HIPPOSTREAM: HIPPOCAMPAL MEMORY FOR STREAMING VIDEO UNDERSTANDING

Anonymous authors
Paper under double-blind review

ABSTRACT

The hippocampus is associated with the rapid encoding of recent experience into a compact representation that other regions can later draw on. Streaming Video LLMs face the same problem with no comparable mechanism: they either keep a sliding window of recent frames and forget the past, or grow a key–value cache that scales with stream length and breaks on long videos. HIPPOSTREAM is a small parametric token bank that the inference loop rewrites with K functional gradient steps per chunk while the Video LLM trunk stays frozen. The bank plays the role of hippocampus, the frozen trunk the role of neocortex. Its initial state, writer head, and read projection are meta-learned offline; per-chunk cost is independent of stream length, and the bank's footprint stays constant across videos of any length.

On OVO-Bench with Qwen3-VL-8B, HIPPOSTREAM improves the Backward-summary cluster (Action-State Inference, Hold-out reasoning) by $+5.28$ over the recent-window baseline. The gain traces to a single design choice in the WRITE step: anchoring the bank's output positions to each chunk's first vision token preserves the event-specific evidence that backward retrieval needs. Combined with offline meta-learning of the slow weights, this gives a streaming Video LLM reliable access to past events long after the recent window has scrolled by. We release code, meta-trained checkpoints, and the training-data replay buffer.

1 INTRODUCTION

A streaming Video LLM watches a continuous frame stream and must answer questions at any moment about anything it has already seen. Those questions span three families: backward recall ("what colour was the lamp three minutes ago?"), real-time perception ("what is on the table now?"), and forward prediction ("what will happen in the next ten seconds?"). The model cannot rewind, cannot see the future, and cannot grow GPU memory without bound. Today's streaming architectures handle at most one of these families at a time.

Three families of designs dominate the literature, and each cedes the others. Recent-window models keep only the last few seconds of vision tokens and discard every prior event, foreclosing backward recall. Key–value caches retain older content as raw activations, but the cache either grows with stream length and breaks on long videos, or evicts by handcrafted rules (oldest-first, similarity merging) with no learned policy for what to keep. Retrieval-augmented memories store compressed activations in an external buffer that the model reads at decode time He et al. (2024); Zhang et al. (2025a); Song et al. (2024); Di et al. (2025); the buffer still scales with stream length, and the writer is not optimized end-to-end for the question distribution. None of the three places a learned, fixed-size store inside the model's own attention path.

Complementary Learning Systems McClelland et al. (1995) pair a fast hippocampal trace with a slow consolidated store; the two together, rather than either alone, support recall across recent and prospective queries. Streaming Video LLMs already carry the slow side as a pre-trained backbone, but they have no learned fast trace. We use this fast–slow split as design intuition, not as a biological claim.

We propose HIPPOSTREAM, which instantiates that split as a small token bank of $M{=}4$ entries that the inference loop rewrites in place with one inner-loop gradient step per chunk on a frozen Video LLM trunk (Figure 1). The bank's initial state, writer head, and read projection are meta-learned

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.

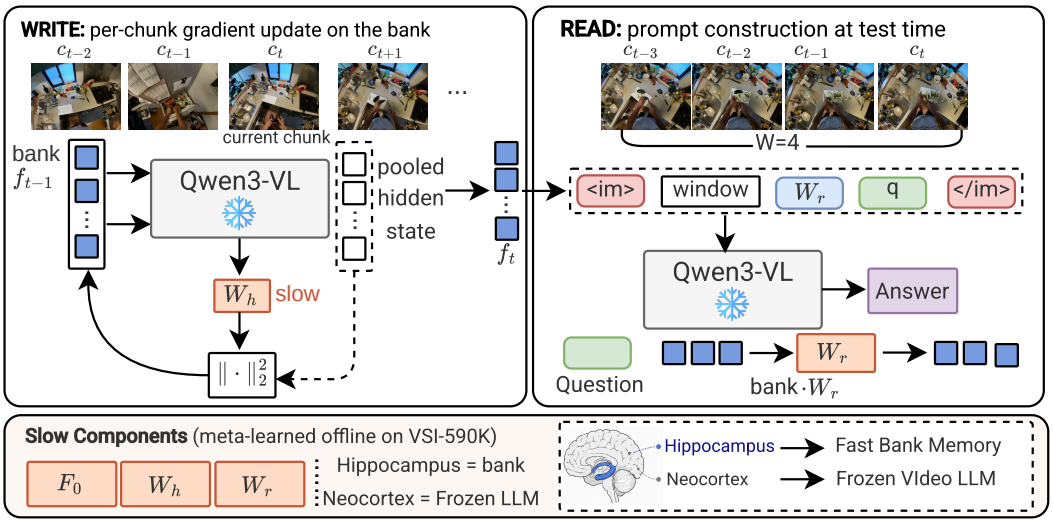

Figure 1: **HIPPOSTREAM architecture.** *Top-left:* at every chunk, the bank takes K functional gradient steps against a target derived from the chunk's vision tokens, updating $M \times d$ floats while the Qwen3-VL trunk stays frozen. *Top-right:* at question time, the post-write bank is projected by W_r and inserted between the recent vision window and the question; the frozen trunk decodes the answer. *Bottom-left:* the slow components M_0 (bank seed), W_h (writer head), and W_r (read projection) are meta-learned offline on VSI-590K and fixed at deployment. *Bottom-right:* the bank plays the role of hippocampus; the frozen trunk plays the role of neocortex.

offline so that each inner-loop write produces a representation the frozen backbone consumes natively. Per-chunk cost is independent of stream length, and the bank itself stays a fixed $M \times d$ across videos of any length.

Section 2 develops the architecture and Section 3 reports the empirical findings on OVO-Bench. The technical contributions of this work are threefold.

1. **A meta-learned, gradient-written memory bank for streaming Video LLMs.** We cast streaming long-term memory as a small set of token-shaped slow weights rewritten by test-time gradients on a frozen Video LLM trunk, an alternative to KV-cache growth, handcrafted eviction, and external retrieval buffers. The bank's initial state, writer head, and read projection are meta-learned offline so the inner-loop write produces a representation the frozen backbone decodes natively.
2. **The WRITE target as the design lever for streaming recall.** We identify the per-chunk WRITE target as the design choice that determines which question family the bank serves. Anchoring the bank's M output positions to each chunk's first vision token preserves event-specific evidence and lifts the Backward-summary cluster (Action-State Inference, Hold-out reasoning) by $+5.28$ points on Qwen3-VL-8B over the recent-window baseline (± 1.07 across three chunk-order shuffle seeds), where a mean-pooled target smooths the same signal away.
3. **Attention bandwidth as the binding constraint on bank size.** The bank's effective capacity is bounded not by parameter count but by attention probability mass at the question: memory tokens compete with recent vision tokens for that mass, and $M{=}4$ resolves the trade-off across our sweeps on both the 2B and 8B backbones.

2 HIPPOSTREAM

HIPPOSTREAM pairs a frozen Video LLM (Qwen3-VL-2B and Qwen3-VL-8B in our experiments) with a small streaming memory the LLM can read at any time. The memory has two halves. A *fast bank* of $M{=}4$ tokens rewrites itself once per second of video through a short inner-loop optimization

step (Section 2.2). We meta-learn a *slow triple* offline and freeze it at deployment: an initial bank state M_0, a writer head W_h, and a read projection W_r (Sections 2.3 and 2.4). The LLM backbone never updates. The fast–slow split mirrors the complementary-learning-systems account McClelland et al. (1995), applied here at the timescale of a single video; we use the analogy as design intuition, not as a biological claim. Figure 1 shows the architecture.

