LONGCONTEXTVLA: SCALING ROBOT MEMORY TO HUNDREDS OF IMAGES

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models typically condition on only the current observation or a small set of past keyframes. This limits their ability to solve long-horizon tasks that require persistent visual memory. We develop a VLA that conditions on more than 200 images of visual history while maintaining real-time inference speed. Our approach builds on Qwen3.5-4B, a multimodal model with a 262K-token context window, which we fine-tune for robotic control. We find that long visual context alone is insufficient for robust generalization on memory-intensive tasks, so we train the VLA to reason over and narrate events in its visual history, using parallel decoding to avoid the latency cost of autoregressive reasoning. By caching past visual tokens and processing only newly arriving images, our streaming implementation achieves per-step latency close to that of a single-image VLA. We evaluate our approach on a long-horizon shell game and object retrieval task, achieving average 83% success in unseen scenarios, as well as competitive performance on the RoboMME benchmark. Finally, we demonstrate memory-based prompting, where the robot observes a human demonstration and reproduces the demonstrated behavior, and can generalize to new scenarios. Together, our results show that scalable visual memory, learned reasoning, and streaming inference enable real-time, long-horizon robot control. Our paper website is: Project website.

1 INTRODUCTION

To realize the vision of general-purpose embodied intelligence, robots must seamlessly incorporate dense memory of the past when making decisions and manipulating objects in their surroundings. Consider a robot tasked with putting away tools and materials that were retrieved for a project an hour prior, or a robot that needs to disinfect all of the surfaces in an environment. Dense, long-term memory of past video observations would enable robots to complete tasks like these. However, training a generalist long-context policy presents multiple major challenges that has lead many prior robotic foundation models to avoid memory altogether (Kim et al., 2024; Black et al., 2025; Bjorck et al., 2025). Processing many past frames introduces significant computation burden, while robot policies must operate in real-time to generate fast, smooth motions. Moreover, such memory introduces an influx of information that significantly complicates the learning problem. In this work, we study how to overcome these challenges to scale the context of generalist robot policies to hundreds of images.

Recent works have aimed to develop long-term memory in generalist policies by integrating a small number of frames as context or learning to use compressed, e.g. text-based, representations of the past, or both. Research on the former has used a handful of strided frames (Ghosh et al., 2024; Torne et al., 2026) or carefully selected keyframes (Sridhar et al., 2026; Mark et al., 2026). These methods sidestep the computational and learning challenges by using fewer frames of context, but either offer only dense, short-term memory from a few seconds or sparse, long-term memory that can miss important context. Compressed representations of the past, e.g. through textual summaries, offer an avenue for storing task-relevant information over long time-horizons (Torne et al., 2026; Dai et al., 2026; Shi et al., 2026; Jang et al., 2025), but also can miss important context that might become relevant later and are not directly applicable to tasks that require detailed spatial or visual long-term memory. By studying dense, long-context visual memory, we aim to address these shortcomings and develop an approach that is applicable to tasks with a wide variety of memory demands.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

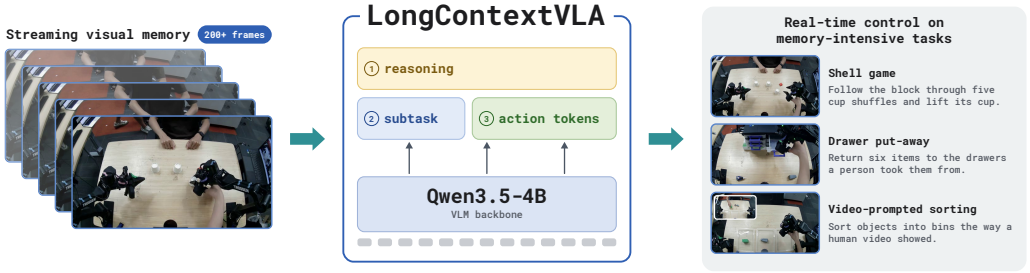


Figure 1: **LongContextVLA overview.** We present LongContextVLA, a vision-language-action model that scales robot memory to hundreds of images while running in real time. By narrating and reasoning over its streaming visual history before acting, LongContextVLA solves three memory-intensive real-robot tasks that require tracking a hidden object, remembering where items came from, and imitating a human video prompt.

To this end, we introduce LongContextVLA, a vision-language-action (VLA) model that can ingest hundreds of visual frames for real-time decision making in memory-intensive tasks. Our model is based on Qwen3.5 (Qwen Team, 2026), a vision-language model (VLM) that natively supports context lengths of 262K tokens (extendable up to 1M tokens) and is pre-trained on Internet-scale data that includes video question answering problems over long sequences of frames. To unlock real-time use, we introduce two key design choices. First, since interaction at test-time provides streaming video, we cache computation from past images and only run encoding over newly received observations, substantially reducing inference costs at each timestep by amortizing across time. Second, we use tokenized action chunks with parallel decoding of multiple blocks of these tokens to further reduce inference costs under the real-time threshold. Beyond inference speed, we must be able to extract relevant information from video, and imitation learning to predict future actions is a relatively weak learning signal for end-to-end learning from dense video. We therefore train the VLA to narrate its own visual past with past timestamps before predicting what to do next in text and predicting actions. This objective teaches the VLA to explicitly understand its past before deciding what to do next. This reasoning is decoded in parallel to maintain fast inference.

We evaluate LongContextVLA on a large suite of simulated robot manipulation tasks that demand many forms of memory through the RoboMME benchmark (Dai et al., 2026) and on three challenging real robot manipulation tasks that involve human video prompting. On the simulated robot memory benchmark tasks, the results show that LongContextVLA shows far greater consistency across different tasks, achieving 84.9% average success, 40 points higher than the best prior method. We attribute this consistency to choosing to use dense visual memory over sparse or compressed memory representations.On the real robot, we study three manipulation tasks that stress both density and length of memory: the "shell game," where a block is hidden under one of three cups and the robot must lift the correct cup after five shuffles; a drawer put-away task, where the robot returns six items to the drawers a person removed them from; and video-prompted sorting, where the robot sorts objects into bins the way a short human video showed. Importantly, we measure performance with objects and cup shuffle orders that are not seen in the demonstration data. With LongContextVLA, the robot can complete each task with real-time inference and achieves 89% average success, 39 points higher than the strongest baseline and 59 points higher than $\pi_{0.7}$ (Physical Intelligence, 2026) We also find that narration and subtask prediction plays a critical role in extracting signal from the video context on these tasks, yielding 16 points higher success on average, and 20 points higher on unseen scenarios.

