HIPPOVLA: INTEGRATING SHORT-TERM VISUAL MEMORY INTO VISION-LANGUAGE-ACTION POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) policies conditioned primarily on current observations face temporal ambiguity: similar observations may require different actions depending on prior interactions. We introduce HIPPOVLA, which incorporates short-term visual history through the backbone's existing intermediate visual injection pathways. Its Short-Term Memory Bank (STMB) organizes sparsely sampled historical features into sets corresponding to selected visual levels and camera views, retrieves historical context through cross-attention, and fuses it with current features through learnable gates. The enhanced representations contribute to action prediction without appending historical observation tokens, keeping the language-model input sequence length unchanged. On CALVIN ABC→D, HIPPOVLA improves average completed sequence length from 4.085 to 4.619 and the success rate for completing five consecutive instructions from 71.2% to 83.0% over the no-STMB baseline with the same backbone and training protocol. Across the four LIBERO suites, it achieves a mean success rate of 98.5%. In real-world repeated pick-and-place manipulation, success rates for completing two and three full cycles both improve over the no-STMB baseline by 30 percentage points.

1 INTRODUCTION

Vision-language-action (VLA) models combine pretrained vision-language representations with action learning for language-conditioned robotic manipulation. Representative models, including OpenVLA and the π family, translate multimodal observations and instructions into robot actions (Kim et al., 2024; Black et al., 2024; Physical Intelligence et al., 2025). However, policies conditioned primarily on current observations face an inherent limitation: information required for action selection is not always visible in the current scene. During occlusion, the current observation may not reveal the target's last observed position; a single frame may also leave recent motion direction ambiguous. In repeated manipulation, visually similar scenes can correspond to different stages of task progress, making completed steps difficult to infer from the current observation alone. As illustrated in Figure 1(a), different interaction histories can require different actions despite similar current visual input, potentially leading policies that rely primarily on current observations to make inappropriate action predictions.

Recent visual history can supplement interaction information missing from current observations and provide temporal context for action selection. One approach integrates history through a policy head or memory module after per-frame vision-language representations are formed. RoboFlamingo aggregates these representations with an LSTM policy head, whereas MemoryVLA retrieves historical context from a perceptual–cognitive memory bank and adaptively fuses it with current representations to condition action generation (Li et al., 2024; Shi et al., 2026b). These methods primarily organize historical interactions in policy states or perceptual–cognitive representations, without directly fusing history at multiple intermediate visual levels. Another approach organizes historical observations as additional inputs for joint processing by the vision-language model. The interleaved history configuration studied in RoboVLMs arranges observations and actions into a sequence spanning multiple time steps, enabling the model to process history within a shared context (Li et al., 2026b). However, when the number of tokens per time step is fixed, extending the history window lengthens the input sequence and increases the associated computation and memory costs.

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.

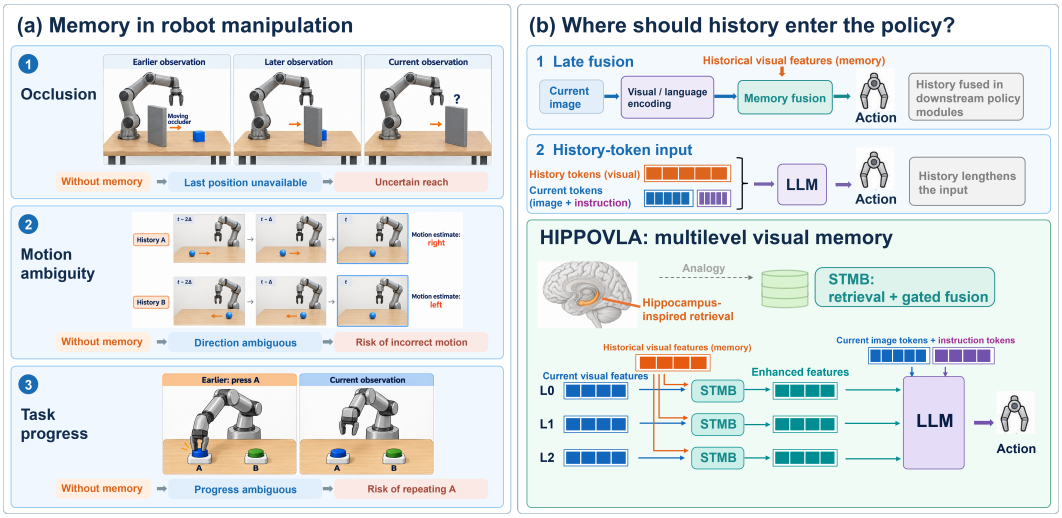


Figure 1: **Motivation and overview of HIPPOVLA.** (a) History-dependent ambiguities arising from occlusion, object motion, and task progress. (b) Comparison of downstream memory fusion, history-token input, and HIPPOVLA's intermediate visual memory integration. HIPPOVLA uses historical features from corresponding visual levels and camera views while preserving the language-model input length. The hippocampal illustration denotes a functional analogy.

To address these issues, we propose **HIPPOVLA**, a VLA method that incorporates short-term visual history through the backbone's existing intermediate visual injection pathways. Inspired by the associative retrieval function of the hippocampus (Moscovitch et al., 2016), we design a Short-Term Memory Bank (STMB) that organizes sparsely sampled recent observations into historical feature sets at corresponding visual levels and camera views. Within each selected level and view, STMB uses features from the most recently sampled historical observation as queries to aggregate historical context through cross-attention, then fuses the retrieved context with current features using a learnable gate conditioned on both. As illustrated in Figure 1(b), the enhanced features enter the language model through its existing visual injection pathways without appending historical observation tokens. This preserves the input sequence length while adding computation for historical visual encoding, retrieval, and fusion.

We evaluate HIPPOVLA in simulation and on a real robot, covering the CALVIN and LIBERO manipulation benchmarks and a real-world repeated pick-and-place task (Mees et al., 2022; Liu et al., 2023). On CALVIN ABC→D, HIPPOVLA achieves an average completed sequence length of 4.619 and a success rate of 83.0% for completing five consecutive instructions, improving over the no-STMB baseline with the same backbone and training protocol by 0.534 and 11.8 percentage points, respectively. Across the four LIBERO suites, the mean success rate reaches 98.5%, exceeding the baseline by 0.7 percentage points. In real-world experiments, success rates for completing two and three full pick-and-place cycles reach 80% and 60%, respectively, both improving over the no-STMB baseline by 30 percentage points.

Our main contributions are:

- **Intermediate visual history integration.** We introduce HIPPOVLA, whose Short-Term Memory Bank (STMB) retrieves historical context from features at corresponding visual levels and camera views and fuses it with current features through learnable gating.
- **Temporal context without extending the language-model input sequence.** HIPPOVLA incorporates historical context through the backbone's existing intermediate visual injection pathways without appending historical tokens, keeping the language-model input length unchanged as the history window expands.
- **Evaluation of memory augmentation benefits and costs.** We evaluate HIPPOVLA on CALVIN, LIBERO, and real-world repeated manipulation tasks, examine history aggrega-

tion, fusion levels, and temporal encoding through ablations, and quantify the additional inference costs of incorporating short-term history.

2 Related Work

Vision-language-action models. Vision-language-action (VLA) models combine pretrained vision-language representations with robot action learning to generate actions from visual observations and natural-language instructions. OpenVLA adapts a pretrained vision-language model into a generalist robot policy and provides an open-source model and training framework (Kim et al., 2024). π_0 combines vision-language representations with a flow-matching action expert to generate continuous action sequences, while $\pi_{0.5}$ extends generalization to open-world environments through co-training on heterogeneous data (Black et al., 2024; Physical Intelligence et al., 2025). For multimodal backbones, DeepStack explores injecting visual representations at different language-model depths, while Qwen3-VL uses multilevel visual features to enhance vision-language interaction (Meng et al., 2024; Bai et al., 2025). However, how to use recent observations to provide temporal context for current visual features at different levels remains an open design question.