2.1 Streaming Setup and Notation

A video arrives as an ordered stream of one-second chunks $\{c_1, c_2, \ldots, c_T\}$ at native frame rate. We sample one frame per chunk and pass it through Qwen3-VL's vision encoder and patch merger. The result is a post-merger embedding $e_t \in \mathbb{R}^{n_t \times d}$, where d is the LLM hidden size and n_t is the merged-token count for chunk c_t. At any chunk index t, the model can receive a question q and must answer from the past $\{c_1, \ldots, c_t\}$. We give the LLM a recent window of $W{=}4$ chunks at read time. This matches the recent-window baseline of Cambrian-S Yang et al. (2026), so any gain over that baseline attaches to the bank rather than to a wider visual context.

2.2 Fast Bank

A learnable bank $M \in \mathbb{R}^{M \times d}$ (throughout, M denotes the memory bank; the initial state $M_0 = M_0$ is the meta-learned seed) holds the per-video episodic state, with $M{=}4$. The bank starts each video at the meta-learned seed M_0. An inner-loop optimization step rewrites it once per chunk; backbone parameters stay frozen throughout, and only the bank's own values move.

WRITE step. We treat M as a free variable inside the frozen LLM and nudge its values until the LLM's output on the bank positions matches a summary of the chunk positions, so the bank absorbs the chunk. For chunk c_t we take K gradient steps on the bank:

$$\begin{aligned} M_t^{(0)} &= M_{t-1}, \\ M_t^{(k)} &= M_t^{(k-1)} - \eta \, \nabla_{M_t^{(k-1)}} \mathcal{L}_{\text{write}}\big(M_t^{(k-1)}, e_t\big), \quad k = 1, \ldots, K, \\ M_t &= M_t^{(K)}. \end{aligned} \tag{1}$$

The write loss compares the LM's first M output positions, projected through the writer head, to the mean-pooled hidden state of the chunk positions, with the target detached from the gradient graph:

$$\mathcal{L}_{\text{write}}(M, e) = \Big\| W_h \cdot \texttt{LM}([M; e])_{\le M} \; - \; \texttt{sg}\Big[\overline{\texttt{LM}([M; e])_{>M}} \Big] \Big\|_2^2. \tag{2}$$

Here $\texttt{LM}(\cdot)$ is one forward pass of the frozen LLM on the concatenation of bank and chunk tokens, and $\texttt{sg}[\cdot]$ is the stop-gradient operator: the write target is treated as a constant so no gradient flows into the frozen backbone. The gradient flows into M only, so the WRITE step costs one forward and one backward pass with no optimizer state. We use $K{=}1$ and $\eta{=}0.05$ throughout, which collapses Equation 1 to a single update $M_t = M_{t-1} - \eta \nabla \mathcal{L}_{\text{write}}(M_{t-1}, e_t)$; we report the indexed form for generality.

Choice of write target. Equation 2 matches the bank to the chunk's mean-pooled hidden state. We also study a first-position variant that anchors all M memory slots to the chunk's first vision token, $\mathcal{L}^{\text{first}}_{\text{write}}(M, e) = \big\| W_h \cdot \texttt{LM}([M; e])_{\le M} - \texttt{sg}[\texttt{LM}([M; e])_{M+1}] \big\|_2^2$. The mean-pooled target is smoother but discards within-chunk temporal structure; the first-position target keeps a chunk-specific anchor at the cost of higher per-chunk variance. Section 3.2 reports the resulting trade-off across the three OVO-Bench question families.

READ step. At question time, we project the bank by $W_r \in \mathbb{R}^{d \times d}$ and splice it between the recent vision window and the question:

$$[\,\langle\text{im}\rangle \; e_{t-W+1} \ldots e_t \;\; M \cdot W_r \;\; q \;\; \langle/\text{im}\rangle\,]. \tag{3}$$

Qwen3-VL's grid embedding requires one image block per frame. We tile the recent vision tokens into W per-frame blocks and insert the projected bank between the last vision block and the question. The LLM then decodes the answer autoregressively. READ is a standard forward pass; no parameter update happens at inference.

Algorithm 1 First-order meta-training for M_0, W_h, W_r.

Require: Training videos $\{v_i\}$, frozen backbone Φ, outer LR α, aux weight λ
Initialize M_0, W_h, W_r small random
for outer step $t = 1, \ldots, T_{\text{meta}}$ **do**
  Sample $v_i = (\{c_j\}, q, a)$
  $M \leftarrow M_0$
  for each chunk c_j in v_i **do**
    $M \leftarrow$ functional WRITE step on M via Eq. 1 {inner loop}
  end for
  $\mathcal{L}_{\text{outer}} \leftarrow \text{CE}(a_1 \mid M \cdot W_r,\, e_{\text{recent}},\, q)$
  $\mathcal{L}_{\text{aux}} \leftarrow \mathcal{L}_{\text{write}}(M_0, e_{|v_i|})$ {clean-bank write loss}
  Update M_0, W_h, W_r by $\alpha\nabla\big(\mathcal{L}_{\text{outer}} + \lambda\mathcal{L}_{\text{aux}}\big)$
end for
return M_0, W_h, W_r

2.3 Slow Meta-Learned Weights

Three small weight matrices play the slow side of the architecture, and the inference loop never updates them. The initial bank state $M_0 \in \mathbb{R}^{M\times d}$ seeds every video. A writer head $W_h \in \mathbb{R}^{d\times d}$ projects bank hidden states into the WRITE-loss target space (Equation 2). At READ time, a projection $W_r \in \mathbb{R}^{d\times d}$ places the post-write bank where the LLM expects it (Equation 3). For Qwen3-VL-2B (d=2048) the triple totals roughly 8.4M parameters against a frozen 2B backbone; for Qwen3-VL-8B (d=3584) it totals roughly 25.7M against the 8B backbone. Both are well under 1.5% of the backbone parameter count.

2.4 Meta-Learning the Slow Weights

Because the WRITE step routes gradients only into bank values, the bank's eventual usefulness at READ time rests on the three slow weights M_0, W_h, W_r. We fit them jointly with a first-order MAML schedule Nichol et al. (2018); Finn et al. (2017).

Each outer step samples one training video and runs Equation 1 across all of its chunks (the inner loop). The outer loss is the cross-entropy of the answer token under the post-write bank. The outer gradient updates M_0, W_h, W_r only; the backbone stays frozen. Algorithm 1 states the procedure.

First-order MAML breaks the inner-loop chain at the WRITE/READ boundary, so the writer head W_h would otherwise receive no direct outer gradient. We restore that signal with an auxiliary write loss on the clean bank M_0, weighted by λ=2.

Why meta-learning matters. A random M_0 shifts the LLM's outputs measurably, but the shift is uncorrelated with task accuracy on most splits (Section 3). Without a meta-trained writer, the four bank positions read as noise to the LLM. Meta-training addresses both gaps: the writer learns to encode information the LLM can use when the question asks about earlier moments, and W_r learns to place the bank where the LLM already attends. We sweep three meta-training depths (20, 30, and 40 outer steps); each run finishes in under an hour on a single H100.

2.5 End-to-End Training and Inference

Training. We run Algorithm 1 for T_{meta} steps over a fixed pool of 80 videos drawn from VSI-590K Yang et al. (2026), the spatial-reasoning instruction-tuning corpus released by the Cambrian-S team. The output is the slow triple (M_0, W_h, W_r). VSI-590K is held out from OVO-Bench at both the sample-ID and source-corpus levels; OVO-Bench appears only at evaluation time.

