MemoryVAM: Integrating Memory into Video Action Model for Robot Manipulation

Yuxin Jiang[1*], **Chang Yu**[1*], **Yunuo Chen**[1], **Xiang Feng**[1,3], **Yin Yang**[4], **Nishank Gite**[2], **Chenfanfu Jiang**[1]
[1]University of California, Los Angeles, [2]Nirvana Robotics,
[3]University of California, San Diego, [4]University of Utah
* Equal contributions.

Abstract: Video-world-model policies often fail when the correct action depends on earlier events outside the current observation window. We present **MemoryVAM**, an episodic memory mechanism that injects compact history tokens into both the video backbone and action decoder. A Perceiver-based Recap Compressor summarizes CLIP-encoded episode history, while a lightweight Cue Gate predicts task completion from memory and language. The module is trained with video prediction, delta reconstruction, and episode-boundary supervision, requiring no dense progress labels. Across UNet and DiT backbones, MemoryVAM improves LIBERO-Mem average success from 5% to 42.5%; on real robots, it achieves 78.3% success on counting, 80.0% on spatial recall, and 75.0% on sequential tracking. Project page: https://MemoryVAM.github.io/.

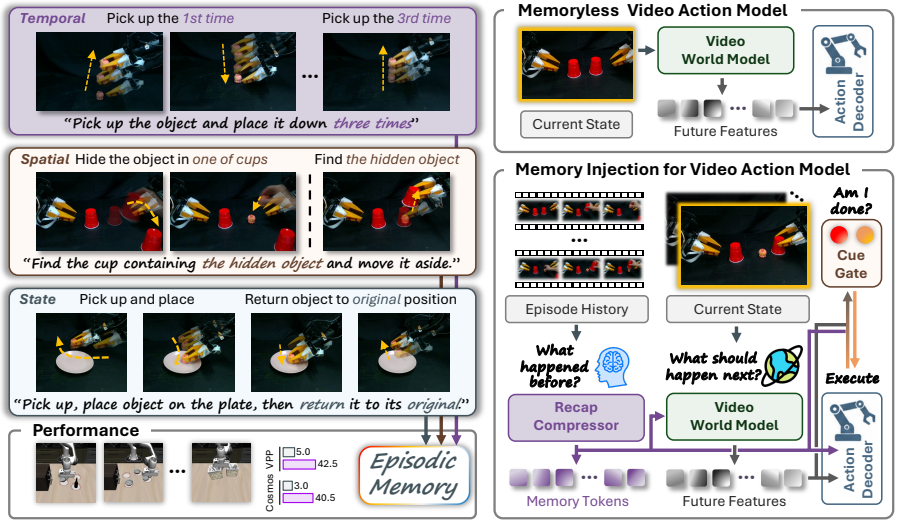

Figure 1: **Method Overview.** Memory-dependent manipulation requires recalling temporal progression, latent state, and spatial history. MemoryVAM introduces episode-level memory to enable history-aware prediction, task completion, and action decoding.

1 Introduction

Video world models are increasingly used as representation engines for robot policies: the model predicts plausible future observations, and the policy decodes actions from the resulting visual features [1, 2, 3]. This is effective when the current observation is a sufficient state, but many manipulation tasks are not Markovian under a short image window. Repeated pickups, temporary occlusion, and place-then-return behaviors can produce visually similar frames whose correct continuation depends on hidden episode progress.

As sketched in Figure 1, we argue that robot world models need an explicit episode state, not only a larger local context window. For a video-action world model, this state should shape both what future the model imagines and which action the decoder selects. **MemoryVAM** addresses this by injecting the same compact memory tokens into the video backbone and the action decoder, avoiding a split between a history-aware controller and a history-agnostic world model.

Accepted to the RSS 2026 Workshop on Robot World Models.

Our contribution is a simple dual-conditioning interface for memory-aware robot world models. A Recap Compressor summarizes episode history into fixed-size tokens, a Cue Gate learns task completion from demonstration endpoints, and cross-attention adapts UNet- and DiT-based video-action policies. Experiments on LIBERO-Mem [4] and real manipulation tasks show that this shared memory state improves both imagined rollouts and policy success when the current image alone is ambiguous.

2 Related Work

Robot world models. Video prediction has been used as a policy interface, subgoal generator, or representation source for robot learning [5, 6, 7, 1, 8]. Recent systems also adapt large generative video backbones, including diffusion and transformer-based models, for action prediction [9, 10, 2, 11, 3]. These approaches provide strong short-horizon predictive features, but most condition on recent frames and language rather than a persistent episode state.

Memory for robot policies. Memory has been explored through long-context vision-language models, compressed histories, recurrent policies, and explicit memory modules [12, 13, 14, 15, 16, 17, 18]. We study a complementary question: for policies built around a video world model, where should episode memory enter? MemoryVAM treats memory as part of the world-model state itself, conditioning both future prediction and action decoding rather than appending history only to the policy head.