History modeling and memory in robot policies. Robot policies incorporate historical context through recurrent states, explicit memory retrieval, and sequence modeling. RoboFlamingo aggregates per-frame vision-language representations with an LSTM policy head, using recurrent states to support action prediction (Li et al., 2024). ReMem-VLA propagates context at different temporal scales through frame-level and chunk-level recurrent queries, with a connector enabling interaction between memory and action queries (Li et al., 2026a). MemoryVLA uses a perceptual–cognitive memory bank to store visual details and high-level semantics, retrieves historical information, and adaptively fuses it with current representations to condition a diffusion action expert (Shi et al., 2026b). These methods integrate historical context through policy states, query representations, or perceptual–cognitive representations, rather than directly fusing history at multiple intermediate visual levels. Another approach provides historical observations as sequential inputs for joint processing by a vision-language model. The interleaved history configuration studied in RoboVLMs organizes observations and actions into a sequence spanning multiple timesteps; with a fixed number of tokens per timestep, extending the history window increases input length, and its reported computational and GPU memory costs exceed those of the policy-head fusion configuration (Li et al., 2026b). In contrast, HIPPOVLA uses STMB to retrieve historical context and fuse it with current features through learnable gating at multiple intermediate visual levels. The enhanced features introduce historical context through the backbone's existing visual injection pathways while preserving the language-model input sequence length.

3 Method

3.1 Problem Formulation and Architecture

We consider language-conditioned robotic manipulation. At timestep t, the policy conditions on a language instruction ℓ, the current multiview observation $o_t = \{I_t^v\}_{v=1}^V$, and a sparsely sampled recent visual history $\mathcal{H}_t$ to predict continuous actions for the next K timesteps:

$$\widehat{A}_t = [\widehat{a}_t, \ldots, \widehat{a}_{t+K-1}] \sim \pi_\theta(\cdot \mid o_t, \ell, \mathcal{H}_t), \qquad \widehat{A}_t \in \mathbb{R}^{K \times d_a}. \tag{1}$$

Here, V is the number of camera views, K is the action prediction horizon, and d_a is the dimension of a single action.

As shown in Figure 2(a), HIPPOVLA introduces a Short-Term Memory Bank (STMB) into a pretrained backbone with multilayer visual injection pathways. A shared vision encoder extracts features from current and historical observations at selected intermediate levels, and the backbone's existing spatial mergers and projections map them to the language-model hidden dimension. At each level, STMB processes features from each camera view separately, enhancing current visual representations through historical retrieval and gated fusion. The enhanced features enter the language model through its existing visual injection pathways and contribute to action prediction together with the current images and language instruction. Finally, hidden representations at action-placeholder

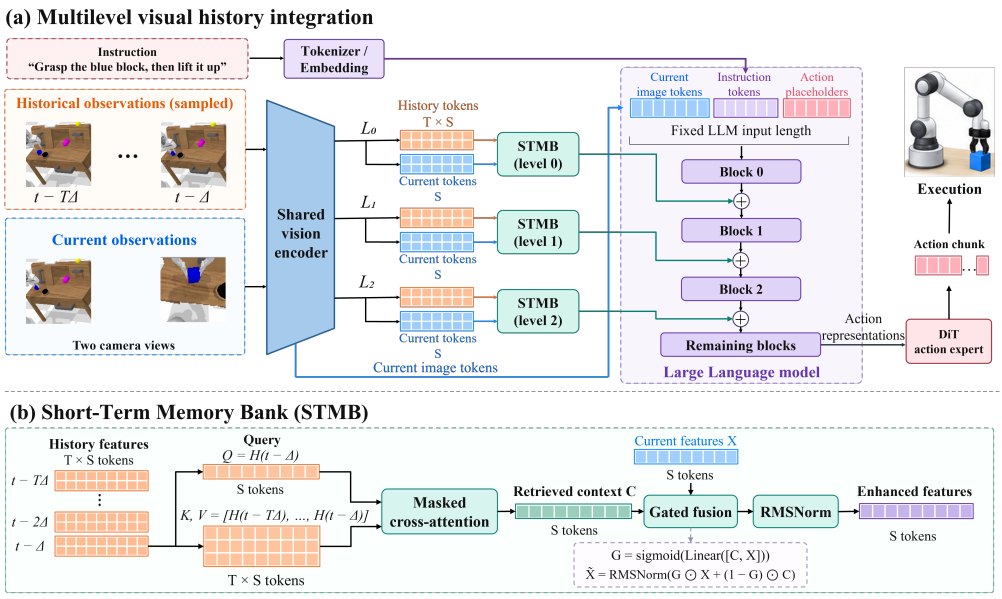


Figure 2: **Overall architecture and short-term memory module of HIPPOVLA.** (a) A shared vision encoder extracts multilevel features from current and sparsely sampled historical observations. STMB retrieves and fuses history through gating within corresponding levels and camera views. Enhanced representations contribute to action prediction through the backbone's existing visual injection pathways without increasing the language-model input length. The historical-observation panel shows only fixed-camera images; the illustrated two-view configuration uses history from both fixed and wrist cameras. (b) STMB uses historical features at the most recent sampled timestep $t - \Delta$ as queries and features from all T historical timesteps as keys and values. Cross-attention with a validity mask aggregates TS historical tokens into S context tokens, which are fused with current features through gating, followed by RMS normalization. S denotes the number of visual tokens per level and camera view.

positions condition a diffusion action expert to generate an action chunk. This process does not append historical tokens to the language-model input sequence.

3.2 Multilevel Visual History Construction

At timestep t, we sample T historical timesteps from recent observations at a fixed interval Δ, forming a chronologically ordered visual history:

$$\tau_j = t - (T - j + 1)\Delta, \qquad \mathcal{H}_t = (o_{\tau_1}, \ldots, o_{\tau_T}), \quad j = 1, \ldots, T. \tag{2}$$

The history window spans $t - T\Delta$ to $t - \Delta$ and excludes the current observation. Taking the start of the current history segment as timestep zero, we define the validity mask

$$m_{t,j} = \mathbf{1}[\tau_j \geq 0]. \tag{3}$$

Unavailable observations before the segment start are represented by black-image placeholders, and their corresponding historical tokens are excluded from subsequent retrieval using this mask.

We extract current and historical features from R selected intermediate levels of the shared vision encoder. Let b_r denote the encoder block corresponding to selected level r, E_{b_r} the visual encoding process up to that block, and P_r the backbone's existing spatial merger and projection. For level r and camera view v, the current and historical features are

$$X_t^{r,v} = P_r(E_{b_r}(I_t^v)), \qquad H_{t,j}^{r,v} = P_r(E_{b_r}(I_{\tau_j}^v)), \tag{4}$$

where $X_t^{r,v}, H_{t,j}^{r,v} \in \mathbb{R}^{S \times D}$, S is the number of spatially merged visual tokens per view, and D is the language-model hidden dimension. Current and historical observations share feature extraction

and projection parameters, yielding feature collections at corresponding levels and views. We retain the backbone's existing spatial position handling without adding explicit temporal encoding. STMB performs historical retrieval and fusion on these extracted intermediate representations without changing attention computation inside the vision encoder.

3.3 Short-Term Memory Retrieval and Gated Fusion

Historical feature retrieval. As shown in Figure 2(b), for each selected visual level and camera view, STMB concatenates the sampled historical features along the token dimension into a history collection and uses the features of the most recent sampled observation, at $t - \Delta$, as the query:

$$M_t^{r,v} = \mathrm{Concat}_{j=1}^{T} H_{t,j}^{r,v} \in \mathbb{R}^{TS \times D}, \qquad U_t^{r,v} = H_{t,T}^{r,v} \in \mathbb{R}^{S \times D}. \tag{5}$$

Here, $U_t^{r,v}$ supplies the query features, while $M_t^{r,v}$ supplies the features for keys and values and includes the most recent sampled observation. The most recent sampled observation provides the visual reference closest to the current timestep within the history window. Using its features as queries, STMB first aggregates context from the history collection, then determines fusion weights through a gate conditioned jointly on the current features and retrieved context. Historical context construction and its influence on current representations are thus handled by the retrieval and gating stages, respectively.