Inference. Given a video and a question, the frozen backbone encodes each chunk's vision tokens. The bank starts at M_0 and K inner-loop gradient steps update it per chunk via Equation 1. We then project the post-write bank by W_r, splice it between the recent-window vision tokens and the question, and let the LLM decode the answer.

3 EXPERIMENTS

3.1 SETUP

We evaluate on the full 1640-sample OVO-Bench Niu et al. (2025), spanning three task families: *backward* tracing, *real-time* perception, and *forward* reasoning (multi-step queries with several test instances per video). We use the official `score_ovo_bench` scorer, report per-task accuracies, and average across tasks within each family.

Model and configuration. The frozen backbone is Qwen3-VL-2B-Instruct Bai et al. (2025a) or Qwen3-VL-8B-Instruct, with FlashAttention-2. The recent window holds $W{=}4$ chunks at one frame per second; the bank holds M memory tokens; the slow prefix holds $N{=}8$ tokens (reported only where used). The fast WRITE step runs $K{=}1$ functional gradient step per chunk at learning rate 0.05. All meta-trainable parameters total $\sim$8.4M, against a frozen $\sim$2B backbone (the 8B configuration runs with random-initialised slow components; see Appendix S).

Training schedule and data splits. Meta-training (Algorithm 1) and slow-prefix training run on an 80-video subsample of VSI-590K Yang et al. (2026), the spatial-reasoning instruction-tuning corpus released by the Cambrian-S team. The subsample is filtered to single-letter A/B/C/D MCQ samples that match our answer head and drawn with seed 42. VSI-590K is disjoint from OVO-Bench at both the source-corpus and sample-ID levels: the two share zero source corpora, and the OVO-Bench evaluation set is the only OVO data the pipeline touches. Meta-training runs 30 outer steps and finishes in $\sim$ 45 min on a single H200; replay generation and slow-prefix training add another $\sim$25 min. Appendix A gives the full protocol, including corpus-filter details, sampling seeds, save schedule, and the earlier preliminary configuration we retain in the appendix tables for context.

Evaluation protocol. Headline numbers are reported on the full 1640-sample OVO-Bench on H100-80GB with FlashAttention 2 (631 Bwd, 837 RT, 172 Fwd queries). The $M{=}4$ row reported in the abstract is multi-seeded over three shuffle seeds (42, 137, 7); other rows are single-seed unless noted. A preliminary 10/10/4 matched-subset exploration that identified the design choices we now test at full N is reported in Appendix A (Table 6).

3.2 MAIN RESULTS: PER-FAMILY DECOMPOSITION ON THE FULL BENCHMARK

We ask three questions on the full benchmark. (a) How much of OVO-Bench does the backbone alone close when scaled from 2B to 8B? (b) Which task family does the bank serve under the two WRITE targets we study (mean-pooled vs. first-position)? (c) Do backbone scaling and the gradient bank stack or compete?

Per-task lift on the headline configuration. Figure 2(a) visualises the per-task lift of the 8B + memory $M{=}4$ configuration over the 8B baseline, organised by question family; Table 1 gives the precise numbers. The Bwd-summary cluster (ASI, HLD) gains $+5.28{\pm}1.07$ across three shuffle seeds, the multi-seed value reported in the abstract. The Real-time family loses -1.65 on average, with the largest hits on Spatial-Temporal Understanding (-6.74) and Frame Position Detection (-4.95), the subtypes whose answer is in the visible recent window. The benchmark-wide All accuracy gains $+0.18$, within seed noise. The architecture's effect is therefore not "memory helps everything" but "memory redistributes representational capacity from RT-detection toward Bwd-summary"; the All accuracy averages the two.

Backbone scaling sets the headroom. Table 2 reports the full benchmark across backbones and configurations. The 2B$\rightarrow$8B step gives $+5.87$ Bwd, $+5.61$ RT, and $+0.06$ Fwd at $N{=}1640$, for $+5.74$ on (B+R)/2 and $+3.84$ All. The lift concentrates on the two families that benefit from a stronger frozen trunk; Forward is indifferent. Scaling is the largest single source of accuracy on this benchmark, and the bank's contribution is reported on top of it.

Bank capacity is an attention-bandwidth trade-off. The default $M{=}8$ memory tokens regress the 2B baseline by -1.26 All. Reducing the bank to $M{=}4$ recovers most of the loss (-0.84 All) and lifts Backward; $M{=}2$ falls between (-0.97). On 8B, the same pattern repeats, and the $M{=}4$

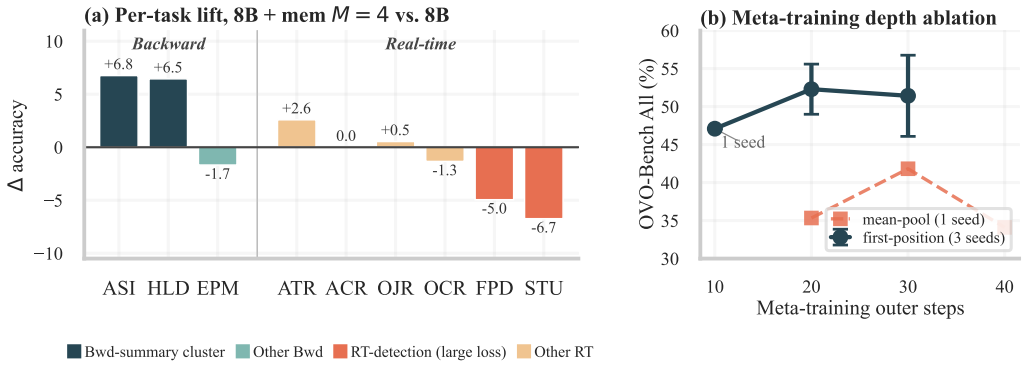

Figure 2: **Main results. (a)** Per-task lift of 8B + memory $M{=}4$ vs. 8B baseline on full 1640-sample OVO-Bench, organised by question family. The Bwd-summary cluster (ASI, HLD; dark teal) is the architecture's intended target; the RT-detection subtypes that share an answer space with the recent vision window (STU, FPD; coral) lose the most. **(b)** Meta-training depth ablation: solid teal is the first-position target on 50/50/50 (3 shuffle seeds, mean ± std); dashed coral is the mean-pool target on 10/10/4 (single seed). Both curves peak near one full pass over the 80 training videos.

Table 1: Per-task lift on full 1640-sample OVO-Bench, 8B + memory $M{=}4$ vs. 8B baseline. Bolded cluster row is the multi-seed headline (seeds 42/137/7). Full per-task table in Appendix M.

Sub-task	8B base	+ mem	Δ
Backward			
ASI	56.76	63.51	**+6.76**
HLD	34.95	41.40	**+6.45**
EPM	54.21	52.53	−1.68
Bwd-summary ASI+HLD (3 seeds)	—	**51.13±1.07**	**+5.28±1.07**
Real-time (RT cost concentrates on current-vision subtypes)			
STU	77.53	70.79	−6.74
FPD	76.24	71.29	−4.95
OCR	95.97	94.63	−1.34
other RT (ATR, ACR, OJR)	82.78	83.82	+1.04
RT mean	83.01	81.36	−1.65
Overall (sample-weighted)	**71.55**	**71.73**	+0.18

row is the single configuration that crosses the baseline (+0.20 All). The mechanism is attention-bandwidth competition: M memory tokens absorb roughly $M/(M+W\cdot V_{\text{recent}})$ of the question-time attention budget against the recent-window vision tokens. At $M{=}8$ that competition costs more on RT-detection than the bank earns on Bwd-summary; smaller M tightens the trade-off without erasing the bank.