3 Method

At timestep t, the policy observes a short local image window $\mathbf{I}_{t-r+1:t}$ and instruction ℓ. MemoryVAM adds an episode memory $\mathbf{G}_t$ computed from earlier frames $\mathcal{H}_t = \{\mathbf{I}_0, \dots, \mathbf{I}_{t-1}\}$. A video backbone p_ϕ predicts future observations and exposes denoising features, while an action decoder π_θ predicts an action chunk:

$$\hat{\mathbf{I}}_{t+1:t+H_v} \sim p_\phi(\cdot \mid \mathbf{I}_{t-r+1:t}, \ell, \mathbf{G}_t), \qquad \hat{\mathbf{A}}_t \sim \pi_\theta(\cdot \mid f_\phi(\mathbf{I}_{t-r+1:t}, \ell, \mathbf{G}_t), \mathbf{G}_t, \ell).$$

As shown in Figure 2, the goal is to make $\mathbf{G}_t$ a shared progress state for both imagined futures and control.

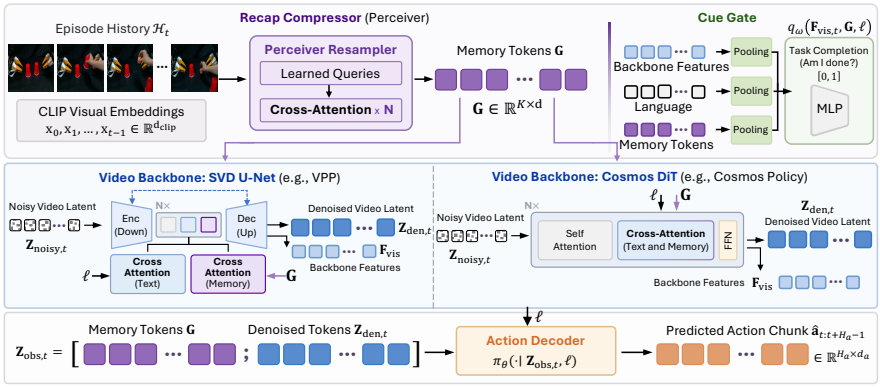

Figure 2: **MemoryVAM architecture.**

Recap memory. Each history frame is embedded as $\mathbf{x}_i = \text{CLIP}_{\text{vis}}(\mathbf{I}_i)$ [19]. A Perceiver-style resampler [20, 21, 22] maps the variable-length sequence into K fixed memory tokens:

$$\mathbf{G}_t = \text{Recap}_\psi(\mathbf{x}_{0:t-1}) \in \mathbb{R}^{K \times d}.$$

Learned queries select events that are useful for explaining task progress, while positional encodings preserve temporal order. This gives the rest of the policy a constant-size memory interface independent of episode length.

Dual memory conditioning. The same memory tokens are injected through two pathways. First, $\mathbf{G}_t$ conditions the video backbone's cross-attention layers, so the predicted future is consistent with episode history. For DiT backbones, memory is appended to the text-conditioning matrix; for UNet

backbones, we use a separate memory cross-attention branch with layer-specific projections. Second, $\mathbf{G}_t$ is exposed directly to the action decoder, either by concatenating it with denoised visual tokens or by appending projected memory tokens to the decoder's visual conditioning. These two pathways avoid a mismatch between a history-aware controller and a history-agnostic world model.

Training and completion. Training uses two stages. Stage 1 trains the video backbone and Recap memory with the video prediction objective, a delta-reconstruction auxiliary loss that encourages memory to preserve changes across the episode, and a Cue Gate loss. The Cue Gate predicts a completion score $q_\omega(f_\phi, \mathbf{G}_t, \ell)$ using final demonstration frames as positives and earlier frames as negatives, requiring no dense progress annotation. Stage 2 trains the action decoder on memory-conditioned backbone features. At inference time, the same Cue Gate supplies an autonomous stopping signal for tasks where completion is not visible in the current frame.

4 Experiments

We evaluate whether episodic memory improves both the predictive backbone and the downstream robot policy. We first assess video prediction quality in memory-dependent scenarios, then evaluate policy success on LIBERO-Mem [4] and real-robot tasks that require recalling earlier episode events.

4.1 Experimental Setup

Backbones and variants. We evaluate MemoryVAM on two representative video backbone families: the UNet-based VPP [1] and the DiT-based Cosmos-RF built upon Cosmos [2, 3]. Each no-memory baseline keeps the same backbone and action module while removing episodic memory. On VPP, we also report single-path ablations that retain only Pathway 1 or only Pathway 2, isolating memory used by the predictive backbone from memory exposed directly to the action decoder.

Baselines and tasks. External baselines include memoryless VLA references π_0 [23] and SlotVLA [4]; memory-augmented VLA methods SlotVLA-with-history [4], E-SlotSSM [4], and MemoryVLA [15]; and the memory-augmented diffusion-policy method Gated Memory Policy [14]. We evaluate on LIBERO-Mem, a 10-task simulation benchmark spanning counting, sequencing, and spatial reasoning, and on real-robot tasks that test counting, spatial recall, and occluded-object tracking.

4.2 Video Prediction Quality

Memory improves history alignment during autoregressive rollout. Compared to VPP, our UNet-based model reduces FVD from 43.36 to 18.32 at H_v and from 37.81 to 19.84 at $8H_v$. Compared to Cosmos-RF, our DiT-based model reduces single-chunk FVD from 22.54 to 18.65 and long-horizon FVD from 26.11 to 16.24 at $8H_v$. These results suggest that memory is most useful when predictions must remain aligned with hidden task progress over long horizons.