2 Related Work

Memory for Robot Policies. Several past works have explored how to extend the context of robot policies to solve partially observable tasks. Extending the context of a robot policy is non-trivial, as the policy must learn to act on observation frames from many past timestamps, and a longer history often leads the policy to learn spurious correlations. To address this, Torne et al. (2025) add an auxiliary loss that regularizes what the policy retains, Dashora et al. (2026) train visual

features by reconstructing salient image patches selected by a VLM, and Koo et al. (2026) initialize visual context tokens with time-contrastive training to promote temporally distinct features. Other works use VLMs directly at inference time, exploiting their strong video-understanding priors. For instance, Sridhar et al. (2026); Mark et al. (2026) use a VLM to select task-relevant keyframes and discard the rest, which can hurt performance on tasks that require dense video memory such as imitation or motion understanding (Dai et al., 2026), as we demonstrate in our experiments. Torne et al. (2026) use textual memory, which is limited to information that can be described in text and can fail on tasks that require spatial memory. Our method is closest to the concurrent SimpleMemVLA (Yin et al., 2026), which also exploits the dense video-understanding prior of Qwen3.5 (Qwen Team, 2026) and trains a VLA to consume the full video context, predict a subtask, and then predict the action. However, we show that extending the context and predicting the subtask is not enough on its own: predicting a narration of the past video provides additional supervision that makes better use of the video-understanding prior, as shown in our real-robot results.

Chain-of-Thought for Robotics. For robotics tasks that require high-level planning, such as the long-horizon memory tasks we evaluate, many works have shown that first predicting a grounded scene or plan description and then conditioning action prediction on this chain-of-thought improves task performance (Zawalski et al., 2024). Some works argue that this helps because it allocates more computation to action prediction, and show that latent reasoning helps as well (Bai et al., 2026). Others argue that chain-of-thought makes better use of the VLM's priors by turning a robotics problem into something closer to a visual reasoning problem (Chen et al., 2026). We believe both factors contribute to the benefit of predicting a narration of the video before the subtask and action. However, autoregressive decoding of the narration is expensive, so prior work has explored cheaper ways to incorporate reasoning, such as an auxiliary output (Chen et al., 2025) or caching reasoning across timesteps (Duan et al., 2025). Following Zhong et al. (2026), we decode the narration from a block of reserved tokens in parallel, avoiding the autoregressive generation otherwise needed during rollouts.

3 LongContextVLA

3.1 Problem Setup

We consider a policy π_θ that controls a robot from a long history of visual observations. At each decision step t, the policy receives a language instruction ℓ, the robot's proprioceptive state q_t, the current left and right wrist images I_t^{left} and I_t^{right}, and a history of top-down images $V_t = (I_0^{\text{top}}, I_s^{\text{top}}, I_{2s}^{\text{top}}, \ldots, I_t^{\text{top}})$ sampled every s steps from the start of the episode through the current step. The full input, in the order consumed by the model, is

$$x_t = \big(V_t,\ I_t^{\text{left}},\ I_t^{\text{right}},\ \ell,\ q_t\big). \tag{1}$$

Each demonstration step is annotated with three training targets. The first is an action chunk A_t, a sequence of discrete tokens produced by a tokenizer that encodes the demonstrated actions over the next H steps. The second is a subtask u_t, a short description of what the robot should do now, such as "pick up the block." The third is a narration r_t, a timestamped account of the task-relevant events in the history, such as "At 2 s, the robot picks up the block. At 4 s, the block is placed inside the box." We generate the ground-truth labels for these narrations automatically by simply concatenating the timestamped subtask labels corresponding to the memory context. The subtask and action chunk are all the robot needs in order to act. The narration is not needed to act, but producing it requires the model to explicitly understand and reason about its own past.

3.2 Reasoning Across Memory

We consider three variants of the same policy. They share the architecture and inputs, are all trained with cross-entropy loss on their targets, and differ only in what the policy must produce before its actions. For a target token sequence $y = (y^1, \ldots, y^N)$ and a context c, we write the autoregressive cross-entropy loss as

$$\mathcal{L}(y \mid c) = -\sum_{i=1}^{N} \log \pi_\theta\big(y^i \mid c,\ y^{<i}\big). \tag{2}$$

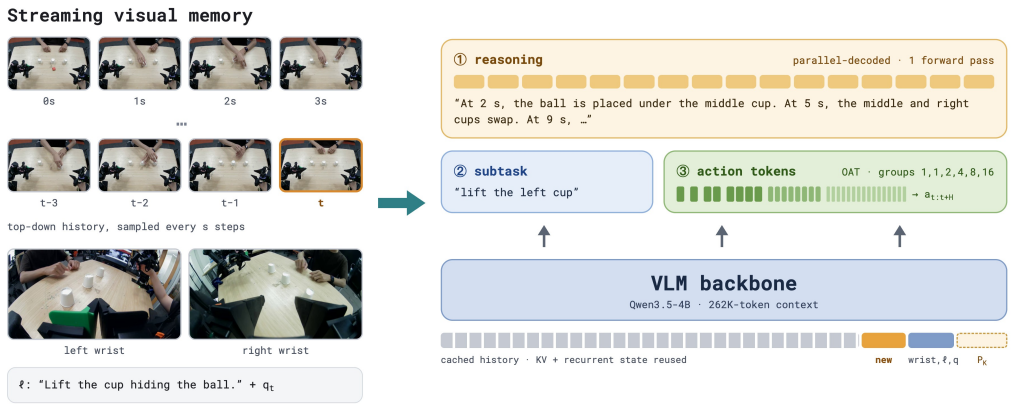


Figure 2: **Architecture of LongContextVLA.** The model takes as input a streaming history of top-down images sampled every s steps since the start of the episode, together with the current wrist images, task instruction, and robot joint states. The history tokens are cached and reused across steps, so only the newest image is encoded at each step. The model first narrates the timestamped events in its history, decoding all narration tokens in a single forward pass from a block of reserved tokens, and then generates the language subtask and action tokens, which are decoded in groups and converted to an action chunk by the OAT decoder.

