

Memory	Method	Counting	Permanence	Reference	Imitation	Avg.
None	$\pi_{0.5}$	28.8	17.0	17.2	8.8	17.9
Action history	$\pi_{0.5}$ + past actions	29.1	22.8	15.9	11.2	19.7
Symbolic	SimpleSG + QwenVL	44.6	19.6	25.2	26.6	29.0
	GroundSG + QwenVL	38.0	<u>39.3</u>	31.6	21.9	32.7
Episodic	MemER	48.8	**53.2**	**38.0**	29.5	42.4
Perceptual	SAM2Act+	35.3	26.0	16.8	7.3	21.4
	TokenDrop + Context	57.9	26.9	23.7	29.4	34.5
	TokenDrop + Modul	52.3	26.8	34.7	38.3	38.0
	TokenDrop + Expert	59.4	23.6	25.3	31.1	34.9
	FrameSamp + Context	50.1	23.3	21.0	28.3	30.7
	FrameSamp + Modul	65.2	25.1	36.3	**51.4**	44.5
	FrameSamp + Expert	66.8	25.2	24.1	28.9	36.3
Recurrent	TTT + Modul	34.5	22.2	20.4	10.7	22.0
	RMT + Modul	34.1	15.8	21.1	9.7	20.2
None	LaST_0	<u>71.5</u>	28.0	36.5	49.0	<u>46.3</u>
Reasoning flow	**FLOWMEM (Ours)**	**75.0**	29.0	<u>37.5</u>	<u>50.5</u>	**48.0**

Table 2: **Main results on LIBERO-Plus.** Success rate (%) following suite-specific standard-LIBERO training and zero-shot LIBERO-Plus evaluation. **Avg.** pools all 10,030 instances across the four suites. Higher is better. Best and second-best are bolded and underlined.

Method	Venue	Spatial	Object	Goal	Long	Avg.
OpenVLA	CoRL'24	19.4	14.0	15.1	14.3	15.6
UniVLA	RSS'25	55.5	36.7	40.7	39.9	42.9
OpenVLA-OFT	RSS'25	<u>84.0</u>	66.5	<u>63.0</u>	66.4	69.6
MemoryVLA	ICLR'26	64.7	52.4	50.7	52.8	55.0
MergeVLA	CVPR'26	83.7	80.4	**65.6**	59.0	72.0
VLA-Adapter	AAAI'26	44.7	40.8	43.6	47.9	44.2
LaST_0	ICML'26	72.8	<u>83.7</u>	62.6	<u>73.8</u>	<u>73.2</u>
FLOWMEM (Ours)	–	**84.1**	**84.0**	62.9	**78.7**	**77.3**

Main findings. Tables 1 and 2 show that FLOWMEM achieves the highest aggregate result among the compared methods on both benchmarks. On RoboMME, it improves the Full-16 average from 46.3% to 48.0% over the controlled no-memory policy, with gains in Counting (75.0% vs. 71.5%), Reference (37.5% vs. 36.5%), Imitation (50.5% vs. 49.0%), and Permanence (29.0% vs. 28.0%). On LIBERO-Plus, FLOWMEM raises the pooled success rate from 73.2% to 77.3%. The largest gains occur under spatial and long-horizon perturbations, improving Spatial from 72.8% to 84.1% and Long from 73.8% to 78.7%, while retaining gains on Object and Goal. Together with the controlled analyses below, these results are consistent with reusable reasoning flows supplying transferable prior computation and current-context refinement supporting their use under closed-loop task variation and zero-shot distribution shifts.

4.3 EVIDENCE FOR LATENT-REASONING REUSE

Compositional reuse. On the full RoboMME protocol, we use the same memory bank as in Table 1 and vary how latent reasoning is retrieved and recomposed. The target episodes, policy interface, and action budget remain fixed. Table 3 contrasts nearest-state and single-source retrieval with ordered multi-source reuse. Nearest-state retrieval reaches 45.6%, below the 46.3% no-memory policy, showing that local latent similarity alone is insufficient. Restricting retrieval to one source reduces valid-route coverage from 100.0% to 89.1% and success to 44.0%. In contrast, FLOWMEM composes compatible fragments across sources for 70.5% of policy queries and reaches 48.0%. Thus, multi-source recomposition expands the available compatible reasoning evidence and is associated

with the strongest closed-loop performance. The same direction holds on LIBERO-Plus: Table 5 improves from 76.5% with a single source to 77.3% with the full model.