STMB applies separate RMS normalization (Zhang & Sennrich, 2019) to the query features and history collection, then uses linear projections to construct queries, keys, and values for multihead cross-attention (Vaswani et al., 2017). With H_a attention heads and a per-head dimension of $d = D/H_a$, we compute

$$\begin{aligned} Q_h &= \mathrm{RMSNorm}_{q,r}(U_t^{r,v}) W_{q,r,h}, \\ K_h &= \mathrm{RMSNorm}_{kv,r}(M_t^{r,v}) W_{k,r,h}, \\ V_h &= \mathrm{RMSNorm}_{kv,r}(M_t^{r,v}) W_{v,r,h}, \end{aligned} \tag{6}$$

$$C_t^{r,v} = \mathrm{Concat}_{h=1}^{H_a} \left[\mathrm{MaskedSoftmax}\left(\frac{Q_h K_h^\top}{\sqrt{d}}, m_t \right) V_h \right] W_{o,r}. \tag{7}$$

The validity mask m_t is expanded to all tokens of each historical observation, so attention weights are normalized only over valid historical features; the retrieved context is set to zero when no valid history is available. Retrieval aggregates TS historical tokens into S context tokens, yielding $C_t^{r,v} \in \mathbb{R}^{S \times D}$ with the same shape as the current features. Different visual levels use independent retrieval parameters, while camera views at the same level share parameters and perform retrieval separately.

Feature-wise gated fusion. To regulate the influence of historical context on current visual representations, we jointly use the retrieved context and current features to compute a token-wise, channel-wise fusion gate:

$$G_t^{r,v} = \sigma([C_t^{r,v} \| X_t^{r,v}] W_{g,r} + b_{g,r}), \tag{8}$$

$$\widetilde{X}_t^{r,v} = \mathrm{RMSNorm}_{o,r}(G_t^{r,v} \odot X_t^{r,v} + (1 - G_t^{r,v}) \odot C_t^{r,v}). \tag{9}$$

Here, $\|$ denotes concatenation along the channel dimension, σ is the sigmoid function, and $\odot$ denotes element-wise multiplication. The gate $G_t^{r,v} \in (0,1)^{S \times D}$ weights current features and historical context by $G_t^{r,v}$ and $1 - G_t^{r,v}$, respectively, followed by RMS normalization to obtain the enhanced visual representation $\widetilde{X}_t^{r,v}$. This process preserves the number and dimension of visual tokens. When no valid history is available, $C_t^{r,v} = 0$, but the same gating and normalization still apply.

3.4 Visual Injection and Action Learning

Integration into existing visual pathways. As shown in Figure 2(a), final-layer visual representations of the current observations form the language model's image input through the backbone's original interface, while the intermediate representations enhanced by STMB are injected through its existing multilayer visual pathways. Let $Z^{(k)}$ denote the input hidden states of decoder block k, and let $\mathcal{D}_k$ denote its transformation. For a block receiving visual level r, the update is

$$Z^{(k+1)} = \mathcal{D}_k(Z^{(k)}) + \mathcal{S}_{\mathcal{I}}\left(\mathrm{Concat}_{v=1}^{V} \widetilde{X}_t^{r,v} \right), \tag{10}$$

The enhanced features from different views are concatenated in the order of the input image tokens. The operator $\mathcal{S}_{\mathcal{I}}$ maps them to the existing image-token positions $\mathcal{I}$ and fills all other positions with zeros. Enhanced features are added element-wise to the hidden states after the corresponding decoder block and contribute to action representations through subsequent blocks. Decoder blocks without visual injection retain their original computation.

Because historical context enters the language model through feature addition at existing image-token positions, HIPPOVLA does not allocate extra input tokens to historical observations. For a fixed instruction, image resolution, camera-view count, and action prediction horizon, the language-model input length is

$$N_{\mathrm{LM}} = N_{\mathrm{text+special}} + VS + K, \tag{11}$$

where $N_{\mathrm{text+special}}$ counts text and other special tokens, excluding the K action placeholders counted separately. The language-model input length therefore does not increase with the number of sampled historical frames T. Additional computation arises from historical visual encoding and STMB: the current implementation re-encodes sampled historical images at every policy query and computes $S \times TS$ attention matrices within each level and view. Excluding visual encoding and linear projections, attention computation across R levels has complexity $O(RVTS^2D)$.

Action learning and inference. We linearly project the final hidden states at the K action-placeholder positions to condition a DiT action expert (Peebles & Xie, 2023). The model is trained with a diffusion noise-prediction objective (Ho et al., 2020), without an additional language-generation loss. At inference, the action expert generates continuous action chunks through DDIM denoising (Song et al., 2020), conditioned on the current observations, instruction, and sampled visual history. The training objective and inference procedure are detailed in Appendix A.1.

4 Experiments

We evaluate HIPPOVLA in simulation and on a real robot. Our experiments examine (1) sequential language-instruction execution and multi-task manipulation on CALVIN and LIBERO; (2) stage-wise completion rates in repeated real-world manipulation, with and without STMB; (3) the effects of history aggregation, temporal encoding, and fusion levels on task performance; and (4) the additional cost of history encoding and fusion at a fixed language-model input sequence length.

4.1 Experimental Setup

Simulation benchmarks and real-world tasks. We evaluate HIPPOVLA on CALVIN (Mees et al., 2022), LIBERO (Liu et al., 2023), and a real-world repeated manipulation task. CALVIN uses the ABC→D setting: training on environments A, B, and C, and evaluating sequential language-instruction execution in the unseen environment D. LIBERO evaluation covers Spatial, Object, Goal, and Long, testing manipulation across spatial arrangements, objects, task goals, and long-horizon tasks. We additionally evaluate repeated pick-and-place manipulation of a blue block on a real robot, comparing stage-wise completion with and without STMB. Task-specific settings and evaluation protocols are described in the corresponding subsections.

Implementation and training. HIPPOVLA is built on RynnBrain-8B (Dang et al., 2026) with a DiT-B action expert and STMB at three selected visual levels. The main configuration samples five historical observations at intervals of ten environment steps, uses latest-history queries and gated fusion, and omits explicit temporal encoding. Simulation models are trained on eight NVIDIA H100 GPUs with a per-device batch size of four and one gradient accumulation step, giving a global batch size of 32, for 100,000 parameter updates. We use AdamW (Loshchilov & Hutter, 2017) and a cosine learning-rate schedule with warmup. Detailed model and optimization settings are provided in Appendix A.2.

Comparisons. For simulation experiments, we construct a no-STMB baseline with the same backbone and training protocol. It retains the same pretrained initialization, multilevel visual pathways for current observations, and action expert, but excludes historical retrieval and gated fusion. Training budgets and evaluation settings are matched. This comparison assesses the overall contribution

of the STMB memory-augmented configuration. We also include published results of representative methods as benchmark references, presented separately from our experiments. The main HIPPOVLA results use evaluation seed 42. Additional evaluations of the same trained checkpoints with evaluation seeds 42–47 are reported in Appendix A.3.

4.2 Long-Horizon Evaluation on CALVIN

Training and evaluation settings. We follow CALVIN's ABC→D protocol, training on demonstrations from environments A, B, and C and evaluating in environment D. Evaluation comprises 1,000 instruction chains, each containing five language instructions. We report average completed sequence length (Avg.Len) and the success rate for completing at least k consecutive instructions (SRk, $k = 1, \ldots, 5$). Visual history is reset at the start of each language subtask, so the policy uses recent observations within the current subtask to support action prediction.

Table 1: CALVIN ABC→D results. Success rates are percentages; higher is better for all metrics.