4.3 Main Benchmark Results

Table 1 shows that the main challenge lies in recovering hidden task progress rather than executing isolated pick-and-place motions. Memoryless policies can sometimes solve single-step variants, but performance drops when visually similar scenes correspond to different counts, object histories, or sequence states. Memory injection improves both VPP and Cosmos-RF, indicating that the mechanism is not tied to one video backbone family.

4.4 Real Robot Experiments

We evaluate our model on three real-robot tasks, with task illustrations deferred to Figure 3. *Toy Pickup* $k\times$ tests counting, *Place and Return* requires remembering the initial object position after an intermediate placement, and *Shell Game* requires tracking an occluded object after cup swapping. The common stressor is observation aliasing: at key decision points, the current image is ambiguous because multiple episode states can produce similar observations.

Table 2 follows the same trend as LIBERO-Mem. The memoryless Cosmos-RF policy collapses when task progress is not observable, while our memory-injected policy achieves 78.0% average

Table 1: **LIBERO-Mem Results.** Success rate (%) over 20 rollouts per task. Columns cover single placement, counting, and sequencing/spatial tasks. Abbreviations: B=Bowl, Bt=Bottle, P=plate, Cs=Cheese, Bskt=basket.

	Single		Counting				Sequencing / Spatial				
Method	B→P	Bt→P	B×3	Bt×3	B×5	B×7	Swap2	Rot3	Cs→Bskt	Cs+E	Avg
VPP backbone (SVD UNet, 1.5B)											
No memory	30.0	10.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0
Pathway 1 only	**85.0**	40.0	35.0	10.0	15.0	0.0	5.0	0.0	30.0	65.0	28.5
Pathway 2 only	45.0	10.0	5.0	0.0	5.0	5.0	0.0	0.0	5.0	0.0	7.5
Ours (UNet)	**85.0**	**65.0**	**90.0**	10.0	35.0	0.0	**25.0**	5.0	**40.0**	**70.0**	**42.5**
Cosmos-DiT backbone (rectified-flow, 2B)											
No memory	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	15.0	10.0	3.0
Ours (DiT)	75.0	**65.0**	65.0	**15.0**	**55.0**	**35.0**	10.0	**10.0**	25.0	50.0	40.5
VLA baselines, no memory											
π_0 [23]	50.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0
SlotVLA [4]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
VLA baselines, memory-augmented											
SlotVLA$_{h=8}$ [4]	50.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0
E-SlotSSM [4]	50.0	0.0	35.0	0.0	15.0	0.0	0.0	0.0	30.0	20.0	15.0
MemoryVLA [15]	55.0	0.0	35.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	9.5
Diffusion-policy baseline, memory-augmented											
Gated Memory Policy [14]	**85.0**	45.0	20.0	5.0	40.0	0.0	0.0	0.0	0.0	0.0	19.5

Table 2: **Real-Robot Results.** Success rate (%) across real-robot memory tasks.

	Toy Pickup k×				*Place and Return*	*Shell Game*
Method	1×	3×	5×	Avg		
Cosmos-RF (no mem)	0.0	0.0	0.0	0.0	0.0	5.0
Ours (DiT)	**95.0**	**100.0**	**40.0**	**78.3**	**80.0**	**75.0**

success across five real-robot evaluation settings. These results indicate that the performance gains transfer to physical rollouts, where history is essential for disambiguating the next action.

4.5 Ablations and Diagnostics

The pathway ablations in Table 1 show that memory is most effective when it shapes both imagined futures and downstream actions. Pathway 1, which injects memory into the predictive backbone only, reaches 28.5% average success, indicating that history-aware visual features already carry substantial task-progress information. Pathway 2, which exposes memory only to the action decoder, reaches 7.5%, suggesting that a history-aware controller cannot fully compensate for a history-agnostic world model. The full dual-injection model reaches 42.5%.

We further evaluate whether the memory state supports completion awareness in Table 4. The reconstruction auxiliary loss reduces mean completion-timing error from 44.3 to 35.5 frames at $p \geq 0.5$ and from 62.0 to 28.8 frames at $p \geq 0.9$, producing a sharper Cue Gate. In generated rollouts, memory also reduces unnecessary post-completion motion to less than half of pre-completion motion, explaining why the same mechanism transfers to real-robot settings where the current frame alone is ambiguous.

5 Conclusion

MemoryVAM treats episodic memory as shared state for video-action world models rather than as an action-head add-on. By conditioning both future prediction and action decoding on the same compact history tokens, it improves simulation and real-robot tasks that require recovering hidden progress. The results suggest that memory interfaces can be portable across UNet and DiT backbones while using supervision from video prediction and episode endpoints.

References

[1] Y. Hu, Y. Guo, P. Wang, X. Chen, Y.-J. Wang, J. Zhang, K. Sreenath, C. Lu, and J. Chen. Video prediction policy: A generalist robot policy with predictive visual representations. In *Forty-second International Conference on Machine Learning*.