Below, we write each target in the order the model produces it, along with the corresponding training loss.

1. **Subtask prediction:** $\pi_\theta : x_t \mapsto (u_t,\, A_t)$
   The policy generates the current subtask and then outputs the action chunk. This is a common training objective widely used for training VLAs. The training loss is
   $$\mathcal{L}_{\text{sub}} = \underbrace{\mathcal{L}(u_t \mid x_t)}_{\text{subtask}} + \underbrace{\mathcal{L}(A_t \mid x_t,\, u_t)}_{\text{action}}. \tag{3}$$

2. **Subtask prediction with memory narration:** $\pi_\theta : x_t \mapsto (r_t,\, u_t,\, A_t)$
   The policy first narrates its history, then generates the subtask, and then outputs the action chunk. Each output is produced with the preceding ones in context, so the decision is made with an explicit account of the past. The training loss is
   $$\mathcal{L}_{\text{narr}} = \underbrace{\mathcal{L}(r_t \mid x_t)}_{\text{narration}} + \underbrace{\mathcal{L}(u_t \mid x_t,\, r_t)}_{\text{subtask}} + \underbrace{\mathcal{L}(A_t \mid x_t,\, r_t,\, u_t)}_{\text{action}}. \tag{4}$$
   The narration is generated token by token at deployment, so latency grows with its length.

3. **Subtask prediction with parallel-decoded memory narration:** $\pi_\theta : (x_t,\, P_K) \mapsto (r_t,\, u_t,\, A_t)$
   This variant is similar to the second except in how the narration is decoded. Because decoding a long narration token by token is slow, we instead append a block P_K of K reserved tokens to the input so that the narration can be decoded in parallel in a single forward pass. The training loss is
   $$\mathcal{L}_{\text{par}} = \underbrace{-\sum_{k=1}^{K} \log \pi_\theta\big(r_t^k \mid x_t,\, P_K\big)}_{\text{parallel narration}} + \underbrace{\mathcal{L}(u_t \mid x_t,\, P_K)}_{\text{subtask}} + \underbrace{\mathcal{L}(A_t \mid x_t,\, P_K,\, u_t)}_{\text{action}}, \tag{5}$$
   where r_t^k is the k-th token of the narration, padded to length K. Unlike the narration loss in the second variant, each narration token is predicted from x_t and P_K without conditioning on the preceding narration tokens, which is what allows all K tokens to be decoded at once.

Together, these variants let us study how much supervision about the past a VLA needs to use a long visual memory effectively. Subtask prediction is simple and widely used in robot learning, and it already encourages the policy to attend to its history, since the correct subtask often depends on what has happened so far. However, as the history grows to thousands of tokens, the policy must identify which visual events among many thousands of tokens are relevant to the current decision, and a few-word subtask label may be too sparse a signal to learn this reliably. Narrating the past provides a denser signal. By recounting what has happened since the start of the episode, the policy forms an explicit account of its history that can better inform what it should do now. This is especially important when the task depends on subtle past events. However, generating the narration autoregressively is too slow for real-time control, since latency grows with the length of the narration. Parallel decoding aims to retain the benefits of narration while producing it in a single forward pass, at little additional inference cost.

3.3 Enabling Fast Model Inference

Long visual histories introduce two sources of inference latency: repeatedly processing past observations and sequentially generating reasoning and action tokens. We address both by reusing computation across control decisions and predicting groups of output tokens in parallel. Consecutive decisions share almost all of their historical images. We therefore place the top-down visual history first in the input and retain its visual features and model state between decisions. For Qwen3.5, this state includes the keys and values of full-attention layers, together with the convolution and recurrent states of GatedDeltaNet layers. Causal computation allows us to append new observations without recomputing the representations of earlier tokens.

Each decision starts by processing the newly received top-down image using the retained history cache. Immediately after processing this image, we save a copy of the resulting convolution and recurrent states and record the key–value cache length. These saved states include the new top-down history, but have not yet processed the current wrist images, instruction, or proprioception. We then process those decision inputs, followed by the reserved narration tokens when using parallel narration, and generate the subtask and action tokens. This computation adds entries to the key–value cache and modifies both the convolution and recurrent states.

After sending the predicted actions to the robot, we restore the exact convolution and recurrent states saved immediately after processing the new top-down image. We also remove all key–value entries added after this recorded length. This preserves the latest top-down history while removing the effects of the current decision's inputs and outputs. At the next decision, we use this restored cache to process the next new top-down image, save the resulting states, and repeat the process. Only the fixed-size convolution and recurrent states require a saved copy, avoiding a copy of the entire growing key–value cache. New tokens still attend to the retained history, so computation is reduced but remains dependent on history length.

We further reduce latency through parallel narration and grouped action prediction. As described above, the reserved narration positions are processed together in one forward pass. The policy uses their representations to predict the subtask and actions without feeding predicted narration words back into the model. For actions, we use ordered action tokenizer (OAT) (Liu et al., 2026) to encode each continuous action chunk as discrete tokens. Our real-robot policies predict 32 tokens in groups of size $1, 1, 2, 4, 8, 16$. Tokens within each group are predicted together, conditioned on the codes from earlier groups. This requires six sequential prediction calls, after which the OAT decoder reconstructs the action chunk in one pass.

4 Experiments

We evaluate our method and alternative approaches on memory-intensive tasks in both real-world and simulation settings. Our experiments address three key research questions:

1. Is subtask prediction sufficient for long-horizon memory tasks, or does narrating the past improve performance?
2. Can VLAs maintain real-time performance with context lengths of up to one million tokens?