Method	Venue / Year	SR1	SR2	SR3	SR4	SR5	Avg.Len
RoboFlamingo (Li et al., 2024)	ICLR 2024	82.4	61.9	46.6	33.1	23.5	2.47
GR-1 (Wu et al., 2024)	ICLR 2024	85.4	71.2	59.6	49.7	40.1	3.06
OpenVLA (Kim et al., 2024)	CoRL 2024	91.3	77.8	62.0	52.1	43.5	3.27
Seer-Large (Tian et al., 2025)	ICLR 2025	96.3	91.6	86.1	80.3	74.0	4.28
FALCON (Zhang et al., 2026c)	ICLR 2026	98.4	94.5	88.6	82.5	75.5	4.40
VLM4VLA (Qwen2.5-VL-7B) (Zhang et al., 2026a)	ICLR 2026	93.5	86.4	80.7	75.8	69.3	4.057
DeFI (multi-view) (Zhang et al., 2026b)	ICLR 2026	97.9	94.2	90.7	87.0	81.2	4.51
UniVLA (Wang et al., 2026)	ICLR 2026	**98.9**	94.8	89.0	82.8	75.1	4.41
MemoryVLA (Shi et al., 2026b)	ICLR 2026	94.8	87.4	81.4	75.9	69.4	4.09
DAWN (DROID) (Nguyen et al., 2026)	CVPR 2026	97.8	91.6	81.3	75.2	64.1	4.10
OptimusVLA (Li et al., 2026c)	CVPR 2026	97.6	93.2	88.8	85.7	78.1	4.45
HiF-VLA (multi-view) (Lin et al., 2026)	CVPR 2026	98.5	94.1	88.1	81.4	73.1	4.35
MemoryVLA++ (Shi et al., 2026a)	arXiv 2026	95.6	90.2	85.7	81.7	76.1	4.29
HIPPOVLA (Without STMB)	—	92.7	86.6	81.4	76.6	71.2	4.085
HIPPOVLA (ours)	—	**98.9**	**97.1**	**94.0**	**88.9**	**83.0**	**4.619**

Results. As shown in Table 1, HIPPOVLA achieves **4.619** Avg.Len and **83.0%** success in completing all five instructions on CALVIN ABC→D. Relative to the no-STMB baseline with the same backbone and training protocol, these improve by **0.534** and **11.8 percentage points**, respectively. HIPPOVLA improves all metrics from SR1 to SR5, showing that the STMB memory-augmented configuration improves sequential language-instruction execution under this protocol.

4.3 Multi-Task Evaluation on LIBERO

Training and evaluation settings. We evaluate HIPPOVLA on LIBERO-Spatial, Object, Goal, and Long, covering spatial relationships, object variation, goal variation, and long-horizon manipulation. One model is jointly trained on demonstrations from all four suites. Each suite contains ten tasks, with 50 evaluation trials per task, totaling 500 trials per suite and 2,000 trials overall. We report success rates for individual suites and their four-suite mean.

Table 2: Success rates on four LIBERO suites (percent; higher is better).

Method	Venue / Year	Spatial	Object	Goal	Long	Mean
OpenVLA (Kim et al., 2024)	CoRL 2024	84.7	88.4	79.2	53.7	76.5
OpenVLA-OFT (Kim et al., 2025)	RSS 2025	97.6	98.4	97.9	94.5	97.1
UniVLA (Wang et al., 2026)	ICLR 2026	95.4	98.8	93.6	94.0	95.5
MemoryVLA (Shi et al., 2026b)	ICLR 2026	98.4	98.4	96.4	93.4	96.7
OptimusVLA (Li et al., 2026c)	CVPR 2026	**99.6**	99.8	**98.4**	96.4	**98.6**
HiF-VLA (multi-view) (Lin et al., 2026)	CVPR 2026	—	—	—	96.4	—
HIPPOVLA (Without STMB)	—	98.0	98.4	98.2	96.6	97.8
HIPPOVLA (ours)	—	98.6	**100.0**	97.6	**97.8**	98.5

Results. As shown in Table 2, HIPPOVLA achieves a mean success rate of **98.5%**. It reaches **100.0%** on Object and **97.8%** on Long, the highest among the listed results, while its overall mean is slightly below OptimusVLA's **98.6%**. Compared with the matched no-STMB baseline, HIPPOVLA improves the mean by **0.7 percentage points**, with gains of **0.6, 1.6, and 1.2 percentage points** on Spatial, Object, and Long, respectively, and a decrease of **0.6 percentage points** on Goal. The improvement on Long is consistent with the motivation for using recent visual history to support multi-stage manipulation, where current observations may not fully reveal task progress.

4.4 Real-World Experiments

Task and evaluation protocol. We evaluate HIPPOVLA on a Piper robot with three camera views in a language-conditioned task requiring three consecutive pick-and-place cycles of a blue cube. HIPPOVLA and the no-STMB baseline are each evaluated under the same set of 10 initial conditions. No manual resets or additional human intervention occur between cycles; a failure or timeout immediately terminates the trial. We report the proportions of all 10 trials that successfully complete at least one, two, and three full cycles. The hardware setup, camera observations, and detailed protocol are provided in Appendix A.4.

Table 3: Success rates on real-world repeated pick-and-place manipulation.

Method	≥ 1 cycle ↑	≥ 2 cycles ↑	3 cycles ↑
No STMB	100% (10/10)	50% (5/10)	30% (3/10)
HIPPOVLA	100% (10/10)	**80%** (8/10)	**60%** (6/10)

Results. As shown in Table 3, both methods complete the first cycle in all trials. HIPPOVLA achieves success rates of 80% and 60% for completing two and three cycles, respectively, exceeding the no-STMB baseline by 30 percentage points in both cases. The improvement is observed in subsequent repeated manipulation stages.

4.5 Ablation Studies

We conduct ablations on CALVIN ABC→D to examine history aggregation, fusion levels, and temporal encoding. We report average completed sequence length (Avg.Len) and the success rate for completing five consecutive instructions (SR5). Other settings are held fixed within each comparison. Detailed variant configurations are provided in Appendix A.5.

History aggregation and fusion levels. We compare cross-attention retrieval with temporal mean aggregation and restrict historical fusion to individual visual levels. All variants retain the backbone's multilevel pathways for current observations and omit explicit temporal encoding. As shown in Table 4, cross-attention retrieval outperforms temporal mean aggregation by 0.047 Avg.Len and 1.0 percentage point in SR5. L0-only fusion performs below the three-level configuration, whereas L1-only and L2-only fusion achieve comparable or higher results. L2-only fusion achieves the highest observed performance, exceeding the default three-level configuration by 0.046 Avg.Len and 2.5 percentage points in SR5. These results indicate that the placement and scope of historical fusion affect task performance, but do not establish an advantage for fusion at all three levels.

Table 4: Ablations of history aggregation and fusion levels.

Configuration	Aggregation	Fusion levels	Avg.Len ↑	SR5 (%) ↑
No STMB	—	—	4.085	71.2
HIPPOVLA	Cross-attention	L0, L1, L2	4.619	83.0
Temporal mean	Temporal mean	L0, L1, L2	4.572	82.0
L0 only	Cross-attention	L0	4.539	81.6
L1 only	Cross-attention	L1	4.631	83.3
L2 only	Cross-attention	L2	**4.665**	**85.5**

Temporal encoding. We compare no explicit temporal encoding, absolute temporal encoding, and relative temporal encoding while keeping three-level fusion, the query source, and gating unchanged. As shown in Table 5, the configuration without explicit temporal encoding performs best, improving Avg.Len by **0.199** and SR5 by **7.2 percentage points** over absolute temporal encoding.

Table 5: Temporal encoding ablation.

Temporal encoding	Avg.Len ↑	SR5 (%) ↑
None (HIPPOVLA)	**4.619**	**83.0**
Absolute	4.420	75.8
Relative	4.316	72.8

4.6 Efficiency Analysis

Measurement settings. We compare HIPPOVLA with a history-token configuration that disables STMB and feeds historical images into the same VLM through its native multi-image interface. Using the same checkpoint and paired observations, we evaluate 3, 5, and 10 historical time steps, each containing two camera views, on a single NVIDIA H100 GPU with batch size one. We report language-model input length, full-model FLOPs, inference latency, and peak allocated GPU memory. FLOPs cover visual encoding, memory integration when enabled, language-model computation, and all action denoising steps. This comparison evaluates computational costs rather than task performance; detailed protocols are provided in Appendix A.6.