[2] N. Agarwal, A. Ali, M. Bala, Y. Balaji, E. Barker, T. Cai, P. Chattopadhyay, Y. Chen, Y. Cui, Y. Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

[3] M. J. Kim, Y. Gao, T.-Y. Lin, Y.-C. Lin, Y. Ge, G. Lam, P. Liang, S. Song, M.-Y. Liu, C. Finn, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

[4] N. Chung, T. Hanyu, T. Nguyen, H. Le, F. Bumgarner, D. M. H. Nguyen, K. Vo, K. Yamazaki, C. Rainwater, T. Kieu, et al. Rethinking progression of memory state in robotic manipulation: An object-centric perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 3407–3415, 2026.

[5] Y. Du, S. Yang, B. Dai, H. Dai, O. Nachum, J. Tenenbaum, D. Schuurmans, and P. Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.

[6] K. Black, M. Nakamoto, P. Atreya, H. Walke, C. Finn, A. Kumar, and S. Levine. Zero-shot robotic manipulation with pre-trained image-editing diffusion models. In *International Conference on Learning Representations*, volume 2024, pages 33431–33452, 2024.

[7] H. Wu, Y. Jing, C. Cheang, G. Chen, J. Xu, X. Li, M. Liu, H. Li, and T. Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, volume 2024, pages 10641–10662, 2024.

[8] B. Hou, G. Li, J. Jia, T. An, X. Guo, S. Leng, H. Geng, Y. Ze, T. Harada, P. Torr, et al. World model for robot learning: A comprehensive survey. *arXiv preprint arXiv:2605.00080*, 2026.

[9] J. Bruce, M. D. Dennis, A. Edwards, J. Parker-Holder, Y. Shi, E. Hughes, M. Lai, A. Mavalankar, R. Steigerwald, C. Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

[10] Y. Liao, P. Zhou, S. Huang, D. Yang, S. Chen, Y. Jiang, Y. Hu, J. Cai, S. Liu, J. Luo, et al. Genie envisioner: A unified world foundation platform for robotic manipulation. *arXiv preprint arXiv:2508.05635*, 2025.

[11] J. Pai, L. Achenbach, V. Montesinos, B. Forrai, O. Mees, and E. Nava. mimic-video: Video-action models for generalizable robot control beyond vlas. *arXiv preprint arXiv:2512.15692*, 2025.

[12] B. He, H. Li, Y. K. Jang, M. Jia, X. Cao, A. Shah, A. Shrivastava, and S.-N. Lim. Malmm: Memory-augmented large multimodal model for long-term video understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13504–13514, 2024.

[13] T. Kerssies, G. Berton, J. He, Q. Yu, W. Ma, D. de Geus, G. Dubbelman, and L.-C. Chen. A frame is worth one token: Efficient generative world modeling with delta tokens. *arXiv preprint arXiv:2604.04913*, 2026.

[14] Y. Gao, J. Liu, S. Li, and S. Song. Gated memory policy. *arXiv preprint arXiv:2604.18933*, 2026.

[15] H. Shi, B. Xie, Y. Liu, L. Sun, F. Liu, T. Wang, E. Zhou, H. Fan, X. Zhang, and G. Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*, 2025.

[16] A. Sridhar, J. Pan, S. Sharma, and C. Finn. Memer: Scaling up memory for robot control via experience retrieval. *arXiv preprint arXiv:2510.20328*, 2025.

[17] M. Torne, K. Pertsch, H. Walke, K. Vedder, S. Nair, B. Ichter, A. Z. Ren, H. Wang, J. Tang, K. Stachowicz, et al. Mem: Multi-scale embodied memory for vision language action models. *arXiv preprint arXiv:2603.03596*, 2026.

[18] Y. Dai, H. Fu, J. Lee, Y. Liu, H. Zhang, J. Yang, C. Finn, N. Fazeli, and J. Chai. Robomme: Benchmarking and understanding memory for robotic generalist policies. *arXiv preprint arXiv:2603.04639*, 2026.

[19] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[20] A. Jaegle, F. Gimeno, A. Brock, O. Vinyals, A. Zisserman, and J. Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR, 2021.

[21] A. Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.

[22] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

[23] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[24] H. Ye, J. Zhang, S. Liu, X. Han, and W. Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721*, 2023.

Supplementary Materials

A Additional Main-Paper Details

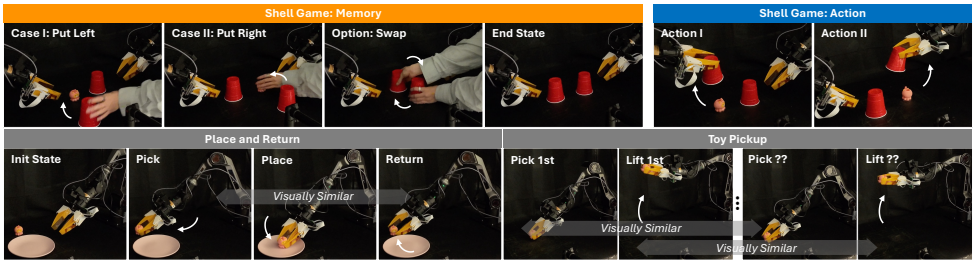

Figure 3: **Real-Robot Task Settings.** Shell Game, Place and Return, and Toy Pickup $k\times$ require recalling hidden object state, initial pose, or repetition count.