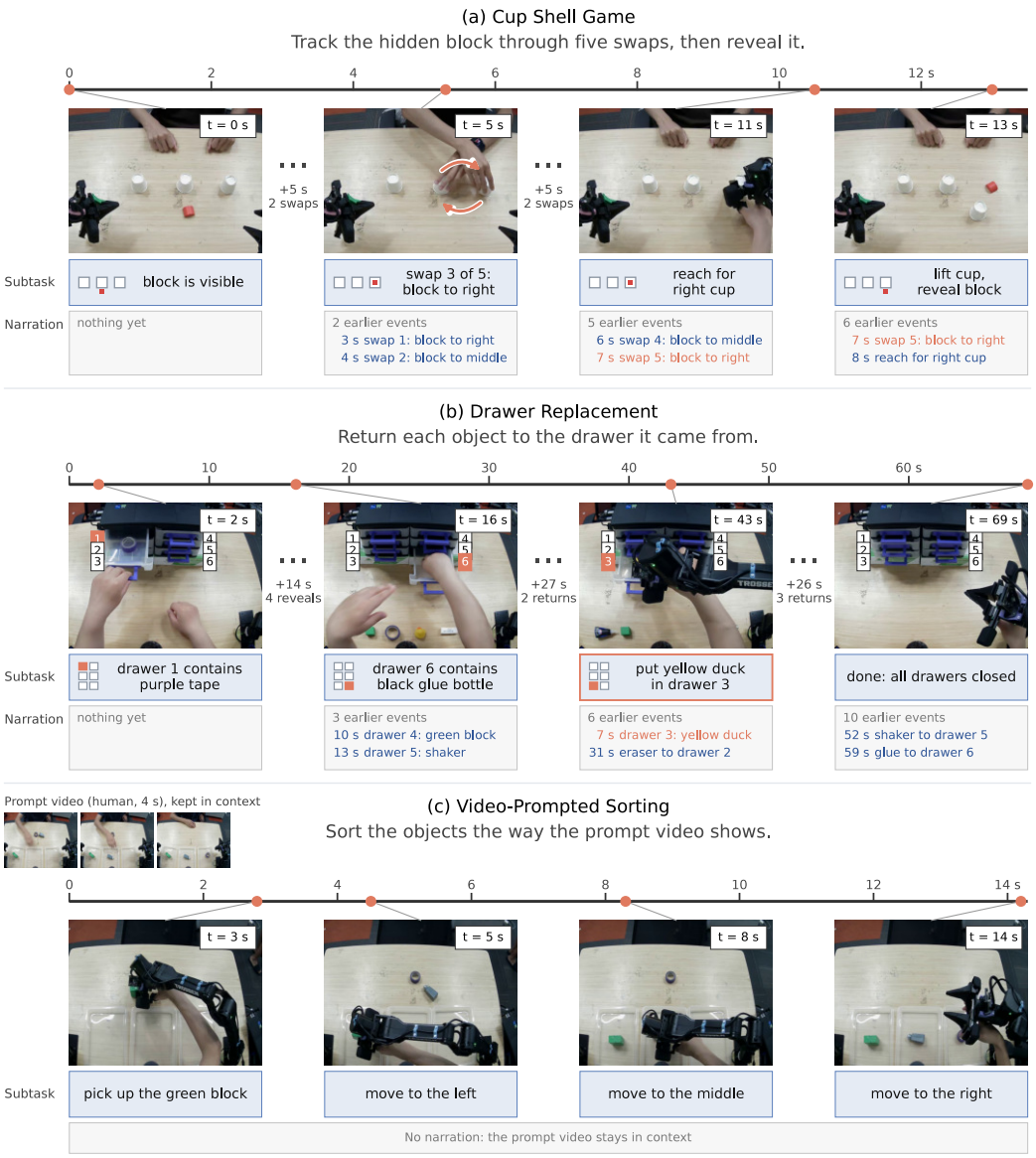


Figure 3: **Real-robot tasks.** Top-down frames along one episode of each task, with the subtask and narration the policy produces at that step: (a) shell game, (b) drawer put-away, and (c) video-prompted sorting, where the human prompt video stays in context and no narration is produced.

3. Can models trained to use memory contexts also use human demonstrations as visual prompts?

To address these questions, we design two real-world memory-intensive tasks and one visual prompting task. We also evaluate memory capabilities in simulation using the RoboMME benchmark (Dai et al., 2026).

Shell Game (Memory Task) The shell game involves three cups and a single red block. A human hides the block under the middle cup, then shuffles the cups five times. The robot must identify the cup containing the hidden block after the shuffling ends. This task spans approximately 15 seconds and requires the model to remember where the block was initially hidden and track the cup motions throughout the sequence.

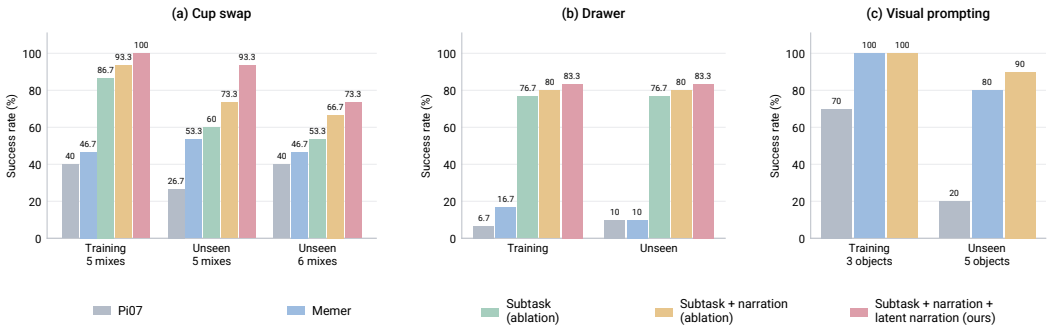


Figure 4: Success rates on cup swap, drawer, and visual prompting tasks. N=15 for each cup swap trials, N=5 for each drawer task where each requires pick and place of six objects each, and N=10 for each visual prompting task, where each trial consists of pick and place of three or five objects

Generalization. We evaluate whether the model can identify the correct cup for held-out sequences of five shuffles. We also test whether it can generalize to unseen sequences of six shuffles, despite being trained only on sequences of five.

Drawer Task (Memory Task) The drawer task consists of six drawers on a table, each containing an object. A human opens each drawer, removes the object, and places it at a random location on the table. Once all objects have been removed, the robot must return each object to its original drawer. This task spans approximately 65 seconds on average, occasionally lasting up to 80 seconds, and requires the model to remember the association between each object and its original drawer.

Generalization. We evaluate whether the model can return objects to their original drawers for object-to-drawer assignments not seen during training.

Sorting (Visual Prompting Task) In this task, we use the memory context to prompt the robot with a human demonstration. The setup consists of three bins and three objects on a table. A human records a ten-second video demonstrating which bin each object should be placed in. We provide this video as the model's memory context, and the robot must use the demonstration to place each object in the correct bin.