Two failure mechanisms in the per-subtask decomposition. Looking inside the families (Table 1) reveals that the −1.65 RT cost and the −1.68 EPM cost in Table 1 arise from *different* architectural mechanisms, not the same one applied twice. The RT cost concentrates on STU (−6.74), FPD (−4.95), and OCR (−1.34): subtypes whose answer is in the recent vision window and that need precise current-frame attention. This is consistent with the bandwidth-competition account above. The EPM cost has a different mechanism. EPM asks about specific past events ("where was the litter bin *before* I picked it up"), and the bank's $M{=}4$ compression discards the frame-specific detail required to answer. The model already does ∼30 percentage points above multiple-choice chance on EPM via language priors and the recent window alone (54.21); the bank's coarse summary neither adds detail nor contradicts the priors, so on those samples it injects noise rather than evidence. We therefore characterise HippoStream's +0.18-aggregate signal as a sum: a +5.28 Bwd-summary gain offset by two distinct costs, *compression loss* on retrieval-precision subtypes (EPM) and *attention-bandwidth competition* on current-vision subtypes (STU, FPD, OCR). The dual decomposition clarifies what

Table 2: Full-1640 readout on Qwen3-VL backbones, H100-80GB, FlashAttention 2. The bank capacity is varied through M; curvature-gated WRITE fires only when the chunk-pooled trajectory's discrete second-difference angle exceeds a $K-\sigma$ running threshold. *Best All within each backbone block in bold.*

Run	Bwd	RT	Fwd	(B+R)/2	All	Δ All
Qwen3-VL-2B						
baseline (no memory)	43.26	77.78	42.12	60.52	**54.39**	—
+ mem M=8	42.95	73.95	42.50	58.45	53.13	−1.26
+ mem M=4	44.06	74.55	42.05	59.30	53.55	−0.84
Qwen3-VL-8B						
baseline (no memory)	49.13	83.39	42.18	66.26	58.23	—
+ mem M=8, no gate	49.92	80.76	39.89	65.34	56.86	−1.37
+ mem M=4, no gate	**51.82**	81.60	41.86	**66.71**	**58.43**	**+0.20**

Full ablation grid including $M\in\{2, 6\}$ and curvature-gated WRITE ($K-\sigma\in\{1.0, 2.0\}$) at 2B and 8B is in Appendix N.

would fix the architecture: increasing M or hybridising with a sparse retrieval store addresses the first; query-conditional attention to the bank addresses the second.

3.3 Ablations

Meta-training depth follows an inverted-U. Figure 2(b) plots OVO-Bench All accuracy as a function of meta-training outer steps under the two WRITE targets. The mean-pool target on the 10/10/4 preliminary subset traces an inverted-U with a peak at step 30; pushing past one full pass over the 80 training videos hurts OVO transfer. Under the first-position target on the larger 50/50/50 subset the same pattern replicates at multi-seed: step 10 (single seed) underfits at 47.09; step 20 peaks at 52.29 ± 3.30; the final checkpoint plateaus within std at 51.42 ± 5.34 across three seeds. We tentatively attribute the overshoot to the high variance of single-video first-order MAML steps.

Matched-pair slow prefix. The slow prefix is the same parameter shape and was trained for the same number of replay-buffer steps in two configurations. Trained against a *random* M_0, the prefix gives $+2.47$ All over the 10/10/4 baseline; trained against the meta-learned M_0 used at inference, it gives $+6.48$. The mismatched prefix specifically hurts forward (29.63 to 18.52). The slow prefix learns a representation calibrated to the fast memory it sits beside and does not transfer when the fast memory's distribution changes. At 50/50/50 the direction replicates qualitatively, though the prefix's net lift falls within seed-noise; we therefore report the main 50/50/50 results without the prefix attached.

Read position is not interchangeable. Swapping the bank's READ position from *before the question* (the meta-trained default) to *before the vision tokens* collapses All-task average by -8.95 on the 10/10/4 preliminary subset and drops backward by nearly forty points. The read projection W_r is calibrated to a position the language model expects mem to occupy (between vision and question, where the chat-template structure positions out-of-distribution tokens), and moving the bank violates that calibration. The negative result falsifies the intuition that mem injection is position-invariant.

Event-gated WRITE is backbone-dependent. A second extension fires the WRITE step only on chunks that cross an event boundary, following training-free token-routing schemes (Yang et al., 2026; Wang et al., 2026, surprise- and curvature-driven respectively). We tested three gates: cosine surprise, curvature, and the bank's own write loss on the new chunk computed without an update (the prediction-error formulation, which matches McClelland's account). Each gate uses a $K-\sigma$ threshold ($\mu + k\sigma$) on the running history. Table 3 reports 50/50/50 numbers for cosine and curvature on Qwen3-VL-2B and -8B paired with the first-position target. At 2B every gate value tested loses -2 to -4 All; at 8B the curvature gate at $K-\sigma=1.0$ recovers the M=8 regression. The pattern is a backbone-dependent inductive bias: smaller backbones cannot afford to skip writes (each refused write leaves the bank with state that has not seen the chunk's content); larger backbones tolerate

Table 3: Event-gated WRITE on top of the first-position target. $K-\sigma$ is the threshold on running novelty. Rows marked "−" do not gate. All numbers on the seed-42 50/50/50 subset unless noted.

Backbone	Gate	$K-\sigma$	All	Δ vs. un-gated
2B	− (un-gated)	−	52.08	—
2B	cosine surprise	0.5	49.97	−2.11
2B	curvature	0.5	49.02	−3.06
2B	curvature	1.0	47.69	−4.39
2B	curvature	2.0	49.44	−2.64
8B	− (un-gated)	−	58.60	—
8B	curvature	1.0	59.34	+0.74

Table 4: Cross-benchmark and cross-backbone results, frozen backbones, no meta-training. OVO-Bench is scored under the same (Bwd+RT)/2 metric as Table 2. Baselines: our recent-window at $W{=}4$ and $W{=}64$ (the LatentStream-protocol configuration), and a bounded-store SigLIP-retrieval reader that keeps 128 novelty keyframes and retrieves 12 for the question. LatentStream numbers are taken from their paper. *Best matched-backbone frozen number in bold.*

Benchmark	Backbone	Recent-4	Recent-64	Bounded store + retrieval	LatentStream (Qu et al., 2026)
OVO-Bench (Bwd+RT)/2	Qwen2.5-VL-3B	**56.8**	55.9	53.4	59.0 (from 52.2)
	Qwen2.5-VL-7B	60.1	**60.7**	58.9	64.2 (from 54.0)
StreamingBench	Qwen2.5-VL-3B	**73.4**	—	72.9	—
	Qwen2.5-VL-7B	**80.5**[†]	—	—	76.9 (from 73.9)

[†] StreamingBench-7B recent-4 is measured on 449 of 499 real-time videos (the other 50 were unavailable at evaluation time); the matched subset of a comparable Qwen3-VL-8B run is +3 points easier than the full set, so 80.5 on the subset corresponds to ≈ 77.6 full-set-equivalent.

selectivity. We therefore report the un-gated configuration as the headline at both backbones in Table 2.