A.1 Ablation Studies

Completion timing. The reconstruction auxiliary loss reduces mean completion-timing error from 44.3 to 35.5 frames at $p \geq 0.5$, with the largest gains on repetitive tasks where counting matters. At the stricter threshold $p \geq 0.9$, the gain is larger, improving from 62.0 to 28.8 frames. This suggests that reconstruction produces a sharper and better-calibrated completion signal.

Video generation termination quality. We also evaluate termination behavior in generated video. A world model that understands task progress should not continue imagining unnecessary motion after the goal is complete. Without memory, the generated future continues moving after completion, producing a stop ratio above one. With memory injection, post-completion motion drops to less than half of pre-completion motion. This indicates that episodic memory helps the video backbone represent not only what should happen next, but also when no further motion is needed.

Dual injection. Table 1 shows that memory is most effective when it shapes both the predictive representation and the action decoder. Pathway 1 alone reaches 28.5% average success, indicating that history-aware backbone features carry substantial task-progress information. Pathway 2 alone reaches 7.5%, suggesting that exposing memory only at the action head cannot compensate for a history-agnostic backbone. The full model reaches 42.5%, showing that memory should shape both imagined futures and downstream actions.

B Conclusion

We presented **MemoryVAM**, an episodic memory module that is trained through the video generative objective and conditions both future prediction and action decoding via cross-attention. Across SVD UNet and Cosmos DiT backbones, the same memory mechanism improves long-horizon manipulation on LIBERO-Mem, raising average success from 5% to 42.5%, and transfers to real robots with 78.3% success on counting tasks, 80.0% on spatial recall, and 75.0% on sequential tracking. More broadly, we reframe episodic memory as an integral component of the video world model, rather than an auxiliary policy module. By allowing episode history to shape both what the robot imagines and how the policy acts, our approach points toward memory systems whose supervision scales with video data and world-model training, rather than relying solely on action-labeled demonstrations.

C Limitations

Parallel vs. recurrent memory. The Recap Compressor trades direct access to retained history for memory and computation that grow with episode length. Very long deployments may therefore require truncation, subsampling, or an online recurrent state, which we leave as future work within the same dual-injection interface.

Scaling. Our method suggests a scaling opportunity because its memory module can be supervised by video prediction rather than only by demonstrations with action labels. Our experiments validate this mechanism on LIBERO-scale robot data and two video backbone families, but they do not establish how performance changes with substantially broader video pre-training or larger backbones. Studying this scaling behavior is therefore a promising direction for future work.

D Architecture Details

D.1 Recap Compressor and History Cache

The Recap Compressor implements the Perceiver-style resampler described in Sec. 3. Each cached CLIP visual embedding $\mathbf{x}_i \in \mathbb{R}^{d_{\text{clip}}}$ is projected to the memory dimension d and augmented with temporal and visual-token positional encodings. The current implementation stores one global visual embedding per frame, so the visual-token position is constant while the temporal position indexes episode order. The K learned queries are updated through L cross-attention blocks with N_{head} attention heads and feed-forward width multiplier m_{ffn}. This design gives the video backbone and action decoder a constant-size memory interface after the variable-length history has been compressed.

CLIP ViT-B/32 visual embeddings are precomputed for all episodes and stored in a cache. Each frame contributes one global visual embedding. Histories are padded or truncated to $T_{\max}$ embeddings during training and capped at T_{hist} frames during evaluation, with a binary validity mask marking padded entries.

D.2 Backbone Memory Interface

For UNet video backbones, each text cross-attention layer receives a parallel memory branch with layer-specific key and value projections $\mathbf{W}^{\text{mem}}_{\text{key},m}$ and $\mathbf{W}^{\text{mem}}_{\text{val},m}$. The output is added to the original text-conditioned branch following decoupled cross-attention [24]. For DiT video backbones, memory tokens are appended to the conditioning matrix. In both cases, the Recap Compressor output $\mathbf{G}$ is unchanged, and only the injection interface depends on the backbone family.

D.3 Action Decoder Memory Interface

For token-input action decoders, Pathway 2 concatenates the memory tokens $\mathbf{G}$ with denoised video tokens $\mathbf{Z}_{\text{den},t}$. For cross-attention action transformers, projected memory tokens are appended to the visual conditioning sequence. These two instantiations implement the same conditional distribution in Sec. 3, where π_θ receives history either through $\mathbf{Z}_{\text{obs},t}$ or through an equivalent memory-augmented conditioning matrix.

D.4 Cue Gate Architecture

The Cue Gate is a lightweight classifier that fuses three information streams to estimate the episode-completion score $q_\omega(\mathbf{F}_{\text{vis},t}, \mathbf{G}, \ell) \in [0, 1]$. Its dimensional constants are listed in Table 3.