Generalization. We evaluate whether the model can sort up to five objects using the visual prompt, despite being trained only to sort three objects at a time.

5 RESULTS

Q1: Is subtask prediction sufficient for long-horizon memory tasks, or does narrating the past improve performance? Our results show that narrating the memory context improves performance on both the cup swap and drawer tasks. In the cup swap task, subtask prediction alone achieves strong performance on in-distribution sequences. However, performance drops substantially on unseen sequences, including those with six swaps rather than the five seen during training. Training the VLA to narrate past subtasks improves generalization to these scenarios, and adding parallel-decoded latent narration further improves performance. We observe a similar trend in the drawer task, although the gains are smaller.

We conjecture that the benefits of narration depend on the memory demands of the task. In the cup swap task, all cups appear identical, and the model must correctly track each swap from the beginning of the sequence. Success therefore requires combining information across a chain of events. In contrast, the drawer task requires remembering individual object-to-drawer associations, which can be recalled independently without reconstructing the full sequence of events.

We also compare against Pi0.7 and MeMER. Pi0.7 performs near chance on the cup swap task and achieves low placement success on the drawer task. We suspect that its compression of visual history into 256 tokens per camera limits its ability to retain the details needed for these tasks.

Table 1: **RoboMME category-level success rates (%).** AVG is averaged over all sixteen tasks. Gray rows are reference-only and excluded from ranking. **Bold** and underline indicate the best and second-best results among the compared methods. LongContextVLA is evaluated on 50 test episodes per task (800 episodes in total).

Memory	Method	Integr. / VLM	Counting	Permanence	Reference	Imitation	AVG
Human	Human Performance	–	88.5	91.0	93.0	89.5	90.5
Symbolic (Oracle)	SimpleSG (Dai et al., 2026)	GT VLM	82.6	21.6	32.3	61.9	49.6
Symbolic (Oracle)	GroundSG (Dai et al., 2026)	GT VLM	83.9	93.3	95.2	64.0	84.1
Symbolic	SimpleSG	Gemini	39.0	13.5	22.5	18.0	23.3
Symbolic	SimpleSG	QwenVL	44.6	19.6	25.2	26.6	29.0
Symbolic	GroundSG	Gemini	12.8	15.8	11.3	6.5	11.6
Symbolic	GroundSG	QwenVL	38.0	39.3	31.6	21.9	32.7
Perceptual	TokenDrop (Dai et al., 2026)	Context	57.9	26.9	23.7	29.5	34.5
Perceptual	TokenDrop	Modul	52.3	26.8	34.7	38.3	38.0
Perceptual	TokenDrop	Expert	59.4	23.6	25.3	31.1	34.9
Perceptual	FrameSamp (Dai et al., 2026)	Context	50.1	23.3	20.9	28.3	30.7
Perceptual	FrameSamp	Modul	65.2	25.1	36.3	<u>51.4</u>	<u>44.5</u>
Perceptual	FrameSamp	Expert	<u>66.8</u>	25.2	24.1	28.9	36.3
Recurrent	TTT (Dai et al., 2026)	Context	36.1	21.4	22.0	9.7	22.3
Recurrent	TTT	Modul	34.5	22.2	20.4	10.7	22.0
Recurrent	TTT	Expert	36.0	22.9	19.7	10.8	22.3
Recurrent	RMT (Dai et al., 2026)	Context	32.2	15.4	19.7	10.6	19.5
Recurrent	RMT	Modul	34.1	15.8	21.1	9.7	20.2
Recurrent	RMT	Expert	34.4	15.7	14.7	7.8	18.1
Other	$\pi_{0.5}$ (Intelligence et al., 2025)	–	28.8	17.0	17.2	8.8	17.9
Other	WeaveVLA (Zhu et al., 2026)	–	31.0	19.5	12.5	9.5	18.1
Other	$\pi_{0.5}$ w/ past actions (Dai et al., 2026)	–	29.1	22.8	15.9	11.2	19.7
Other	SAM2Act+ (Fang et al., 2025)	–	35.3	26.0	16.8	7.3	21.4
Other	MemER (Sridhar et al., 2026)	–	48.8	<u>53.2</u>	<u>38.0</u>	29.5	42.4
Variant	SimpleMemVLA (Retrieval)	–	46.5	42.0	13.5	24.0	31.5
Variant	SimpleMemVLA (Token Compression)	–	46.0	13.5	12.5	18.5	22.6
Variant	SimpleMemVLA (Recurrent State)	–	31.0	24.0	17.5	10.0	20.6
Ours	Long Context VLA	–	**86.5**	**94.0**	**84.5**	**74.5**	**84.9**

MeMER also struggles on both tasks. We observed failures involving incorrect keyframe selection or selected keyframes that differed from those encountered during training, suggesting that sensitivity to keyframe selection may limit its generalization.

Q2: Can VLAs maintain real-time performance with context lengths of up to one million tokens? As shown in Fig. 5, caching substantially reduces inference latency as the memory context grows. Cached inference remains below one second at approximately 258K visual tokens and takes approximately 1.04 seconds at 516K tokens on both GPUs. At one million visual tokens, latency reaches 1.82 seconds on B200 and 1.78 seconds on H100. These results demonstrate that our approach can process substantially longer visual histories with modest increases in latency, although it exceeds a one-second inference budget at the longest contexts. Further optimizations to action decoding, such as reducing the number of tokens used to represent an action trajectory, may help reduce this latency.

Q3: Can models trained to use memory contexts also use human demonstrations as visual prompts? Our results show that the memory context can also be used to prompt robots with human demonstrations. Although our model was trained to sort three objects, it generalizes to sorting five objects, achieving a 90% success rate compared with 100% in the training setting. MeMER also performs well, achieving 100% and 80%, respectively, consistent with its ability to use video demon-



Figure 5: Inference latency as a function of historical visual tokens on H100 and B200 GPUs, with and without caching. Both axes use logarithmic scales.

strations. Pi0.7 achieves 70% in the training setting but drops to 20% with five objects. Despite its strong pick-and-place capabilities, it struggles to follow the demonstrated sorting assignments, suggesting that interpreting the visual prompt is a limiting factor.