Table 3: **Compositional reuse on RoboMME.** Memory-conditioned methods use the same bank. Higher SR is better. Best and second-best are bolded and underlined.

Method	Valid	Multi	SR
LaST_0	–	–	<u>46.3</u>
Nearest state	–	–	45.6
Single source	89.1	0.0	44.0
w/o refinement	100.0	69.4	<u>46.3</u>
FLOWMEM (Ours)	100.0	70.5	**48.0**

Table 4: **Memory counterfactuals on RoboMME.** ✓: retained property; blank: corrupted. Higher SR is better. Best and second-best are bolded and underlined.

Method	Task	Order	Prog.	SR
LaST_0	–	–	–	<u>46.3</u>
Wrong task		✓		46.1
Out of order	✓			44.4
Wrong progress	✓	✓		46.1
Correct memory	✓	✓	✓	**48.0**

Memory counterfactuals. On the same RoboMME Full-16 manifest and memory bank as Table 3, the memory-conditioned variants share a checkpoint and embodied input; only the preserved properties of the retrieved memory change. The memory-free LaST_0 row uses its own checkpoint. Table 4 tests whether performance depends on task identity, temporal order, and progress alignment rather than the presence of arbitrary extra context. Correct memory attains 48.0%, exceeding wrong-task and wrong-progress memory by 1.9 points and out-of-order memory by 3.6 points. The largest drop follows disruption of temporal order, while the remaining counterfactuals are consistent with task identity and progress alignment contributing to reliable reasoning reuse. Under this protocol, the pattern is inconsistent with the explanation that arbitrary additional latent context is equally useful.

Core ablations. Table 5 isolates the three proposed ingredients: representing experience as an ordered flow, recomposing fragments across episodes, and refining the resulting route with the current observation. All variants retain the same action interface and latent-token budget. On RoboMME, the memory-conditioned variants use the same checkpoint. On LIBERO-Plus, the single-source variant is trained separately from the same LaST0 base with the corresponding standard-LIBERO suite and matched training budget.

Table 5: **Core ablation of FlowMem.** Variants share the same action interface and latent-token budget. Higher is better. Best and second-best are bolded and underlined.

Method	Flow	Multi-src.	Refine	RoboMME	LIBERO-Plus
LaST_0				<u>46.3</u>	73.2
Single-source FlowMem	✓		✓	44.0	76.5
FlowMem w/o refinement	✓	✓		<u>46.3</u>	<u>76.7</u>
FLOWMEM (Ours)	✓	✓	✓	**48.0**	**77.3**

The ablations are consistent with benefits from the full design on both benchmarks. On RoboMME, removing multi-source recomposition reduces success from 48.0% to 44.0%, while removing current-context refinement reduces it to 46.3%. The same ordering appears on the complete 10,030-instance LIBERO-Plus evaluation: the single-source and no-refinement variants obtain 76.5% and 76.7%, respectively, below the full model's 77.3% but above the 73.2% no-memory baseline. Thus, on these benchmark protocols, multi-source recomposition is associated with gains beyond RoboMME, and current-observation refinement with a further, consistent improvement when the retrieved route is deployed under closed-loop variation.

5 Related Work

Recent vision-language-action models have introduced intermediate reasoning between multimodal perception and action generation. ECoT generates embodied reasoning about plans, subgoals, motions, and visually grounded entities, whereas CoT-VLA predicts future images as visual goals before producing actions (Zawalski et al., 2024; Zhao et al., 2025). ThinkAct compresses action-aligned planning into a visual latent representation, and Fast-ThinkAct further distills lengthy reasoning

traces into a small number of continuous latent tokens for efficient control (Huang et al., 2025; 2026). More recent approaches internalize reasoning directly in continuous representation spaces. LaRA-VLA combines textual reasoning and future-oriented visual prediction in latent reasoning slots, while LaST_0 learns a spatio-temporal latent chain-of-thought supervised by future visual dynamics, 3D structure, and robot proprioception (Bai et al., 2026; Liu et al., 2026). RD-VLA instead performs weight-tied recurrent refinement to adapt the amount of latent computation at test time (Tur et al., 2026). Complementary geometric analysis models reasoning as a context-cumulative flow in representation space and shows that its velocity, curvature, and ordering capture structure beyond individual embedding positions (Zhou et al., 2026). This perspective motivates characterizing reasoning through its ordered evolution rather than as exchangeable latent vectors. Existing reasoning-enhanced VLAs nevertheless produce or refine such intermediate computation only for the current input: successful latent reasoning trajectories are not retained as experience that can initialize and guide future reasoning.