Table 6: Inference costs under different history lengths. FLOPs are full-model estimates per prediction of an eight-action chunk. Latency is averaged over three measurement rounds; memory is the maximum peak allocated memory across rounds. Token-count ranges reflect differences in instruction length.

Method	History	LLM tokens	FLOPs (T)	Latency (ms)	Memory (GB)
Current only	0	197–205	3.572	98.62	17.964
History-token input	3	593–601	11.201	119.11	18.267
HIPPOVLA	3	197–205	5.147	133.47	18.577
History-token input	5	857–865	16.391	132.74	18.479
HIPPOVLA	5	197–205	6.162	144.92	18.585
History-token input	10	1517–1525	29.730	194.88	18.979
HIPPOVLA	10	197–205	8.701	177.15	18.700

Results. As shown in Table 6, HIPPOVLA maintains a fixed language-model input length as the history window expands, reducing full-model FLOPs by 54.1%, 62.4%, and 70.7% relative to history-token input at 3, 5, and 10 historical steps, respectively. Total computation nevertheless increases with history length because historical images must still be encoded and processed by STMB. Lower FLOPs do not translate uniformly into lower latency: HIPPOVLA has higher latency and memory usage at 3 and 5 steps, but lower measured values at 10 steps. These results demonstrate reduced computational growth without implying a uniform runtime advantage.

5 Conclusion

We introduce HIPPOVLA, which integrates short-term visual history through a VLA backbone's existing intermediate visual injection pathways. Its hippocampus-inspired STMB retrieves and gates historical context into current visual features without extending the language-model input sequence. Experiments on CALVIN, LIBERO, and real-world repeated pick-and-place tasks demonstrate improvements over the no-STMB baseline, while ablations examine key memory design choices. Efficiency analysis shows lower full-model FLOPs than history-token input, although runtime benefits depend on history length. Future work will explore historical feature caching and adaptive memory selection, investigate how intermediate visual injection locations affect the use of historical context and action prediction, and extend evaluation to longer-horizon robotic tasks.

AI USE STATEMENT

AI tools assisted with identifying relevant literature, drafting sections of the manuscript, and editing the text for clarity and readability. The authors are responsible for verifying the cited sources and all manuscript content, including results and technical claims.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025. URL https://arxiv.org/abs/2511.21631.

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

Ronghao Dang, Jiayan Guo, Bohan Hou, Sicong Leng, Kehan Li, Xin Li, Jiangpin Liu, Yunxuan Mao, Zhikai Wang, Yuqian Yuan, et al. Rynnbrain: Open embodied foundation models. *arXiv preprint arXiv:2602.14979*, 2026. URL https://arxiv.org/abs/2602.14979.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. URL https://arxiv.org/abs/2006.11239.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. URL https://arxiv.org/abs/2406.09246.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025. URL https://arxiv.org/abs/2502.19645.

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. URL https://arxiv.org/abs/2603.12942.

Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, et al. Vision-language foundation models as effective robot imitators. In *International Conference on Learning Representations*, volume 2024, pp. 26703–26721, 2024. URL https://arxiv.org/abs/2311.01378.

Xinghang Li, Peiyan Li, Long Qian, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Xinlong Wang, Di Guo, et al. What matters in building vision–language–action models for generalist robots. *Nature Machine Intelligence*, 8(2):158–172, 2026b. URL https://arxiv.org/abs/2412.14058v4.

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. *arXiv preprint arXiv:2602.20200*, 2026c. URL https://arxiv.org/abs/2602.20200.

Minghui Lin, Pengxiang Ding, Shu Wang, Zifeng Zhuang, Yang Liu, Xinyang Tong, Wenxuan Song, Shangke Lyu, Siteng Huang, and Donglin Wang. Hif-vla: Hindsight, insight and foresight through motion representation for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20732–20742, 2026. URL https://arxiv.org/abs/2512.09928.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023. URL https://arxiv.org/abs/2306.03310.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. URL https://arxiv.org/abs/1711.05101.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022. URL https://arxiv.org/abs/2112.03227.

Lingchen Meng, Jianwei Yang, Rui Tian, Xiyang Dai, Zuxuan Wu, Jianfeng Gao, and Yu-Gang Jiang. Deepstack: Deeply stacking visual tokens is surprisingly simple and effective for lmms. *Advances in Neural Information Processing Systems*, 37:23464–23487, 2024. URL https://arxiv.org/abs/2406.04334.

Morris Moscovitch, Roberto Cabeza, Gordon Winocur, and Lynn Nadel. Episodic memory and beyond: the hippocampus and neocortex in transformation. *Annual review of psychology*, 67(1): 105–134, 2016.

E-Ro Nguyen, Yichi Zhang, Kanchana Ranasinghe, Xiang Li, and Michael S Ryoo. Pixel motion diffusion is what we need for robot control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23663–23672, 2026. URL https://arxiv.org/abs/2509.22652.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023. URL https://arxiv.org/abs/2212.09748.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael 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, 2025. URL https://arxiv.org/abs/2504.16054.

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. URL https://arxiv.org/abs/2606.09827.

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, 2026b. URL https://arxiv.org/abs/2508.19236.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. URL https://arxiv.org/abs/2010.02502.

Yang Tian, Sizhe Yang, Jia Zeng, Ping Wang, Dahua Lin, Hao Dong, and Jiangmiao Pang. Predictive inverse dynamics models are scalable learners for robotic manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 92033–92052, 2025. URL https://arxiv.org/abs/2412.15109.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. URL https://arxiv.org/abs/1706.03762.

Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. Unified vision-language-action model. In *International Conference on Learning Representations*, volume 2026, pp. 80929–80944, 2026. URL https://arxiv.org/abs/2506.19850.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, volume 2024, pp. 10641–10662, 2024. URL https://arxiv.org/abs/2312.13139.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019. URL https://arxiv.org/abs/1910.07467.

Jianke Zhang, Xiaoyu Chen, Qiuyue Wang, Mingsheng Li, Yanjiang Guo, Yucheng Hu, Jiajun Zhang, Shuai Bai, Junyang Lin, and Jianyu Chen. Vlm4vla: Revisiting vision-language-models in vision-language-action models. *arXiv preprint arXiv:2601.03309*, 2026a. URL https://arxiv.org/abs/2601.03309.

Wenyao Zhang, Bozhou Zhang, Zekun Qi, Wenjun Zeng, Xin Jin, and Li Zhang. Disentangled robot learning via separate forward and inverse dynamics pretraining. *arXiv preprint arXiv:2604.16391*, 2026b. URL https://arxiv.org/abs/2604.16391.

Zhengshen Zhang, Hao Li, Yalun Dai, Zhengbang Zhu, Lei Zhou, Chenchen Liu, Dong Wang, Francis EH Tay, Sijin Chen, Ziwei Liu, et al. From spatial to actions: Grounding vision-language-action model in spatial foundation priors. In *The Fourteenth International Conference on Learning Representations*, 2026c. URL https://arxiv.org/abs/2510.17439.

A Appendix

A.1 Action Expert and Training Objective

Action conditioning. At each policy query, we extract the final language-model hidden states at the K action-placeholder positions in sequence order. A linear projection maps these states to conditioning tokens $C_t^a \in \mathbb{R}^{K \times d_c}$ for the DiT action expert (Peebles & Xie, 2023), where d_c is the conditioning dimension. These representations incorporate the current observations, language instruction, and retrieved visual history.

Diffusion training. Given a normalized demonstration action chunk $A_t \in \mathbb{R}^{K \times d_a}$, we sample Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$ and construct noisy actions at diffusion timestep n:

$$A_t^{(n)} = \sqrt{\bar{\alpha}_n} A_t + \sqrt{1 - \bar{\alpha}_n}\epsilon, \tag{12}$$

where $\bar{\alpha}_n$ is the cumulative signal retention coefficient defined by the noise schedule. The action expert is trained with the noise-prediction objective (Ho et al., 2020):

$$\mathcal{L}_{\text{act}} = \mathbb{E}_{A_t, n, \epsilon}\left[\frac{1}{K d_a}\left\|\epsilon_\theta(A_t^{(n)}, n, C_t^a) - \epsilon\right\|_F^2\right]. \tag{13}$$

The DiT action expert embeds the noisy actions as action tokens and adds the diffusion timestep embedding to the conditioning tokens. It concatenates conditioning and action tokens internally, models their interactions with a Transformer, and predicts noise at the action-token positions. HIPPOVLA uses $\mathcal{L}_{\text{act}}$ as its training objective without an additional language-generation loss.