RoboMME Results Table 1 reports our results on RoboMME. Our approach achieves the highest overall score among the compared methods. Peg insertion remains particularly challenging and is our lowest-scoring task. We suspect that reconstruction errors in the OAT action tokenizer limit the precision required for insertion. Improving the tokenizer's reconstruction accuracy may improve performance on this task and potentially on other tasks as well. For our RoboMME tasks, we used up to 256 history images total.

6 Conclusion and Future Work

We presented LongContextVLA, a vision-language-action model that conditions on hundreds of past images and still acts in real time. Three design choices make this possible: dense visual memory inherited from a long-context VLM, a narration objective that teaches the policy to explicitly recount its timestamped history before acting, and a streaming inference scheme that caches history state across decisions and decodes narration and action tokens in parallel. Our experiments show that long context on its own is not enough: subtask prediction suffices in distribution, but narrating the past is what lets the policy generalize to unseen shuffle sequences, drawer assignments, and object counts, and parallel decoding preserves these gains at little added latency. On RoboMME, the same recipe outperforms symbolic, perceptual, and recurrent memory baselines across all four categories, and the same memory context can be repurposed as a visual prompt, letting the robot imitate a short human video it has never seen.

Several limitations point to future work. Our narration labels are obtained by concatenating timestamped subtask annotations, so the model only learns to recount events that the labeling scheme names; richer, automatically generated descriptions of spatial layout and object state could provide a denser signal for tasks that hinge on fine visual detail. Because new tokens still attend over the entire retained history, per-step cost grows with episode length, and scaling from minutes to hours will likely require combining dense context with learned compression or retrieval over the cache rather than replacing it. Our history is also drawn from a single top-down camera and from demonstrations collected in a small set of scenes; extending the memory to multiple views and training on larger, more diverse data is a natural next step toward a generalist long-context policy. Finally, video prompting suggests a broader opportunity: a policy that can read its own past could also learn from it, using memory of prior attempts and corrections to improve at deployment time.

AI USE STATEMENT

In this work, we used generative AI tools to propose and refine research ideas, to provide feedback on research methodology and experiment design, and to implement methods, where LLM-based coding assistants helped write, refactor, and debug parts of the training, inference, and evaluation code. We have not used generative AI tools to generate synthetic data sets, develop theoretical models or conceptual frameworks, formulate mathematical claims, clean or reformat datasets, or interpret results, and assisting with the proof of mathematical claims, translation, and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for brainstorming, identifying and summarizing relevant literature, sourcing information, creating and editing software code, drafting parts of the paper, editing the paper to improve readability, and formatting references. We have reviewed all AI-assisted work. All LLM-assisted code was reviewed and tested by the authors and validated through the real-robot experiments reported in this paper; AI-suggested ideas and methodological feedback were adopted only after the authors' own assessment, and the final method, experiment design, demonstration collection, narration annotation, and interpretation of results are the authors' own; literature identified with AI tools was read and checked against the original sources before being cited; and all AI-drafted or AI-edited text was verified by the authors for factual accuracy and consistency with our method and results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Shuanghao Bai, Jing Lyu, Wanqi Zhou, Zhe Li, Dakai Wang, Lei Xing, Xiaoguang Zhao, Pengwei Wang, Zhongyuan Wang, Cheng Chi, Badong Chen, and Shanghang Zhang. Latent reasoning VLA: Latent thinking and prediction for vision-language-action models. *arXiv preprint arXiv:2602.01166*, 2026.

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, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 27–30 Sep 2025.

William Chen, Suneel Belkhale, Suvir Mirchandani, Karl Pertsch, Danny Driess, Oier Mees, and Sergey Levine. Training strategies for efficient embodied reasoning. In *Conference on Robot Learning (CoRL)*, 2025. arXiv:2505.08243.

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*, 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 *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=8m30ogkPk2.

Nitish Dashora, Douglas Chen, Idan Shenfeld, John Marangola, Pulkit Agrawal, and Max Simchowitz. Workspace models: Lightweight robotic memory via saliency-driven supervision, 2026. URL https://arxiv.org/abs/2609.20820.

Zhekai Duan, Yuan Zhang, Shikai Geng, Gaowen Liu, Joschka Boedecker, and Chris Xiaoxuan Lu. Fast ECoT: Efficient embodied chain-of-thought via thoughts reuse. *arXiv preprint arXiv:2506.07639*, 2025.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems*, 2024. URL https://doi.org/10.15607/RSS.2024.XX.090.

Huiwon Jang, Sihyun Yu, Heeseung Kwon, Hojin Jeon, Younggyo Seo, and Jinwoo Shin. Contextvla: Vision-language-action model with amortized multi-frame context. *arXiv preprint arXiv:2510.04246*, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=ZMnD6QZAE6.

Myungkyu Koo, Daewon Choi, Taeyoung Kim, Kyungmin Lee, Changyeon Kim, Younggyo Seo, and Jinwoo Shin. HAMLET: Switch your vision-language-action model into a history-aware policy. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.00695.

Chaoqi Liu, Yue Zhao, Haonan Chen, Xiaoshen Han, Jiawei Gao, Ehsan Adeli, and Yilun Du. Ordered action tokens for visuomotor policy learning, 2026. URL https://arxiv.org/abs/2607.21670.

Max Sobol Mark, Jacky Liang, Maria Attarian, Chuyuan Fu, Debidatta Dwibedi, Dhruv Shah, and Aviral Kumar. Bpp: Long-context robot imitation learning by focusing on key history frames. *arXiv preprint arXiv:2602.15010*, 2026.

Physical Intelligence. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities. *arXiv preprint arXiv:2604.15483*, 2026.

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

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.

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.

Marcel Torne, Andy Tang, Yuejiang Liu, and Chelsea Finn. Learning long-context diffusion policies via past-token prediction. In *Conference on Robot Learning (CoRL)*, 2025. arXiv:2505.09561.

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.

Cheng Yin, Wang Xu, Junpeng Yang, Sikyuen Tam, Hanyu Liu, Yuan Yao, Xiangrui Zeng, Junbo Cui, Yequan Wang, Zhouping Yin, and Yankai Lin. SimpleMemVLA: A simple but effective native-video memory for vision-language-action models. *arXiv preprint arXiv:2609.05533*, 2026.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In *Conference on Robot Learning (CoRL)*, 2024. arXiv:2407.08693.