A separate line of work equips robot policies with memory to address history-dependent control. Notes-to-Self maintains a structured textual scratchpad that records grounding, plans, and completed actions, while MemoryVLA retrieves low-level perceptual details and high-level cognitive semantics from a temporal memory bank. MemoryVLA++ further combines temporal modeling with memory and imagination (Haresh et al., 2026; Shi et al., 2026b;a). Recurrent approaches such as μVLA, ReMem-VLA, and TFP instead propagate compact memory tokens, recurrent queries, or latent belief states across environment steps (Cherepanov et al., 2026; Li et al., 2026a; Liang et al., 2026), whereas MemER retrieves bounded visual experience for temporally dependent control (Sridhar et al., 2026). RoboMME systematizes these requirements into temporal, spatial, object, and procedural memory and evaluates multiple memory representations and integration strategies under a shared $\pi_{0.5}$ backbone (Dai et al., 2026). These methods demonstrate that historical context is essential for non-Markovian manipulation. Their memories, however, represent textual task state, perceptual-cognitive evidence, retrieved observations, or recurrent policy state rather than the latent reasoning process that connects embodied observations to action generation.

Several recent approaches lie near the intersection of reasoning and memory, but expose different interfaces. TRM-VLA retrieves explicit reasoning history at key decision points to maintain temporal consistency within an episode (Li et al., 2026b). LaMem-VLA retrieves visual and action history, condenses it into latent memory tokens, and injects them as policy context (Qu et al., 2026). OptimusVLA retrieves trajectory-level action priors and combines them with local execution history to improve action generation (Li et al., 2026c), while WeaveLA transfers a compressed latent state across event-defined subtask boundaries (Zhu et al., 2026). Beyond robotic control, MemGen generates latent memory tokens from experience consolidated in auxiliary parameters and uses them to alter subsequent reasoning in language agents (Zhang et al., 2026). Thus, prior work has separately explored explicit reasoning history, latent policy memory, trajectory-conditioned action generation, event-triggered latent state transfer, and memory-conditioned language reasoning. We study their intersection in embodied VLAs: successful latent reasoning trajectories are preserved as ordered, source-grounded flows; compatible local flows are retrieved from multiple episodes and recomposed into a task-conditioned reasoning route; and the resulting route is iteratively refined against the current observation before being decoded by the original action expert.

6 Conclusion

We presented FlowMem, a VLA framework that turns successful latent computation into reusable reasoning-flow memory. FlowMem stores ordered latent fragments with progress and transition metadata, retrieves anchor-aligned local suffixes, recomposes compatible multi-source routes, and refines the resulting tokens with the current embodied context before action generation. Across RoboMME and LIBERO-Plus, FlowMem improves the reported main-task success over the memory-free LaST_0 baseline. Controlled retrieval and memory-counterfactual experiments are consistent with source compatibility, temporal order, and progress alignment contributing to performance. These results support reusable latent reasoning flows as a practical mechanism for connecting successful prior computation to closed-loop VLA control.

AI USE STATEMENT

We used generative AI tools to assist with experiment and methodology planning, implementation and debugging, analysis and interpretation of experimental results, and drafting, editing, and formatting parts of the manuscript and figures. Generative AI was not used to create synthetic datasets or to establish theoretical or mathematical claims. All experiments were executed using the reported code and evaluation protocols, and the authors manually reviewed AI-assisted text, code, figures, and analyses against the underlying implementations, logs, and cited sources. We take responsibility for the final content of this work, including all claims and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Sections 3 and 4 specify the model formulation, training objective, benchmark protocols, controlled baselines, and ablation settings. Table captions state the metrics and aggregation rules, and Appendix A provides per-task RoboMME and perturbation-wise LIBERO-Plus results. All comparisons use the stated protocols and fixed test episodes, with success rates aggregated from episode-level records. An anonymous code repository is available at `https://anonymous.4open.science/r/FlowMem-C963/README.md`.