Task-conditional routing recovers RT. The bandwidth competition can be sidestepped at inference: routing the bank only to Backward subtypes (EPM, ASI, HLD) and taking the no-memory path on RT and Forward lifts All to **59.44** (+0.92 over the 8B baseline), and with a denser-frame schedule (fps=2) on top reaches **59.63** (+1.11). This supervised gate uses OVO subtype labels at inference and serves as an oracle ceiling for learned routers; a label-free dual-pass proxy that gates on first-token softmax confidence recovers RT toward baseline on a partial run (Section F).

3.4 Cross-benchmark and cross-backbone validation

The paper's main table (Table 2) uses one benchmark (OVO-Bench) and one backbone family (Qwen3-VL). To test whether the findings hold outside that box, we re-run the pipeline on two other Qwen2.5-VL (Bai et al., 2025b) sizes and two additional streaming benchmarks under a matched protocol (frozen backbone, MAX_PIXELS=100352 ≈128 tokens/frame, recent window $W{=}4$ unless noted). All rows below use *no* meta-training so they measure what the frozen backbones do with the memory contract alone; the meta-learned lift from Table 2 would sit on top.

Three observations. **(a)** LatentStream's reported gains (3B +6.8, 7B +10.2) sit against a baseline 5–6 points below the recent-4 we measure on the same backbones (3B 52.2 vs. our 56.8; 7B 54.0 vs. our 60.1), so the real lift over a fair frozen reader is +3–4. **(b)** Bounded-store retrieval, the +18 headline for VSI-SUPER-Recall (Section 3.5), is net-negative on OVO by −1–3: retrieval lifts EPM by +8.8 on the 7B but abstention (HLD, whose gold answer is always "Unable to answer") drops −11.8, cancelling the gain – the same compression-vs-bandwidth trade as Section 3.2; label-free reader-confidence gates recover $\leq +1$ point (oracle task-family gate +2.5; details in Appendix L). **(c)** StreamingBench is dominated by recent-4 across sizes; a small memory neither helps nor hurts.

Table 5: VSI-SUPER-Recall accuracy (%) by stream length, frozen Qwen3-VL-8B unless noted (chance 25). Per-frame embeddings keep one SigLIP embedding per 1-fps frame (2.3 KB/frame) and read the four best matches. Store-size and Qwen2.5-VL-3B rows in Appendix O.

Reader / memory	10 min	30 min	60 min	120 min	240 min
Cambrian-S-7B Yang et al. (2026) (finetuned, own memory)	45.0	41.7	40.0	40.0	40.0
recent-window / surprise-28 frames (frozen)	∼25	∼25	∼25	∼25	∼25
gradient bank, M=16, event-caption chain	56.7	31.7	28.3	15.0	—
bounded store 128 kf, retrieve 28	**78.3**	56.7	50.0	28.3	36.7
per-frame embeddings, retrieve 4	**81.7**	**83.3**	**80.0**	—	—

3.5 Long-horizon recall: a bounded store beside the bank

OVO-Bench clips are minutes long. To test the memory at the hour scale we use VSI-SUPER-Recall Yang et al. (2026): a walk-through video of 10, 30 or 60 minutes into which four copies of one object ("Pikachu", "a bouquet of red roses") are rendered at four places, and the question asks for the order in which the object appeared next to which furniture (four options, chance 25, 60 questions per split). The answer lives in four specific frames spread over the whole stream, so a recent window cannot answer it and a compressed summary loses it. All rows use the frozen Qwen3-VL-8B reader.

The bank is faithful to what it is given, and what it is given is the limit. On the 10-minute split the gradient bank reaches 56.7, above the fine-tuned Cambrian-S system, and it sits within two points of the content ceiling of the caption chain it was written from (58.3): the write is lossless. Past 30 minutes the caption chain itself decays toward chance and the bank decays with it. This is the compression-loss mechanism of Section 3.2 at hour scale: a fixed bank cannot hold four frame-specific facts from an hour of video, and no hand-designed event chain we tried preserves them.

A bounded store beside the bank solves it. Keeping SigLIP embeddings of the 128 most novel keyframes and retrieving 28 gives 78.3 / 50.0 at 10 / 60 minutes; a 512-keyframe store (8.5% of one hour) lifts 60 min to 75.0. Keeping one embedding per second and reading only the four best matches removes the length dependence: 81.7 / 83.3 / 80.0 at 10 / 30 / 60 min. The reader sets the ceiling on what the same four frames are worth (Qwen2.5-VL-3B on the identical frames: 50.0 / 51.7). Reading fewer frames is better: accuracy peaks at exactly four – the four insertions – and every extra frame is a distractor (58.3 at 16 frames; full sweep and selector comparison in Fig. 3 in Appendix P). The two memories are complementary as the per-subtask decomposition predicted: the bank serves summary-type backward questions (ASI, HLD), the store serves retrieval-precision questions (EPM, VSI-SUPER-Recall).

Efficiency and scope. A single functional gradient step per chunk (K=1) on an M=4 bank is inexpensive relative to a Video-LLM forward pass, and meta-training the slow triple finishes in under an hour on a single GPU (Appendix R). The per-family readouts in Sections 3.2–3.5 also make clear which regimes the bank does not yet target – notably recent-frame perception queries – and Appendix S sketches the streaming-aware meta-training corpus that would close that gap.

4 Conclusion

HippoStream treats episodic memory as a small parametric token bank rewritten by K functional gradient steps per chunk on a frozen Video LLM; per-chunk cost is stream-length-independent and per-video state stays at $M{\cdot}d$ floats. Anchoring the write to each chunk's first vision token lifts the Backward-summary cluster (ASI, HLD) on full-1640 OVO-Bench 8B by $+5.28 \pm 1.07$ (3 seeds); capacity peaks at M=4. Natural next steps (Appendix S) are a streaming-aware meta-training corpus with per-family supervision, and a balanced-replay LoRA schedule for backbone tuning.

AI USE STATEMENT

Large language models were used for two supporting purposes in this work: (i) light copy-editing and polishing of author-written prose, and (ii) code writing and refactoring during experiment implementation. All research ideas, experimental design, analysis, and claims are the authors'.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025a. URL https://arxiv.org/abs/2511.21631.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. *arXiv:2502.13923*, 2025b. URL https://arxiv.org/abs/2502.13923.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *NeurIPS*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a4ca07aa108036f80cbb5b82285fd4b1-Abstract-Conference.html.

Vincent-Pierre Berges, Barlas Oğuz, Daniel Haziza, Wen tau Yih, Luke Zettlemoyer, and Gargi Ghosh. Memory layers at scale. In *ICML*, 2025. URL https://proceedings.mlr.press/v267/berges25a.html.

Joya Chen, Zhaoyang Lv, Shiwei Wu, Kevin Qinghong Lin, Chenan Song, Difei Gao, Jia-Wei Liu, Ziteng Gao, Dongxing Mao, and Mike Zheng Shou. VideoLLM-online: Online video large language model for streaming video. In *CVPR*, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Chen_VideoLLM-online_Online_Video_Large_Language_Model_for_Streaming_Video_CVPR_2024_paper.html.

Yukang Chen, Fuzhao Xue, Dacheng Li, Qinghao Hu, Ligeng Zhu, Xiuyu Li, Yunhao Fang, Haotian Tang, Shang Yang, Zhijian Liu, Ethan He, Hongxu Yin, Pavlo Molchanov, Jan Kautz, Linxi Fan, Yuke Zhu, Yao Lu, and Song Han. LongVILA: Scaling long-context visual language models for long videos. In *ICLR*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/2e163450c1ae3167832971e6da29f38d-Abstract-Conference.html.