Zhide Zhong, Junfeng Li, Junjie He, Haodong Yan, Xin Gong, Guanyi Zhao, Yingjie Cai, Jiantao Gao, Xu Yan, Bingbing Liu, Yingcong Chen, Liuqing Yang, and Haoang Li. DualCoT-VLA: Visual-linguistic chain of thought via parallel reasoning for vision-language-action models. *arXiv preprint arXiv:2603.22280*, 2026.

A PER-TASK ROBOMME RESULTS

Table 1 reports category-level averages on RoboMME (Dai et al., 2026). Here we break these averages down into the sixteen individual tasks. Figure 6 compares LongContextVLA against every baseline on each task, and Table 2 lists our per-task success rates. All of our numbers are computed over 50 test episodes per task (800 episodes in total).

LongContextVLA reaches at least 84% success on thirteen of the sixteen tasks, and its permanence average (94.0%) exceeds the human reference (91.0%) in Table 1. The remaining three tasks are VideoRepick (76%), BinFill (62%), and InsertPeg (18%). InsertPeg alone accounts for the lower imitation average: excluding it, the other three imitation tasks average 93.3%.

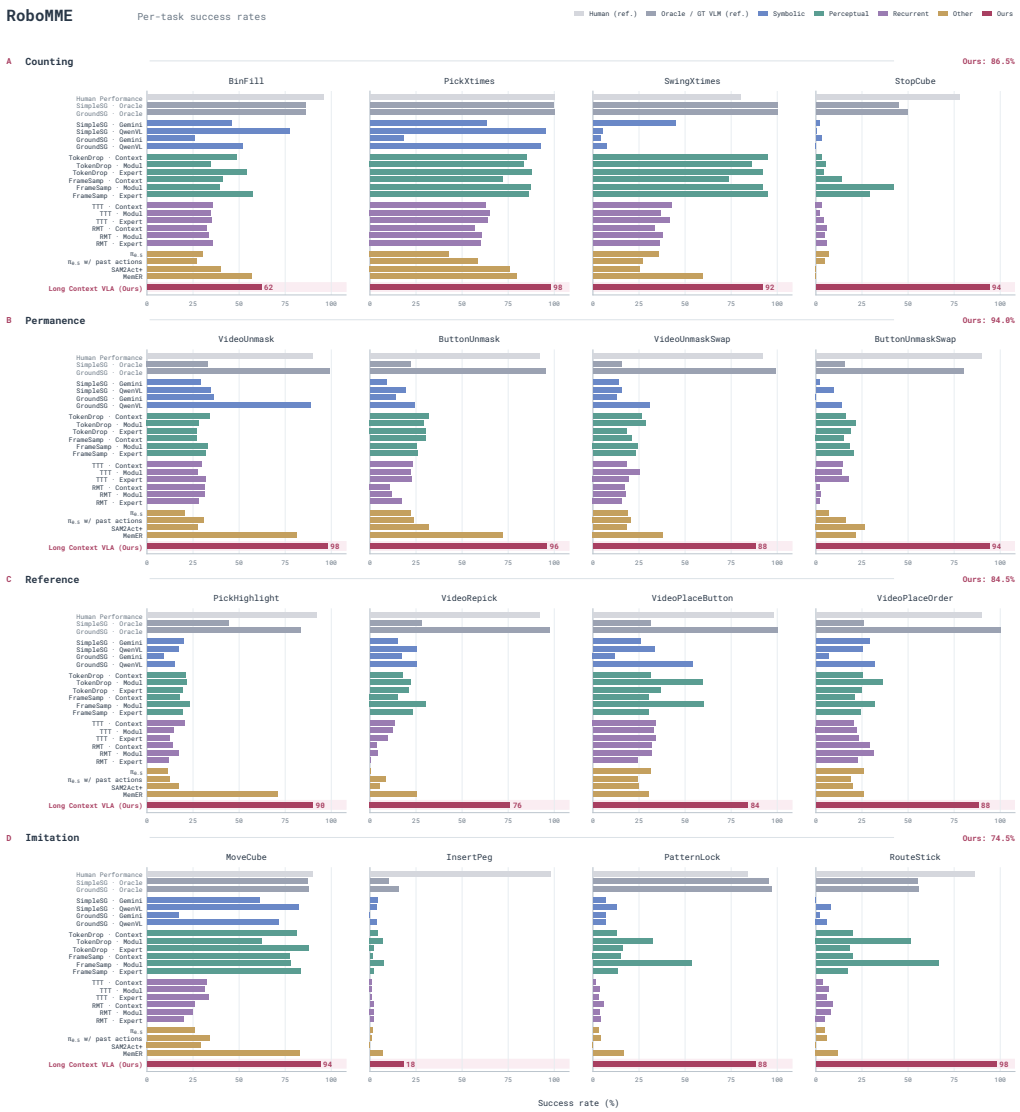


Figure 6: **RoboMME per-task success rates (%).** Each panel shows one of the sixteen RoboMME tasks, grouped by memory category into rows: (A) counting, (B) permanence, (C) reference, and (D) imitation. The right end of each row header reports our category-level average. Baselines are grouped by memory type as in Table 1; human and oracle (ground-truth VLM) rows are reference only. LongContextVLA is the bottom bar of each panel and is evaluated on 50 test episodes per task.

Table 2: **LongContextVLA per-task success rates (%) on RoboMME.** Category averages match the corresponding columns of Table 1. Each entry is computed over 50 test episodes.

Counting		Permanence		Reference		Imitation	
Task	SR	Task	SR	Task	SR	Task	SR
BinFill	62	VideoUnmask	98	PickHighlight	90	MoveCube	94
PickXtimes	98	ButtonUnmask	96	VideoRepick	76	InsertPeg	18
SwingXtimes	92	VideoUnmaskSwap	88	VideoPlaceButton	84	PatternLock	88
StopCube	94	ButtonUnmaskSwap	94	VideoPlaceOrder	88	RouteStick	98
Average	86.5	*Average*	94.0	*Average*	84.5	*Average*	74.5

B Training and Implementation Details

We use Qwen3.5-4B as the vision-language backbone and OAT as the action tokenizer. The visual encoder, visual projection, and language backbone are all fine-tuned. The action tokenizer is fitted separately and remains frozen during policy training. Batch sizes below are global effective batch sizes, including gradient accumulation. Training iterations refer to optimizer updates.