REFERENCES

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

Egor Cherepanov, Nikita Kachaev, Daniil Zelezetsky, Aydar Bulatov, Artem Pshenitsyn, Yuri Kuratov, Alexey Skrynnik, Aleksandr I. Panov, and Alexey K. Kovalev. μVLA: On recurrent memory for partially observable manipulation in VLA models. *arXiv preprint arXiv:2606.12497*, 2026.

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*, 2026.

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.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 38574–38583, 2026.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Sanjay Haresh, Daniel Dijkman, Apratim Bhattacharyya, and Roland Memisevic. Notes-to-self: Scratchpad augmented VLAs for memory dependent manipulation tasks. In *IEEE International Conference on Robotics and Automation*, 2026.

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

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

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Robotics: Science and Systems*, 2025.

Hang Li, Fengyi Shen, Dong Chen, Liudi Yang, Xudong Wang, Jinkui Shi, Zhenshan Bing, Ziyuan Liu, and Alois Knoll. ReMem-VLA: Empowering vision-language-action model with memory via dual-level recurrent queries. *arXiv preprint arXiv:2603.12942*, 2026a.

Xiang Li, Ya-Li Li, Yuan Wang, and Shengjin Wang. TRM-VLA: Temporal-aware chain-of-thought reasoning and memorization for vision-language-action models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026b.

Zaijing Li, Bing Hu, Rui Shao, Gongwei Chen, Dongmei Jiang, Pengwei Xie, Jianye Hao, and Liqiang Nie. Global prior meets local consistency: Dual-memory augmented vision-language-action model for efficient robotic manipulation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35135–35145, 2026c.

Yushen Liang, Yue Peng, Baosheng Jin, Tianluo Zhang, Xinyu Zhang, Shuyi Zhou, Zhuoran Chen, Xinqi Liu, and Shenji Wan. TFP: Temporally conditioned memory-fusion policies for visuomotor learning. *arXiv preprint arXiv:2607.08283*, 2026.

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*, volume 36, 2023.

Zhuoyang Liu, Jiaming Liu, Hao Chen, Ziyu Guo, Chengkai Hou, Chenyang Gu, Jiale Yu, Xiangju Mi, Renrui Zhang, Zhengping Che, et al. LaST_0: Latent spatio-temporal chain-of-thought for robotic vision-language-action model. In *International Conference on Machine Learning*, 2026.

Hongyu Qu, Jianzhe Gao, Xiaobin Hu, Shaohuan Yang, Xinlei Yu, Rui Yan, Wenguan Wang, Xiangbo Shu, and Shuicheng Yan. Dual latent memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2607.07608*, 2026.

Hao Shi, Weiye Li, Bin Xie, Yulin Wang, Renping Zhou, Tiancai Wang, Xiangyu Zhang, Ping Luo, and Gao Huang. MemoryVLA++: Temporal modeling via memory and imagination in vision-language-action models. *arXiv preprint arXiv:2606.09827*, 2026a.

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*, 2026b.

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

Yalcin Tur, Jalal Naghiyev, Haoquan Fang, Wei-Chuan Tsai, Jiafei Duan, Dieter Fox, and Ranjay Krishna. Recurrent-depth VLA: Implicit test-time compute scaling of vision-language-action models via latent iterative reasoning. *arXiv preprint arXiv:2602.07845*, 2026.

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. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18638–18646, 2026.

Michal Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. MemGen: Weaving generative latent memory for self-evolving agents. In *International Conference on Learning Representations*, 2026.

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

Yufa Zhou, Yixiao Wang, Xunjian Yin, Shuyan Zhou, and Anru R. Zhang. The geometry of reasoning: Flowing logics in representation space. In *International Conference on Learning Representations*, 2026.

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. *arXiv preprint arXiv:2606.17463*, 2026.

A Additional Experimental Results

A.1 RoboMME Per-Task Results

Table 6: **Per-task RoboMME results.** Success rate (%) under the official 16-task, 50-episode-per-task protocol. Higher is better. Best results are bolded.