Inference. The action expert initializes an action chunk from Gaussian noise and iteratively denoises it using DDIM (Song et al., 2020), keeping C_t^a fixed throughout the denoising process. The resulting normalized actions are unnormalized and undergo the corresponding action postprocessing before execution. At the next policy query, the model recomputes the conditioning representations using the updated observations and visual history.

A.2 Training Settings

Model configuration. HIPPOVLA initializes its vision-language backbone from RynnBrain-8B (Dang et al., 2026) and uses a DiT-B action expert (Peebles & Xie, 2023). The vision-language backbone, STMB modules, action-conditioning projector, and action expert are jointly optimized without freezing modules or using low-rank adaptation. The DiT-B expert contains 12 Transformer blocks with a hidden dimension of 768 and 12 attention heads. Each prediction contains eight 7-dimensional actions and uses eight action-conditioning tokens.

Visual memory configuration. The three STMB levels, denoted by L0, L1, and L2, correspond to vision-encoder block indices 8, 16, and 24, respectively, using zero-based indexing. Their enhanced representations are injected after language-model decoder blocks 0, 1, and 2 through the backbone's existing visual pathways. The projected visual feature dimension is 4096. At each policy query, the model uses five historical observations sampled at intervals of ten environment steps. Features from the most recent sampled historical observation serve as queries, while all sampled historical features provide keys and values. The main configuration uses gated fusion without explicit temporal encoding.

Optimization. CALVIN and LIBERO models are trained separately for 100,000 updates on eight NVIDIA H100 GPUs using BF16 mixed precision and DeepSpeed ZeRO Stage 2. Each GPU processes four samples per update, with a gradient accumulation factor of one, yielding a global batch size of 32. We use AdamW (Loshchilov & Hutter, 2017) with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, and weight decay 10^{-8}. The peak learning rates are 5×10^{-6} for the vision-language backbone and its STMB modules, 5×10^{-5} for the action expert, and 10^{-5} for the action-conditioning projector. The learning-rate schedule uses 5,000 warmup updates followed by cosine decay to a minimum learning rate of 10^{-6}. The gradient clipping threshold is 1.0. Both main training runs use random seed 42.

Table 7: Training configuration shared by the main CALVIN and LIBERO experiments.

Setting	Value
Pretrained backbone	RynnBrain-8B
Training scope	Full-parameter training
Action expert	DiT-B
Action chunk length	8
Action dimension	7
STMB levels	L0, L1, L2
Vision block indices (zero-based)	8, 16, 24
Decoder injection indices (zero-based)	0, 1, 2
Historical observations	5
History sampling interval	10 environment steps
Retrieval query	Latest sampled history
Fusion	Learnable gating
Explicit temporal encoding	None
Training GPUs	8 NVIDIA H100 GPUs
Numerical precision	BF16 mixed precision
Distributed optimization	DeepSpeed ZeRO Stage 2
Per-device batch size	4
Gradient accumulation factor	1
Global batch size	32
Training updates	100,000
Optimizer	AdamW
Backbone and STMB peak learning rate	5×10^{-6}
Action expert peak learning rate	5×10^{-5}
Action projector peak learning rate	10^{-5}
AdamW betas	$(0.9,\ 0.95)$
AdamW epsilon	10^{-8}
Weight decay	10^{-8}
Warmup updates	5,000
Learning-rate decay	Cosine
Minimum learning rate	10^{-6}
Gradient clipping threshold	1.0
Training seed	42

A.3 Evaluation across Random Seeds

Evaluation protocol. We evaluate HIPPOVLA with six evaluation seeds, 42–47, on CALVIN and LIBERO. For each benchmark, all evaluations use the same checkpoint trained with seed 42, without retraining or changing the model configuration. The history sampling settings and benchmark-specific evaluation protocols remain unchanged across seeds. Each CALVIN evaluation contains 1,000 instruction chains. Each LIBERO evaluation contains 50 trials for each of the ten tasks in each suite, yielding 500 trials per suite and 2,000 trials in total.

The main-text results correspond to evaluation seed 42. Below, we report individual-seed results together with their means and sample standard deviations across the six evaluation seeds. These statistics characterize evaluation variability for fixed trained checkpoints, rather than variability across independent training runs.

CALVIN results. Table 8 reports the results on CALVIN ABC→D. Across the six evaluation seeds, HIPPOVLA achieves an average completed sequence length of 4.607 ± 0.012 and an SR5 of $82.82 \pm 0.57\%$. Avg.Len ranges from 4.589 to 4.623, while SR5 ranges from 82.2% to 83.8%.

LIBERO results. Table 9 reports success rates on the four LIBERO suites. The four-suite mean success rate is $98.32 \pm 0.22\%$ across evaluation seeds, with individual-seed means ranging from 98.00% to 98.60%. Among the four suites, Long exhibits the largest across-seed standard deviation, at 0.83 percentage points.

Table 8: CALVIN ABC→D results across evaluation seeds using the same HIPPOVLA checkpoint trained with seed 42. SR1–SR5 are percentages. The final row reports the mean and sample standard deviation across six evaluation seeds.

Evaluation seed	SR1	SR2	SR3	SR4	SR5	Avg.Len
42	98.9	97.1	94.0	88.9	83.0	4.619
43	98.8	96.7	92.7	88.3	82.4	4.589
44	98.9	96.5	93.3	88.8	82.9	4.604
45	98.9	96.5	93.5	89.1	82.2	4.602
46	99.0	97.0	93.5	89.0	83.8	4.623
47	98.9	97.1	93.5	88.6	82.6	4.607
Mean ± SD	98.90 ± 0.06	96.82 ± 0.29	93.42 ± 0.42	88.78 ± 0.29	82.82 ± 0.57	4.607 ± 0.012

Table 9: LIBERO success rates across evaluation seeds using the same HIPPOVLA checkpoint trained with seed 42. All values are percentages. The final row reports the mean and sample standard deviation across six evaluation seeds.

Evaluation seed	Spatial	Object	Goal	Long	Mean
42	98.6	100.0	97.6	97.8	98.50
43	98.8	99.8	98.4	97.4	98.60
44	98.4	99.6	98.4	96.6	98.25
45	98.2	99.4	98.8	95.6	98.00
46	98.4	99.4	99.0	96.0	98.20
47	98.8	99.8	98.2	96.6	98.35
Mean ± SD	98.53 ± 0.24	99.67 ± 0.24	98.40 ± 0.49	96.67 ± 0.83	98.32 ± 0.22

For each seed, the overall mean is the arithmetic mean of the four suite success rates. The standard deviation in the final column is computed across these seed-level overall means, rather than by averaging the suite-level standard deviations.

A.4 Real-World Experimental Setup and Protocol

Hardware and visual observations. We conduct real-world experiments on a physical Piper robot platform, shown in Figure 3(a). Intel RealSense D435 RGB cameras capture observations at 640×480 resolution. The policy receives three camera views: a front view (`cam_head`), an elevated global view (`cam_high`), and a wrist view mounted on the robot end effector (`cam_wrist`). Figure 3(b) presents example observations from the three views during task execution. The stated resolution refers to image acquisition, rather than the model's processed input resolution.