B.1 Policy training

Hyperparameter	**Cup**	**Drawer**	**RoboMME**
Global batch size	64	64	512
Maximum policy updates	4,000	6,000	6,000
Learning rate	10^{-5}	10^{-5}	10^{-5}
Linear warmup updates	20	20	100
Schedule after warmup	Constant	Constant	Constant
Optimizer	AdamW	AdamW	AdamW
Adam coefficients	$(0.9, 0.999)$	$(0.9, 0.999)$	$(0.9, 0.999)$
Adam epsilon	10^{-8}	10^{-8}	10^{-8}
Weight decay	0	0	0
Gradient norm clipping	1.0	1.0	1.0
Random seed	42	42	42
Parameter precision	BF16	BF16	BF16
Optimizer master / accumulation	FP32 / FP32	FP32 / FP32	FP32 / FP32
Distributed optimizer	ZeRO stage 2	ZeRO stage 2	ZeRO stage 2
Gradient checkpointing	Enabled	Enabled	Enabled

B.2 Backbone and attention

The language backbone has 32 layers, hidden width 2,560, and feed-forward width 9,216. Each group of four layers contains three GatedDeltaNet layers followed by one full-attention layer, giving 24 recurrent layers and eight full-attention layers. Full-attention layers use 16 query heads and four key/value heads, with head dimension 256. GatedDeltaNet uses a causal convolution of width four, 16 key heads, and 32 value heads; the key and value head dimensions are 128.

The vision encoder has 24 layers, hidden width 1,024, and 16 attention heads. It uses 16×16 spatial patches, a temporal patch size of two, and 2×2 spatial merging. Its output projection maps visual features to the language backbone's width of 2,560.

The decoder is causal. Historical video precedes current wrist images, the task instruction, and proprioception. The policy predicts language and action tokens through the same output head. Training uses FlashAttention 2 for full-attention layers. Reserved narration positions and action-query positions retain causal attention; no bidirectional mask is introduced.

B.3 VISUAL INPUTS AND HISTORY BUDGET

Setting	Cup	Drawer	RoboMME
History camera	Top-down	Top-down	Front
History image size (H by W)	288×512	288×512	256×256
Recorded frame rate	30 Hz	30 Hz	20 Hz
History stride (recorded frames)	15	15	10
History sampling rate	2 Hz	2 Hz	2 Hz
Maximum history images	30	160	256
Current wrist images	2	2	1
Maximum images including wrists	32	162	257
History tokens per temporal pair	144	144	64
History tokens per image (pair average)	72	72	32
Maximum history visual tokens	2,160	11,520	8,192
Visual tokens per current wrist	120	120	64
Maximum visual tokens including wrists	2,400	11,760	8,256
Reserved narration positions (P mode)	96	192	336
Visual tokens plus P-mode positions	2,496	11,952	8,592

B.4 OAT ARCHITECTURE AND ACTION PREDICTION

Setting	Cup	Drawer	RoboMME
Action dimensions	10	10	10
Action horizon (steps)	30	30	30
Encoder / decoder layers	6 / 6	6 / 6	6 / 6
Transformer width	256	256	256
Attention head dimension	32	32	32
Tokenizer dropout	0.1	0.1	0.0
Register tokens / action codes	32	32	16
FSQ levels	$(8, 8, 8, 8)$	$(8, 8, 8, 8)$	$(8, 8, 6, 5)$
Code vocabulary size	4,096	4,096	1,920
Policy prediction groups	1, 1, 2, 4, 8, 16	1, 1, 2, 4, 8, 16	16 singleton groups
Sequential action prediction calls	6	6	16
Tokenizer updated with policy	No	No	No

B.5 ACTION REPRESENTATION

Each action consists of three end-effector position coordinates, six rotation coordinates formed from the first two columns of the rotation matrix, and one gripper command. The first nine coordinates are expressed relative to the measured pose at the start of the chunk. The same anchor is used for every future action in that chunk. The gripper command remains absolute. Actions are quantile-normalized before tokenization. After decoding, normalization is inverted and the anchor is added back. The rotation columns are orthonormalized before inverse kinematics.

OAT compresses a continuous action chunk into learned register tokens and quantizes them with finite scalar quantization (FSQ). Earlier code groups condition later groups. Codes within the current group are predicted together from query placeholders, without exposing that group's targets. Prediction logits are restricted to the action-code vocabulary.

B.6 TOKENIZER FITTING

Setting	Cup	Drawer	RoboMME selected codec
Fitting batch size	2,048	2,048	2,048
Adam coefficients	$(0.9, 0.95)$	$(0.9, 0.95)$	$(0.9, 0.95)$
Weight decay / gradient clip	0 / 1.0	0 / 1.0	0 / 1.0
Learning rate	4×10^{-4}	4×10^{-4}	5×10^{-5} (continuation)
Warmup updates	500	500	50 (continuation)
Selected cumulative update	20,000	60,000	33,000
Sampled code-prefix lengths	16, 32	16, 32	1, 2, 4, 8, 16
Prefix sampling probabilities	0.5, 0.5	0.5, 0.5	0.15, 0.15, 0.15, 0.20, 0.35

B.7 TOKENIZER FITTING PROTOCOL

Cup and drawer tokenizer fitting minimizes reconstruction error in normalized action space. The fitting schedule warms up linearly, remains constant until update 36,000, then decays with a cosine schedule to 2% of the initial learning rate at update 60,000. The selected cup tokenizer is the 20,000-update snapshot, before cosine decay begins. The selected drawer tokenizer is the 60,000-update snapshot.

The selected RoboMME tokenizer is an exponential-moving-average checkpoint from a continuation of an earlier fit. The table's learning rate and warmup describe that continuation, not its entire lineage. The continuation uses normalized reconstruction loss with additional full-prefix position, threshold, and tail-error terms. Their coefficients are 1.0, 1.0, and 0.3, respectively. The physical position scale is 0.1 m and the threshold is 0.008 m. Half of the sampled windows are drawn from a hard-example pool. EMA uses a reference decay of 0.995 per 1,024 examples. All 1,600 official training demonstrations supply tokenizer fitting and normalization; the reconstruction audit is in-sample.