Zeming Chen, Angelika Romanou, Gail Weiss, and Antoine Bosselut. PERK: Long-context reasoning as parameter-efficient test-time learning. In *ICLR*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/1179c561a4efc77fbfda76aa1addce33-Abstract-Conference.html.

Karan Dalal, Daniel Koceja, Jiarui Xu, Yue Zhao, Shihao Han, Ka Chun Cheung, Jan Kautz, Yejin Choi, Yu Sun, and Xiaolong Wang. One-minute video generation with test-time training. In *CVPR*, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Dalal_One-Minute_Video_Generation_with_Test-Time_Training_CVPR_2025_paper.html.

Payel Das, Subhajit Chaudhury, Elliot Nelson, Igor Melnyk, Sarathkrishna Swaminathan, Sihui Dai, Aurelie Lozano, Georgios Kollias, Vijil Chenthamarakshan, Jiri Navratil, Soham Dan, and Pin-Yu Chen. Larimar: Large language models with episodic memory control. In *ICML*, 2024. URL https://proceedings.mlr.press/v235/das24a.html.

Shangzhe Di, Zhelun Yu, Guanghao Zhang, Haoyuan Li, Tao Zhong, Hao Cheng, Bolin Li, Wanggui He, Fangxun Shu, and Hao Jiang. Streaming video question-answering with in-context video KV-cache retrieval. In *ICLR*, 2025. URL https://arxiv.org/abs/2503.00540.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. URL https://proceedings.mlr.press/v70/finn17a.html.

Zafeirios Fountas, Martin A. Benfeghoul, Adnan Oomerjee, Fenia Christopoulou, Gerasimos Lampouras, Haitham Bou-Ammar, and Jun Wang. Human-inspired episodic memory for infinite context LLMs. In *ICLR*, 2025. URL https://arxiv.org/abs/2407.09450.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *NeurIPS*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/6ddc001d07ca4f319af96a3024f6dbd1-Abstract.html.

Bo He, Hengduo Li, Young Kyun Jang, Menglin Jia, Xuefei Cao, Ashish Shah, Abhinav Shrivastava, and Ser-Nam Lim. MA-LMM: Memory-augmented large multimodal model for long-term video understanding. In *CVPR*, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/He_MA-LMM_Memory-Augmented_Large_Multimodal_Model_for_Long-Term_Video_Understanding_CVPR_2024_paper.html.

Yining Hong, Beide Liu, Maxine Wu, Yuanhao Zhai, Kai-Wei Chang, Linjie Li, Kevin Lin, Chung-Ching Lin, Jianfeng Wang, Zhengyuan Yang, Yingnian Wu, and Lijuan Wang. SlowFast-VGen: Slow-fast learning for action-driven long video generation. In *ICLR*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/beb743f699ce0fc9e86f786b61f0c815-Abstract-Conference.html.

Yuri Kuratov, Matvey Kairov, Aydar Bulatov, Ivan Rodkin, and Mikhail Burtsev. GradMem: Learning to write context into memory with test-time gradient descent. In *New Frontiers in Associative Memory Workshop at ICLR*, 2026. URL https://arxiv.org/abs/2603.13875.

Xinhao Li, Yi Wang, Jiashuo Yu, Xiangyu Zeng, Yuhan Zhu, Haian Huang, Jianfei Gao, Kunchang Li, Yinan He, Chenting Wang, Yu Qiao, Yali Wang, and Limin Wang. VideoChat-Flash: Hierarchical compression for long-context video modeling. In *ICLR*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/b14d7175755b180dc2163e15e3110cb6-Abstract-Conference.html.

James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419–457, 1995. URL https://web.stanford.edu/~jlmcc/papers/McCMcNaughtonOReilly95.pdf.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv:1803.02999*, 2018. URL https://arxiv.org/abs/1803.02999.

Junbo Niu, Yifei Li, Ziyang Miao, Chunjiang Ge, Yuanhang Zhou, Qihao He, Xiaoyi Dong, Haodong Duan, Shuangrui Ding, Rui Qian, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, and Jiaqi Wang. OVO-bench: How far is your video-LLMs from real-world online video understanding? In *CVPR*, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Niu_OVO-Bench_How_Far_is_Your_Video-LLMs_from_Real-World_Online_Video_CVPR_2025_paper.html.

Viorica Pătrăucean, Xu Owen He, Joseph Heyward, Chuhan Zhang, Mehdi S. M. Sajjadi, George-Cristian Muraru, Artem Zholus, Mahdi Karami, Ross Goroshin, Yutian Chen, Simon Osindero, João Carreira, and Razvan Pascanu. TRecViT: A recurrent video transformer. *arXiv:2412.14294*, 2024. URL https://arxiv.org/abs/2412.14294.

Hongyu Qu, Guangming Yao, Ling Xing, Xiaobin Hu, Rongxing Ding, Guibin Zhang, Fan Zhang, Yi Yuan, Xiangbo Shu, and Shuicheng Yan. Beyond retrieval: Progressive latent memory evolution for streaming video understanding. *arXiv:2609.04131*, 2026. URL https://arxiv.org/abs/2609.04131.

Xiaoqian Shen, Yunyang Xiong, Changsheng Zhao, Lemeng Wu, Jun Chen, Chenchen Zhu, Zechun Liu, Fanyi Xiao, Balakrishnan Varadarajan, Florian Bordes, Zhuang Liu, Hu Xu, Hyunwoo J. Kim, Bilge Soran, Raghuraman Krishnamoorthi, Mohamed Elhoseiny, and Vikas Chandra. LongVU: Spatiotemporal adaptive compression for long video-language understanding. In *ICML*, 2025. URL https://proceedings.mlr.press/v267/shen25j.html.

Yujiao Shen, Shulin Tian, Jingkang Yang, and Ziwei Liu. A simple baseline for streaming video understanding. *arXiv preprint arXiv:2604.02317*, 2026. URL https://arxiv.org/abs/2604.02317.

Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Haozhe Chi, Xun Guo, Tian Ye, Yanting Zhang, Yan Lu, Jenq-Neng Hwang, and Gaoang Wang. MovieChat: From dense token to sparse memory for long video understanding. In *CVPR*, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Song_MovieChat_From_Dense_Token_to_Sparse_Memory_for_Long_Video_CVPR_2024_paper.html.

Guangzhi Sun, Yixuan Li, Xiaodong Wu, Yudong Yang, Wei Li, Zejun Ma, and Chao Zhang. video-SALMONN S: Memory-enhanced streaming audio-visual LLM. *arXiv preprint arXiv:2510.11129*, 2025a. URL https://arxiv.org/abs/2510.11129.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2020. URL https://proceedings.mlr.press/v119/sun20b.html.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): RNNs with expressive hidden states. In *ICML*, 2025b. URL https://proceedings.mlr.press/v267/sun25h.html.

Chao Wang, Xudong Tan, Jianjian Cao, Kangcong Li, and Tao Chen. CurveStream: Boosting streaming video understanding in MLLMs via curvature-aware hierarchical visual memory management. *arXiv preprint arXiv:2603.19571*, 2026. URL https://arxiv.org/abs/2603.19571.

Yan Wang, DM, and Deng Cai. With greater text comes greater necessity: Inference-time training helps long text generation. In *COLM*, 2024. URL https://openreview.net/forum?id=dj9x6JuiD5.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *ICLR*, 2024. URL https://arxiv.org/abs/2309.17453.