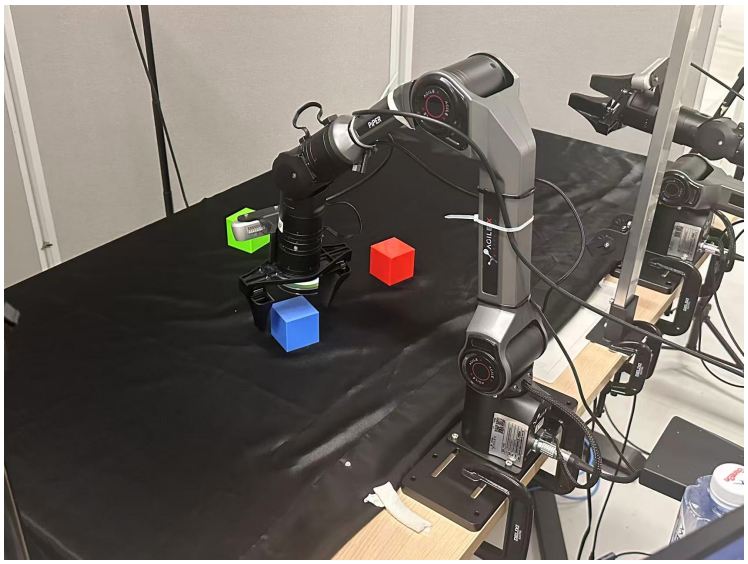
Task instruction. The robot receives the instruction:

> "pick up the blue cube and place it on the target, then repeat the same action sequence two times."

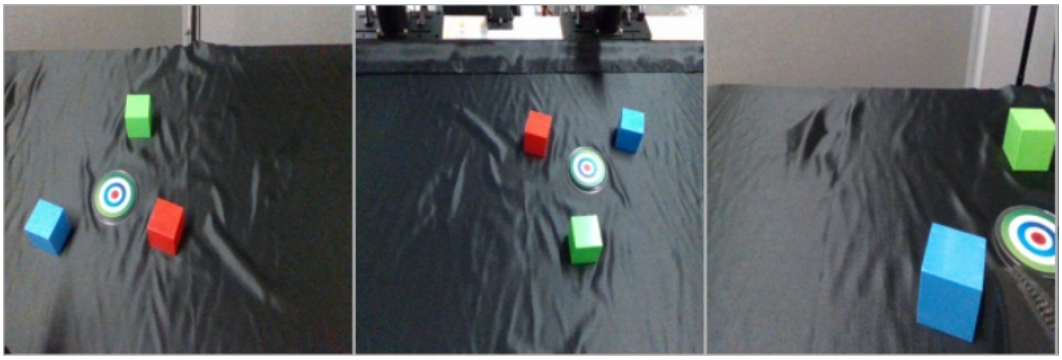
The task comprises three consecutive pick-and-place cycles. Each cycle requires grasping the blue cube and placing it at the designated target location. The complete sequence is executed within a single trial, without manual resets between cycles.

Initial conditions. At the beginning of each trial, the robot arm is placed in a predefined default pose. The cube's initial position is randomly perturbed by $\pm 2\,\mathrm{cm}$ around its nominal position, and its initial orientation is also randomized. HIPPOVLA and the no-STMB baseline are each evaluated under the same set of 10 initial conditions.

Execution and termination. The robot executes all cycles without additional human intervention. A failure or timeout in any cycle immediately terminates the trial. Manual scene resets are performed only after a trial has completed or terminated, to prepare the next initial condition. Only full cycles completed successfully before termination are counted.



(a) Physical robot platform



(b) Three camera observations

Figure 3: **Real-world experimental setup and visual observations.** (a) The Piper robot platform and tabletop workspace. (b) Example observations from the three camera views used by the policy. The panels show different viewpoints rather than successive task stages.

Evaluation metrics. Let $c_i \in \{0, 1, 2, 3\}$ denote the number of consecutive cycles successfully completed in trial i. The cumulative success rate for completing at least k cycles is

$$\mathrm{SR}_{\geq k} = \frac{1}{10} \sum_{i=1}^{10} \mathbf{1}[c_i \geq k], \qquad k \in \{1, 2, 3\}. \tag{14}$$

All three metrics use the same 10 trials as their denominator; they are not conditional success rates computed only over trials reaching a particular cycle.

A.5 Ablation Configurations

Common configuration. The ablations use the CALVIN ABC→D setting. The memory-enabled reference configuration uses five historical observations sampled at intervals of ten environment steps, cross-attention retrieval at L0, L1, and L2, latest-history queries, and gated fusion. It does not use explicit temporal encoding. The experiments use training seed 42 and the optimization settings described in Appendix A.2.

No-STMB baseline. The baseline disables historical inputs and the STMB modules while retaining the pretrained backbone, its multilevel visual pathways for current observations, and the action

expert. This comparison evaluates the overall contribution of the memory-augmented configuration, including its additional parameters and historical inputs.

Temporal mean aggregation. This variant disables cross-attention retrieval and instead averages valid historical features along the temporal dimension. The average is computed independently for each visual level, camera view, and spatial token position. Using the notation from the main text, the aggregated context is

$$C_t^{r,v} = \frac{\sum_{j=1}^{T} m_{t,j} H_{t,j}^{r,v}}{\max\left(1, \sum_{j=1}^{T} m_{t,j}\right)}. \tag{15}$$

The context is zero when no valid history is available. The variant retains gated fusion and output RMS normalization at all three levels, without explicit temporal encoding. It does not use a retrieval query. The cross-attention projections and their input normalization parameters are absent, so this comparison is not parameter-matched to cross-attention retrieval.

Single-level historical fusion. The L0-only, L1-only, and L2-only variants retain retrieval, gating, and output normalization only at the selected level. At each unselected level, the module returns the current visual features unchanged:

$$\widetilde{X}_t^{r,v} = X_t^{r,v}, \qquad r \notin \mathcal{R}_{\mathrm{mem}}, \tag{16}$$

where $\mathcal{R}_{\mathrm{mem}}$ is the set of levels with historical fusion. All backbone pathways for multilevel current perception remain active. The single-level variants therefore change both the placement and number of memory modules, rather than removing the corresponding visual pathways. They use the same history sampling, latest-history queries, and absence of explicit temporal encoding as the reference.

Temporal encoding function. Both temporal-encoding variants use a fixed sinusoidal function without learned temporal embeddings. For an even feature dimension D, define

$$\omega_i = 10000^{-2i/D}, \qquad i = 0, \ldots, D/2 - 1, \tag{17}$$

and

$$e(u) = \left[\sin(u\omega_0), \ldots, \sin(u\omega_{D/2-1}), \cos(u\omega_0), \ldots, \cos(u\omega_{D/2-1})\right]. \tag{18}$$

The implementation concatenates the sine and cosine components rather than interleaving them. The same temporal vector is broadcast across spatial tokens and camera views at each selected level.

Absolute temporal encoding. Let $\tau_j = t - (T - j + 1)\Delta$ denote a sampled historical timestep, measured relative to the start of the current history segment. Absolute encoding modifies the historical and current features as

$$\overline{H}_{t,j}^{r,v} = H_{t,j}^{r,v} + e(\max(\tau_j, 0)), \qquad \overline{X}_t^{r,v} = X_t^{r,v} + e(t). \tag{19}$$

The validity mask still excludes unavailable historical tokens from retrieval. The latest encoded historical features supply the queries, and all encoded historical features supply the keys and values. The encoded current features are used in gated fusion.

Relative temporal encoding. Relative encoding uses the temporal offset from the current observation:

$$\overline{H}_{t,j}^{r,v} = H_{t,j}^{r,v} + e(\tau_j - t), \qquad \overline{X}_t^{r,v} = X_t^{r,v} + e(0). \tag{20}$$

For the five-frame history sampled every ten environment steps, the historical offsets are $-50, -40, -30, -20, -10$. The current observation uses offset zero. Because $e(0)$ contains nonzero cosine components, the current features also receive a temporal embedding. Retrieval and gated fusion otherwise follow the same procedure as the absolute-encoding variant.

Placement of temporal encoding. Temporal embeddings are added to the projected historical and current features before retrieval and fusion. They therefore affect the historical queries, keys and values, as well as the current-feature input to the gate and the subsequent weighted fusion. The no-encoding variant adds no temporal vectors. All three temporal-encoding configurations use the same three fusion levels and query source.