Visual stream. Backbone features $\mathbf{F}_{\text{vis},t}$ are pooled over spatial and temporal dimensions to produce $\bar{\mathbf{f}}_{\text{vis},t} \in \mathbb{R}^{d_{\text{vis}}}$. The visual branch projects this vector to the gate dimension,

$$\mathbf{u}_{\text{vis}} = \text{Linear}_{d_{\text{vis}} \to d_{\text{gate}}}\big(\text{LayerNorm}(\bar{\mathbf{f}}_{\text{vis},t})\big)\,.$$

Memory stream. Let $\mathbf{g}_k$ be the kth row of $\mathbf{G}$. The memory branch mean-pools the memory tokens and projects them to the same gate dimension,

$$\bar{\mathbf{g}} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{g}_k, \qquad \mathbf{u}_{\text{mem}} = \text{Linear}_{d \to d_{\text{gate}}}(\text{LayerNorm}(\bar{\mathbf{g}}))\,.$$

Language stream. The CLIP language embedding of the task instruction is denoted by $\mathbf{e}_{\text{lang}} \in \mathbb{R}^{d_{\text{lang}}}$. The language branch computes

$$\mathbf{u}_{\text{lang}} = \text{Linear}_{d_{\text{lang}} \to d_{\text{gate}}}(\text{LayerNorm}(\mathbf{e}_{\text{lang}}))\,.$$

Let $\mathbf{u} = [\mathbf{u}_{\text{vis}};\ \mathbf{u}_{\text{mem}};\ \mathbf{u}_{\text{lang}}] \in \mathbb{R}^{3d_{\text{gate}}}$ denote the fused Cue Gate input. The completion score is

$$q_\omega(\mathbf{F}_{\text{vis},t}, \mathbf{G}, \ell) = \sigma\big(\text{Linear}_{d_{\text{gate}} \to 1}\big(\text{GELU}\big(\text{Linear}_{3d_{\text{gate}} \to d_{\text{gate}}}(\text{LayerNorm}(\mathbf{u}))\big)\big)\big)\,. \tag{1}$$

D.5 Reconstruction Decoder

The reconstruction decoder uses L_{recon} cross-attention layers followed by a feed-forward network to predict deltas between consecutive CLIP history embeddings. The decoder sends gradients directly into the Recap Compressor and is used only as an auxiliary training signal. The loss formulation is given in Appendix E.2.

E Training Details

E.1 Stage 1 Video Model Training

The memory-augmented video model is trained with

$$\mathcal{L} = \mathcal{L}_{\text{video}} + \lambda_{\text{recon}}\mathcal{L}_{\text{recon}} + \lambda_{\text{eos}}\mathcal{L}_{\text{eos}},$$

where $\mathcal{L}_{\text{video}}$ is the denoising loss for SVD or the flow-matching loss for Cosmos, $\mathcal{L}_{\text{recon}}$ is the delta reconstruction loss in Appendix E.2, and $\mathcal{L}_{\text{eos}}$ is the Cue Gate binary cross-entropy. Trainable components include the backbone attention layers, the Recap Compressor, memory injection projections, reconstruction decoder, and Cue Gate. The variational autoencoder (VAE) and CLIP text encoder remain frozen. To avoid early token collapse, we first train a memoryless video backbone and then resume from that checkpoint with memory modules added.

E.2 Reconstruction Auxiliary Loss

Given memory tokens $\mathbf{G} \in \mathbb{R}^{K\times d}$ and the $M \le T_{\max}$ valid CLIP history embeddings $\{\mathbf{x}_j\}_{j=0}^{M-1}$ for the current training sample, the reconstruction decoder predicts frame-to-frame feature deltas:

$$\Delta\mathbf{x}_j = \mathbf{x}_j - \mathbf{x}_{j-1}, \qquad \widehat{\Delta\mathbf{x}}_j = f_{\text{recon}}(\mathbf{G},\ \mathbf{x}_{j-1}), \qquad \mathcal{L}_{\text{recon}} = \frac{1}{M-1}\sum_{j=1}^{M-1}\left\|\widehat{\Delta\mathbf{x}}_j - \Delta\mathbf{x}_j\right\|_2^2. \quad (2)$$

The loss is evaluated only for samples with $M > 1$ valid history embeddings.

Predicting deltas $\Delta\mathbf{x}_j$ rather than absolute features $\mathbf{x}_j$ forces the memory tokens to encode what changed between consecutive frames rather than memorizing static scene appearance. This design is inspired by DeltaTok [13], which shows that delta tokenization improves temporal coherence in video representations.

Without reconstruction, the Cue Gate produces noisy, low-confidence predictions that occasionally cross $q_\omega = 0.5$ near the correct frame but cannot produce a sharp, high-confidence completion signal. As reported in Table 4b, overall timing error improves from 44.3 to 35.5 frames at $q_\omega \ge 0.5$. The gain is larger at $q_\omega \ge 0.9$ (62.0 $\rightarrow$ 28.8), confirming that the auxiliary loss improves calibration as well as timing accuracy.

E.3 Cue Gate Supervision

Training uses binary cross-entropy with $y_t = 1$ for the final w_{eos} frames of each episode and $y_t = 0$ otherwise. Positive frames are upsampled by m_{eos} within each batch to compensate for class imbalance. Gradients flow through the memory and language branches, while $\mathbf{F}_{\text{vis},t}$ is detached to prevent the Cue Gate loss from changing backbone representations.

Language conditioning is beneficial for the Cue Gate because the gate must evaluate which goal should be complete. In contrast, language-conditioned Recap queries hurt the task-agnostic history summary in the ablation in Section A.1.