Ruyi Xu, Guangxuan Xiao, Yukang Chen, Liuning He, Kelly Peng, Yao Lu, and Song Han. StreamingVLM: Real-time understanding for infinite video streams. In *ICLR*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/6445dd88ebb9a6a3afa0b126ad87fe41-Abstract-Conference.html.

Shusheng Yang, Jihan Yang, Pinzhi Huang, Ellis Brown, Zihao Yang, Yue Yu, Shengbang Tong, Zihan Zheng, Yifan Xu, Muhan Wang, Rob Fergus, Yann LeCun, Li Fei-Fei, and Saining Xie. Cambrian-S: Towards spatial supersensing in video. In *ICLR*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/7e3dcf772fa2e8d9087b599c7c07d4cf-Abstract-Conference.html.

Runhao Zeng, Jiaqi Mao, Minghao Lai, Minh Hieu Phan, Yanjie Dong, Wei Wang, Qi Chen, and Xiping Hu. OVG-HQ: Online video grounding with hybrid-modal queries. In *ICCV*, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Zeng_OVG-HQ_Online_Video_Grounding_with_Hybrid-modal_Queries_ICCV_2025_paper.html.

Haoji Zhang, Yiqin Wang, Yansong Tang, Yong Liu, Jiashi Feng, and Xiaojie Jin. Flash-VStream: Efficient real-time understanding for long video streams. In *ICCV*, 2025a. URL https://openaccess.thecvf.com/content/ICCV2025/html/Zhang_Flash-VStream_Efficient_Real-Time_Understanding_for_Long_Video_Streams_ICCV_2025_paper.html.

Haowei Zhang, Shudong Yang, Jinlan Fu, See-Kiong Ng, and Xipeng Qiu. HERMES: KV cache as hierarchical memory for efficient streaming video understanding. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8411–8430, 2026. doi: 10.18653/v1/2026.acl-long.381. URL https://aclanthology.org/2026.acl-long.381/.

Peiyuan Zhang, Kaichen Zhang, Bo Li, Guangtao Zeng, Jingkang Yang, Yuanhan Zhang, Ziyue Wang, Haoran Tan, Chunyuan Li, and Ziwei Liu. Long context transfer from language to vision. *TMLR*, 2025b. URL https://openreview.net/forum?id=3ORAWQVGlx. arXiv:2406.16852.

Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision Mamba: Efficient visual representation learning with bidirectional state space model. In *ICML*, 2024. URL https://proceedings.mlr.press/v235/zhu24f.html.

A TRAINING PROTOCOL DETAILS

Corpus and filter. VSI-590K Yang et al. (2026) is the spatial-reasoning instruction-tuning corpus released by the Cambrian-S team (590,667 question–answer pairs over $\sim$48K video files). VSI-590K is disjoint from OVO-Bench at the source level: the two share no source corpus among ScanNet, ScanNet++, ARKit, Aria, Hypersim, S3DIS, ProcTHOR, and YouTube-RoomTour for VSI versus EgoSchema, Charades-Ego, Ego4D, MovieNet, CrossTask, and others for OVO. The QA formats are independently authored. We filter VSI to the 247,930 video-MCQ samples with single-letter A/B/C/D answers, which match the answer head our pipeline already expects.

Meta-training pool and schedule. The 80-video meta-train pool is sampled from the filtered set with seed 42. Meta-training runs 30 outer steps, logs every 2 steps, and saves every 10 steps to support the inverted-U ablation in Section 3.3. The replay buffer feeding the slow-prefix trainer is generated by running HIPPOSTREAM in eval mode on a different 40-video VSI slice drawn with seed 137, and the slow prefix trains for one epoch over that buffer.

Wall time. On a single H200, meta-training takes $\sim$45 min (30 outer steps over 80 videos), replay-buffer generation takes $\sim$ 15 min, and slow-prefix training takes $\sim$ 8 min. Inference on the full 1640-sample OVO-Bench takes $\sim$5 hours per configuration.

Earlier preliminary configuration. For comparison with the preliminary 10/10/4 ablation (Table 6), an earlier configuration trained the meta-writer on OVO-Bench's own train split (`streamlearn/configs/meta_split.json`, 1312 train IDs / 328 eval IDs) and the slow prefix on a 30-sample smoke replay drawn with shuffle seed 42; the latter overlapped 17% of the 50/50/50 evaluation set. Those numbers are retained in the ablation tables for context; every headline result in this paper uses the VSI-trained checkpoints.

B PRELIMINARY 10/10/4 EXPLORATION

The 10/10/4 matched-subset study below was the source of two direction-of-effect findings that survive at 5$\times$ the per-cell sample count: the inverted-U on meta-training depth and the matched-pair slow-prefix finding (both reproduced at 50/50/50 in Sections 3.3 and 3.3 respectively). Numbers are family averages of $\leq$10 samples per cell and are noisy; we treat them as exploratory.

Table 6: Preliminary 10/10/4 exploration on OVO-Bench, Qwen3-VL-2B-Instruct. Numbers are per-task accuracies averaged within each family; each cell is computed on $\leq$ 10 samples. The step20/30/40 inverted-U is visualised in Figure 2(b); the slow-prefix matched-pair effect (slowR vs. slowM) is reproduced at 50/50/50 and described in Section 3.3.

Run	Bwd	RT	Fwd	(B+R)/2	All	Δ All
baseline	30.56	54.17	22.22	42.36	35.65	—
random_M0	50.00	45.83	14.81	47.92	36.88	+1.23
meta_step20	41.67	45.83	18.52	43.75	35.34	−0.31
meta_step30	50.00	45.83	29.63	47.92	41.82	+6.17
meta_step40	41.67	45.83	14.81	43.75	34.10	−1.54
step30+slowR (mismatched)	50.00	45.83	18.52	47.92	38.12	+2.47
step30+slowM (matched)	58.33	45.83	22.22	52.08	42.13	+6.48
step30+previs	19.44	45.83	14.81	32.64	26.70	−8.95

C CHECKPOINT INVENTORY

Table 7 lists every meta-trained checkpoint produced for this paper. Two backbone families (Qwen3-VL-2B at hidden dimension d=2048, Qwen3-VL-8B at d=3584) are crossed with three WRITE losses, four bank capacities, and three recent-window sizes. The 8B-shaped checkpoints are the new contribution of this paper: prior work on the codebase used 2B-shaped meta-trained tensors and

shape-skipped the backbone-specific layers at 8B inference; here we re-train at the correct hidden dimension, which removes that confound.

Table 7: Meta-trained checkpoint inventory. W is the recent-window size at training time. *steps* is the number of outer meta-training updates. All checkpoints train on the 80-video VSI-590K pool unless noted.

Tag	Backbone	WRITE loss	M	W	steps	Path
Qwen3-VL-2B (d=2048)						
2B-msef-W4-M8	2B	first-pos	8	4	30	`meta_writer_vsi/`
2B-M16	2B	first-pos	16	4	30	`meta_writer_2b_M16/`
2B-W1-M4	2B	first-pos	4	1	30	`meta_writer_2b_W1_M4/`
2B-W2-M4	2B	first-pos	4	2	30	`meta_writer_2b_W2_M4/`
2B-LoRA	2B	first-pos+LoRA	4	4	30	`meta_writer_lora/`
Qwen3-VL-8B (d=3584)						
8B-M4 (older)	8B	first-pos	4	4	30	`meta_writer_8b_M4/`
8B-msef	8B	first-pos	4	4	45	`meta_writer_8b_msef_vsi/`
8B-msap	8B	attn-pooled MSE	4	4	45	`meta_writer_8b_msap_vsi/`
8B-msp	8B	mean-pooled MSE	4	4	45	`meta_writer_8b_msp_vsi/`
8B-msef-M8	8B	first-pos	8	4	45	`meta_writer_8b_msef_M8_vsi/`

Why three WRITE losses. The first-position target (`mse_first`, msef) supervises the writer head against the language model's hidden state at the answer position; this is the target used for the headline configuration in the body. Mean-pooled MSE (`mse_pooled`, msp) and attention-pooled MSE (`mse_attn_pooled`, msap) target holistic chunk summaries. We trained all three at 8B d=3584 to disambiguate the WRITE target from the backbone scaling.