Table 10: Configurations of the reported memory ablations. All memory-enabled variants retain gated fusion. Current-observation pathways are retained in every variant.

Variant	Aggregation	History fusion levels	Temporal encoding
No STMB	—	—	—
HIPPOVLA	Cross-attention	L0, L1, L2	None
Temporal mean	Temporal mean	L0, L1, L2	None
L0 only	Cross-attention	L0	None
L1 only	Cross-attention	L1	None
L2 only	Cross-attention	L2	None
Absolute	Cross-attention	L0, L1, L2	Absolute
Relative	Cross-attention	L0, L1, L2	Relative

A.6 Efficiency Measurement Protocol and Detailed Results

Compared configurations. We compare three inference configurations derived from the same trained HIPPOVLA checkpoint. The *current-only* configuration disables and removes STMB while retaining the backbone's existing visual injection pathways and action expert. The *history-token input* configuration also removes STMB, but supplies historical images through the VLM's native multi-image interface. Historical images are ordered chronologically, followed by the current images, with a consistent camera ordering. The language instruction is included once, and the action placeholders and action-generation procedure are unchanged. *HIPPOVLA* instead retrieves and fuses historical features through STMB without appending historical image tokens to the language-model input. These configurations provide a computational comparison; the current-only and history-token configurations constructed here are not separately trained task-performance baselines.

Paired observations and history construction. We use 20 observations from the CALVIN validation data. Each observation contains a fixed-camera image, a wrist-camera image, and a language instruction. The current observation and instruction are identical across all configurations. We evaluate 3, 5, and 10 historical time steps, sampled at intervals of 10 environment steps. Each historical time step contains both camera views, and shorter history windows are subsets of longer windows.

To obtain complete ten-step histories, historical frames are drawn from the same continuous recording and may cross language-subtask boundaries. This construction is used exclusively for computational profiling, not for evaluating task success. All images are resized with padding to 224×224 before being passed to the processor. The resulting image grids and visual token counts are checked to ensure that all supplied images enter the model. We also verify that the number of action placeholders remains unchanged and that each prediction produces a finite 8×7 action array.

Runtime settings. All measurements use a single NVIDIA H100 80GB GPU, batch size one, and BF16 model precision. HIPPOVLA uses three-level STMB fusion without explicit temporal encoding. The checkpoint is loaded strictly before modifying the runtime history length or removing STMB for the computational controls. The action expert generates an eight-action chunk using five DDIM sampling steps and a classifier-free guidance scale of 1.5. The guidance computation includes both conditional and unconditional branches.

Historical visual features are not cached. All supplied current and historical images are re-encoded on every inference call. Thus, the measurements include the cost of processing the complete history window rather than assuming that historical features are already available.

Latency and memory measurement. Each configuration is measured in a fresh process. We first perform 20 input-validation calls and 60 warmup calls, followed by 100 timed calls obtained by repeating the 20 observations five times. We conduct three measurement rounds with varied configuration order. CUDA synchronization is performed immediately before and after each timed call.

The timing scope includes image resizing, input processing, visual encoding, history integration when enabled, language-model computation, action sampling, and transfer of normalized actions to CPU. It excludes disk access, simulator execution, network communication, and action unnor-

malization. Peak allocated GPU memory is measured after resetting the peak-memory statistics following warmup.

We report mean latency as the mean and sample standard deviation of the three round-level means. P95 latency is computed within each round and then averaged across rounds. Reported memory is the maximum peak allocated memory across the three rounds, expressed in decimal GB ($1\ \mathrm{GB} = 10^9$ bytes). It is not the total device memory reported by `nvidia-smi`. The reported variation describes repeated runtime measurements, not variation across training seeds.

Full-model FLOPs accounting. FLOPs are collected in separate instrumented inference runs so that counting overhead does not affect the latency results. We execute the complete action-prediction procedure for each of the 20 paired observations and average the resulting operation counts.

The accounting covers current and historical visual encoding, STMB retrieval and fusion when enabled, the language-model decoder, the vocabulary projection executed by the current implementation, the action-conditioning projection, and all five action-denoising steps with classifier-free guidance. Although action prediction uses hidden representations, the vocabulary projection is included because it is executed by the evaluated inference code.

We count one multiply-add as two floating-point operations. Matrix multiplications and convolutions are counted from their runtime tensor shapes. External FlashAttention operations are supplemented with explicit attention accounting to avoid omission by the operator counter. Attention matrix products use a dense query–key accounting convention, including for causal attention, rather than attempting to reproduce hardware-specific kernel instruction counts. Bias additions, normalization, elementwise arithmetic, activation functions, and softmax are included using fixed operation-counting rules. Nonlinear operation counts are therefore conventional estimates rather than exact hardware instruction counts. Observed operators are audited, with data movement and non-arithmetic operations excluded explicitly.

Accordingly, the reported values are estimates of *full-model forward FLOPs per action-chunk prediction*. They are not limited to the language-model decoder. CPU image preprocessing, tokenization, random-number generation, data transfer, and I/O are outside this FLOPs accounting; their relevant runtime costs remain reflected in the latency measurement where applicable.

Detailed results. Table 11 reports the complete measurements. HIPPOVLA maintains the same language-model input length for each paired observation as the history window expands. Its total FLOPs nevertheless increase because additional historical images require visual encoding and memory processing. Relative to history-token input, HIPPOVLA reduces full-model FLOPs by 54.1%, 62.4%, and 70.7% at 3, 5, and 10 historical steps, respectively.

These reductions do not imply proportional latency savings. At 3 and 5 historical steps, HIPPOVLA has higher measured latency and peak memory than history-token input. At 10 steps, it has lower mean latency and peak memory under the evaluated settings. The ten-step history-token configuration also exhibits greater round-to-round latency variation. We therefore interpret these measurements as a trade-off between input-sequence growth and memory-processing overhead, rather than evidence of a uniform runtime advantage.

Component-level computation. Table 12 decomposes full-model FLOPs by component. At a given history length, the two history-based configurations have essentially identical visual-encoding costs. HIPPOVLA introduces STMB computation while keeping the decoder and vocabulary-projection costs unchanged. History-token input instead increases computation in those language-model components. The action-conditioning projection and action-sampling costs remain unchanged across configurations.

Table 11: Detailed inference-cost measurements. FLOPs cover the complete model forward computation per eight-action prediction. Latency is reported as the mean ± sample standard deviation of three round-level means. P95 is averaged across rounds, and memory is the maximum peak allocated memory across rounds.

Method	History	FLOPs (T)	Mean latency (ms)	P95 (ms)	Memory (GB)
Current only	0	3.572	98.62 ± 1.02	99.74	17.964
History-token input	3	11.201	119.11 ± 3.80	120.28	18.267
HIPPOVLA	3	5.147	133.47 ± 0.84	134.87	18.577
History-token input	5	16.391	132.74 ± 0.47	133.94	18.479
HIPPOVLA	5	6.162	144.92 ± 1.91	146.46	18.585
History-token input	10	29.730	194.88 ± 11.46	196.41	18.979
HIPPOVLA	10	8.701	177.15 ± 1.70	181.25	18.700

Table 12: Component-level FLOPs in TFLOPs per action-chunk prediction. Vision includes all current and historical images. Vocab. denotes the vocabulary projection. Action includes the conditioning projection and the complete action-sampling procedure. Minor operations outside these components are included in the total; rounded entries may not sum exactly.

Method	History	Vision	STMB	Decoder	Vocab.	Action	Total
Current only	0	0.482	–	2.813	0.250	0.028	3.572
History-token input	3	1.926	–	8.505	0.743	0.028	11.201
HIPPOVLA	3	1.926	0.130	2.813	0.250	0.028	5.147
History-token input	5	2.889	–	12.403	1.071	0.028	16.391
HIPPOVLA	5	2.889	0.183	2.813	0.250	0.028	6.162
History-token input	10	5.297	–	22.513	1.893	0.028	29.730
HIPPOVLA	10	5.297	0.313	2.813	0.250	0.028	8.701