E.4 Stage 2 Action Policy Training

The video backbone remains frozen for stability, but Pathway 1 remains active so the action policy trains on memory-conditioned backbone features. The Recap Compressor, memory injection projections, Cue Gate, Video Former, and GCDenoiser action head continue training with the action-prediction objective. For the VPP instantiation, the action head receives K memory tokens together with N_{den} denoised video tokens, giving $N_{\text{obs}} = K + N_{\text{den}}$ observation tokens.

E.5 Inference

At test time, the current frame is encoded by CLIP and appended to the history cache. The Recap Compressor maps the retained history to $\mathbf{G}$, the video backbone predicts future frames and produces memory-conditioned features, and the action decoder predicts an action chunk from the resulting observation tokens. The Cue Gate score $q_\omega(\mathbf{F}_{\text{vis},t}, \mathbf{G}, \ell)$ terminates the rollout when it exceeds the threshold τ selected for the evaluation setting, or when the rollout reaches the task-specific maximum step count $t_{\max}$.

F Parameter Values

Table 3 lists the concrete values for the symbols used in Sec. 3 and the appendix. Values marked as VPP-specific are used for the SVD UNet implementation; analogous DiT values are determined by the corresponding backbone hidden size. Case-specific choices such as the Cue Gate threshold and maximum rollout length are set by the corresponding evaluation protocol rather than treated as global constants.

G Evaluation Definitions

G.1 Stop Ratio

The *stop ratio* measures whether a generated video physically halts after the task is completed. We compute it as the ratio of post-completion to pre-completion motion energy using the constants in Table 3.

Each generated frame is converted to grayscale at $s_{\text{stop}} \times s_{\text{stop}}$ resolution and denoted by $\mathbf{Y}_t$. Per-frame motion is

$$\delta_t = \text{mean}(|\mathbf{Y}_{t+1} - \mathbf{Y}_t|)\,.$$

Let t^* denote the ground-truth completion frame. We define

$$E_{\text{pre}} = \frac{1}{w_{\text{pre}}} \sum_{u=t^*-w_{\text{pre}}}^{t^*-1} \delta_u, \qquad E_{\text{post}} = \frac{1}{w_{\text{post}}} \sum_{u=t^*}^{t^*+w_{\text{post}}-1} \delta_u, \qquad \text{stop ratio} = \frac{E_{\text{post}}}{E_{\text{pre}} + \epsilon}.$$

A stop ratio below 1 indicates that motion decreases after completion, while a ratio above 1 indicates continued spurious motion. Table 4c reports the corresponding termination results.

Recap Compressor parameters

H Memory Analysis Details

This section provides diagnostic analysis of how the memory tokens encode task progress.

Cue Gate probability over time. We analyze the Cue Gate output q_ω over the course of a $3\times$ repetition task. The curve remains near zero during inter-repetition transitions and crosses the high-confidence threshold $q_\omega > 0.9$ only when the final repetition is completed. During intermediate repetitions, the gate produces small transient bumps in the range 0.1–0.3 that do not trigger termination, demonstrating that the model counts task progress rather than responding to any single placement event.

Decision-critical frame comparison. The third placement in a $3\times$ task and the third placement in a $5\times$ task look nearly identical in raw visual features, with cosine similarity between CLIP frame embeddings exceeding 0.94. Yet the Cue Gate assigns $q_\omega > 0.9$ in the $3\times$ case and $q_\omega < 0.2$ in the $5\times$ case. This confirms that the gate's decision depends on the full history encoded in the memory tokens, including the count of prior repetitions, rather than on the current visual observation alone.

Shuffled memory control. To verify that temporal order matters, we shuffle the CLIP history sequence before passing it through the Recap Compressor at inference time. Shuffling destroys the chronological signal while preserving the same set of visual features. Timing accuracy degrades substantially, confirming that the Recap Compressor exploits temporal ordering to encode task progress and does not rely on bag-of-frames statistics or other spurious cues.

Table 3: **Architecture, training, and evaluation parameter values.**

Group	Symbol	Meaning	Value
Video and policy	r	Recent-frame observation context	4 frames
	H_v	Video prediction chunk length	16 frames
	H_a	Action chunk length	8 actions
	R_{img}	Image resolution	256
History cache	d_{clip}	CLIP visual embedding dimension	512
	$T_{\max}$	Maximum training history embeddings	400
	T_{hist}	Maximum retained episode history	600 frames
Recap Compressor	K	Memory tokens	16 (VPP) / 32 (Cosmos)
	d	Memory hidden dimension	384 / 1024 (Cosmos)
	L	Recap cross-attention layers	4
	N_{head}	Attention heads per Recap layer	8
	d_{head}	Attention head dimension	64
	m_{ffn}	Feed-forward width multiplier	4
	P_{recap}	Recap Compressor parameters	$\sim$4.7M / 32M (Cosmos)
Reconstruction decoder	L_{recon}	Reconstruction cross-attention layers	1
	P_{recon}	Reconstruction decoder parameters	$\sim$1.2M
VPP action decoder	N_{den}	Denoised video tokens	224
	N_{obs}	Action observation tokens	$K + N_{\text{den}}$ (240 for $K = 16$)
Cue Gate	d_{vis}	Pooled backbone feature dimension (VPP)	1280
	d_{lang}	CLIP language embedding dimension	1024
	d_{gate}	Cue Gate branch dimension	256
	w_{eos}	Positive episode-boundary window	10 frames
	m_{eos}	Positive-frame upsampling factor	5
Training	λ_{recon}	Reconstruction loss weight	1.0
	λ_{eos}	Cue Gate loss weight	0.1
	η_{video}	Stage 1 base learning rate	5×10^{-6}
	η_{mem}	Stage 1 memory learning rate	5×10^{-5}
	m_{mem}	Memory learning-rate multiplier	$10\times$
	B_1	Stage 1 effective batch size	32
	S_{base}	Memoryless warm-start checkpoint	270,000 steps
	S_1	Total Stage 1 training	500,000 steps
	σ_{clip}	CLIP embedding noise std.	0.1
	η_{act}	Stage 2 learning rate	10^{-4}
	B_2	Stage 2 batch size	16
	N_{DDIM}	DDIM denoising steps for action decoding	10
Stop ratio	s_{stop}	Grayscale evaluation resolution	64
	w_{pre}	Pre-completion motion window	48 frames
	w_{post}	Post-completion motion window	72 frames
	ϵ	Stop-ratio numerical stabilizer	10^{-8}