Method	BinFill	PickX	SwingX	StopCube	V-Unmask	V-Unmask-S	B-Unmask	B-Unmask-S
FrameSamp+Modul	39.6	87.3	92.0	42.0	32.7	24.4	25.1	18.2
$LaST_0$	46.0	96.0	90.0	54.0	34.0	24.0	**26.0**	**28.0**
FlowMem (Ours)	**50.0**	**98.0**	**96.0**	**56.0**	**40.0**	**26.0**	24.0	26.0

Method	PickHigh	V-Repick	V-PlaceB	V-PlaceO	MoveCube	InsertPeg	PatternLock	RouteStick	Avg.
FrameSamp+Modul	22.9	**30.4**	**60.0**	32.0	77.8	**7.6**	53.6	**66.7**	44.5
$LaST_0$	22.0	22.0	**60.0**	**42.0**	80.0	2.0	52.0	62.0	46.3
FlowMem (Ours)	**24.0**	26.0	**60.0**	40.0	**82.0**	2.0	**56.0**	62.0	**48.0**

Per-task analysis. The 1.7-point Full-16 gain is distributed across 10 of the 16 tasks (34 positive points versus 6 negative points), rather than being produced by one outlier. The largest gains occur on Visual Unmask (40.0% vs. 34.0%), Visual Repick (26.0% vs. 22.0%), and Pattern Lock (56.0% vs. 52.0%); the two B-Unmask variants and Visual PlaceO are lower, and three tasks are unchanged. Together with Table 3, this distribution supports the central claim that ordered latent reasoning can be reused across distinct closed-loop tasks, while showing that useful reuse still requires compatibility with the current embodied context.

A.2 Detailed LIBERO-Plus Results

Table 7: **Perturbation-wise LIBERO-Plus results.** Success rate (%) following suite-specific standard-LIBERO training and zero-shot LIBERO-Plus evaluation. Each perturbation column pools its instances across the four suites; **Total** pools all 10,030 instances. Higher is better. Best and second-best are bolded and underlined.

Method	Venue	Camera	Robot	Language	Light	Background	Noise	Layout	Total
OpenVLA	CoRL'24	0.8	3.5	23.0	8.1	34.8	15.2	28.5	15.6
UniVLA	RSS'25	1.8	<u>46.2</u>	69.6	69.0	81.0	21.2	31.9	42.9
OpenVLA-OFT	RSS'25	56.4	31.9	79.5	88.7	**93.3**	75.8	<u>74.2</u>	69.6
MemoryVLA	ICLR'26	10.3	**50.2**	79.5	65.3	89.4	31.1	**75.1**	55.0
MergeVLA	CVPR'26	61.7	44.8	75.7	**92.0**	<u>93.0</u>	73.7	**75.1**	72.0
VLA-Adapter	AAAI'26	6.1	29.1	66.2	56.5	70.5	25.7	69.2	44.2
$LaST_0$	ICML'26	<u>80.2</u>	36.3	<u>81.8</u>	86.5	88.8	<u>77.5</u>	68.8	<u>73.2</u>
FlowMem	–	**88.4**	36.6	**82.2**	<u>90.2</u>	90.5	**88.3**	71.2	**77.3**

Perturbation-wise analysis. The largest gains occur under Camera (80.2% to 88.4%) and Noise (77.5% to 88.3%), with a further 3.7-point gain under Light. This pattern supports the central claim: recomposition is useful when stored trajectories provide a compatible route, while refinement aligns that route with the current observation rather than treating memory as fixed retrieved context. Gains on Robot and Language are only 0.3 and 0.4 points, respectively, and FLOWMEM does not lead every perturbation column, delimiting where the mechanism offers only marginal benefit.

A.3 INFERENCE COST OF LATENT-REASONING REUSE

Table 8 reports the mean online control time per episode under a fixed profiling protocol. We profile 256 episodes per method on an NVIDIA H800 GPU using the same evaluation configuration. For each episode, we sum the time from an observation entering the policy to its action chunk being returned across all control decisions, then average over the profiling episodes.

Table 8: **Inference efficiency on RoboMME. Time / episode** is the mean total online control time from observation to action-chunk return over the profiling episodes. Lower time and higher SR are better.

Method	Time / episode (s)	SR (%)
LaST_0	19.6	46.3
FLOWMEM	15.5	**48.0**

On RoboMME, FLOWMEM reduces the mean online control time from 19.6 to 15.5 seconds per episode while improving success from 46.3% to 48.0%. Under this profiling protocol, reasoning-flow reuse therefore improves task performance without increasing online control time, supporting its practical use in closed-loop control.