I Real-Robot Experimental Setup

Figure 4 shows the physical setup used for the real-robot evaluations in Sec. 4.4. The hardware uses PiPER tabletop manipulation arms in task-dependent configurations, with single-arm control for the single-arm evaluations and a bimanual setup for the bimanual evaluation. A fixed Intel RealSense D415 camera is mounted above the workspace to capture visual observations for policy execution.

J Quality of Video Generation Results

Figure 5 shows the comparison between video generation results between video generation model w/ and w/o memory.

	(a)						(b)						(c)		
	Single H_v			Iter. FVD↓			$\lvert\Delta\rvert$ (frames↓)						Ratio↓		
Method	SSIM↑	LPIPS↓	FVD↓	$2H_v$	$4H_v$	$8H_v$	Var.	Thr.	1×	3×	5×	All	Var.	Ratio	Energy
VPP	0.7711	0.1673	43.36	39.40	33.10	37.81	w/o	.5	**29.5**	24.5	79.1	44.3	No mem	1.11	4.10
Ours (UNet)	**0.8094**	**0.1145**	**18.32**	**17.01**	**17.11**	**19.84**	**w/**	.5	37.2	**14.8**	**54.6**	**35.5**			
Cosmos-RF	0.8744	0.0897	22.54	**23.78**	**21.23**	26.11	w/o	.9	72.3	33.0	80.7	62.0	w/o	0.53	1.67
Ours (DiT)	**0.8752**	**0.0781**	**18.65**	25.37	22.06	**16.24**	**w/**	.9	**21.0**	**14.6**	**50.8**	**28.8**	**w/**	**0.46**	**1.00**

Table 4: **Generation and completion quality** on LIBERO-Mem [4]. (a) video prediction fidelity, (b) completion-timing error in frames, and (c) post-completion motion. w/ and w/o denote with and without the reconstruction auxiliary loss.

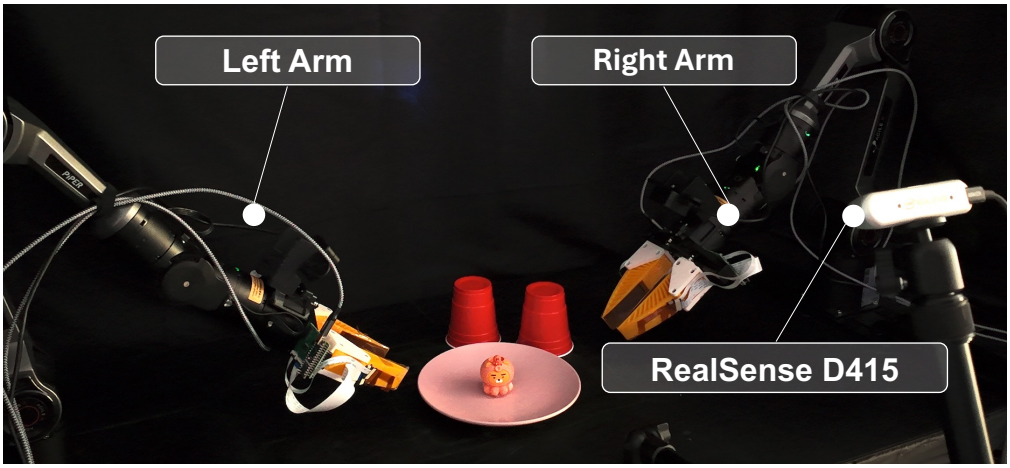

Figure 4: **Real-robot Setup for MemoryVAM Evaluation.** The setup includes PiPER tabletop manipulation arms, the object set used in the real-robot memory tasks, and a fixed Intel RealSense D415 camera for capturing visual observations.

Figure 5: **Comparison of Video Generation Quality**: Video generation results w/ and w/o memory. Upper: Video generation with memory can accurately terminate upon task completion while maintaining stable long-horizon visual consistency. Lower: Video generation without memory exhibits significant temporal fluctuations, progressive degradation over time, and fails to reliably determine when the task has ended.