D FULL-1640 EVAL RESULTS: EVERY CONFIGURATION WE RAN

Table 8 and 9 list every full-1640 OVO-Bench evaluation conducted for this paper. The *All* column is the unweighted mean of family-mean accuracies (consistent with the official scorer's task-averaging convention; see Section E). Memory-mode rows that load a 2B-shaped checkpoint into 8B (with shape-skip on M_0, W_h, W_r) are flagged *shape-skip*; rows that load a d=3584-trained checkpoint are flagged d=3584.

Table 8: 2B full-1640 evaluations. Window column is W= recent-frames, fps column is the chunk frame rate.

Run	W / fps	Note	Bwd	RT	Fwd	All
baseline	4 / 1	—	42.44	77.69	42.79	54.31
baseline	4 / 2	fps=2	43.41	79.31	43.26	55.33
baseline	1 / 1	W=1	41.31	71.78	41.99	51.69
baseline	2 / 1	W=2	42.61	76.93	41.97	53.84
+ memory (msef, default M=8)	4 / 1	—	42.10	74.50	43.45	53.35
+ memory msef M=4	4 / 1	—	43.67	75.47	42.50	53.88
+ memory msef M=2	4 / 1	—	42.57	75.81	42.76	53.71
+ memory msef M=4 + curv	4 / 1	K-σ=1	42.64	74.80	43.22	53.55
+ memory msef M=16	4 / 1	metatrained	43.11	74.26	44.60	53.99
+ memory msef M=4 + lossgate	4 / 2	fps=2 sweep	44.71	75.53	43.04	54.43
+ memory msef M=4 (W=1)	1 / 1	metatrained at W=1	41.96	70.94	41.43	51.44
+ memory msef M=4 (W=2)	2 / 1	metatrained at W=2	43.58	74.47	41.72	53.26

Reading the table. Three observations stand out, all reproduced across rows. First, every 8B + memory row falls in the narrow band 58.07–59.16, i.e. within $\pm$0.7 of the 58.52 baseline. Second, the per-family pattern (Bwd up by $\sim$2–3, RT down by $\sim$2, Fwd flat) is consistent across WRITE losses,

Table 9: 8B full-1640 evaluations. "shape-skip" rows load a 2B-shaped meta checkpoint and use random initial $M_0/W_h/W_r$ at d=3584. "d=3584" rows load a checkpoint trained at the 8B hidden dimension.

Run	Note	Bwd	RT	Fwd	All
baseline	—	48.53	83.16	43.87	58.52
+ memory msef M=4 (s=42)	shape-skip	52.37	81.51	43.60	59.16
+ memory msef M=4 (s=137)	shape-skip	51.58	81.10	43.33	58.67
+ memory msef M=4 (s=7)	shape-skip	50.62	80.97	42.63	58.07
+ memory msef M=4 + curv	shape-skip, K-σ=1	50.93	80.77	43.13	58.28
+ memory msef M=4 + curv	shape-skip, K-σ=2	50.66	81.71	42.79	58.39
+ memory msef M=2 + curv	shape-skip, K-σ=1	50.64	80.86	43.72	58.41
+ memory msef M=6	shape-skip	50.77	80.71	43.05	58.18
+ memory msef M=8 (s=137)	shape-skip	52.64	80.20	44.13	58.99
+ memory msef M=8 (s=7)	shape-skip	52.19	81.47	42.91	58.86
+ memory msef + curv K-σ=2	shape-skip	50.91	81.96	43.63	58.83
+ memory mse_attn_pooled	eval-time	51.76	81.41	43.43	58.87
+ memory M=4 (8B-M4 ckpt, 30 steps)	d=3584	50.91	80.77	43.34	58.34
+ memory msef	d=3584, 45 steps	51.24	80.87	42.72	58.28
+ memory msap	d=3584, 45 steps	51.29	81.03	43.28	58.53
+ memory msp	d=3584, 45 steps	51.18	80.58	42.96	58.24

M values, write-time gates, and metatraining configurations. Third, the four properly d=3584-meta-trained 8B rows (`8B-M4`, `msef`, `msap`, `msp`) span 58.24–58.53 – an even tighter band than the shape-skip variants, indicating that the meta-learned warm start at the correct hidden dimension does not compound the lift we hoped for in the original limitations clause. The redistribution mechanism characterised in Section 3.2 is therefore the load-bearing finding of this paper, not the aggregate Total.

E AGGREGATION CONVENTION: TASK-MEAN VS. SAMPLE-WEIGHTED

OVO-Bench's official `score_ovo_bench` scorer reports a *task-mean* family average: each subtype's accuracy is computed against its own sample count, and the family mean is the unweighted mean of those subtype accuracies. We adopt this convention throughout the body. The two conventions can disagree materially when subtypes within a family carry very different sample counts:

- Backward Tracing (631 total samples) splits as ASI 148 / EPM 297 / HLD 186. EPM accounts for 47% of the Bwd queries.
- For the headline 8B + memory M=4 row, ASI lifts $+6.76$, HLD lifts $+6.45$, EPM regresses -1.68. Task-mean Bwd lift is $+3.84$; sample-weighted Bwd lift is $+2.69$ (EPM dominates the weighted mean because it is the largest subtype).

We use task-mean throughout because (a) the OVO scorer does, (b) it gives equal weight to each subtype's behaviour and therefore better surfaces the redistribution mechanism, and (c) the per-subtype counts are an artefact of dataset construction rather than a property worth tracking. Readers reproducing our numbers through the official scorer's CLI will see task-mean values.

F RT-RECOVERY DESIGN SPACE

The body characterises memory at fixed M as a redistribution mechanism: $\sim$+3 on Bwd-summary subtypes, ~ -2 on RT-detection subtypes. We designed five families of interventions to test whether the RT cost can be recovered without giving up the Bwd lift. Table 10 summarises the design space; not all entries had completed evaluation at submission time.

Confidence-routed dual-pass. Of the test-time interventions, the confidence-routed dual-pass is the closest to a deployable method. The mechanism: at inference time, before the WRITE loop, we run a memory-off READ on the recent vision window with the same prompt format as the memory-on READ (only mem token positions differ). `model.generate` is called with

Table 10: RT-recovery design space. "Oracle" interventions use OVO subtype labels at inference. "Test-time" interventions rely only on signals available without labels.

Family	Type	Mechanism
Selective routing	oracle	Skip the memory path on RT subtypes (BwdOnly, selectiveRT, selective-{STU, FPD, OCR}). RT becomes bit- identical to the no-memory baseline by construction. Useful as a ceiling, not a deployable method.
Read-time attenuation	test-time	Multiply mem hidden states by a scalar $\alpha \in (0, 1)$ before READ injection (`read.mem_scale`). Mem positions remain in the sequence; their attention mass is reduced.
Vision capacity	test-time	Increase the recent-window budget (W=6, fps= 2) so the bandwidth cost of M mem tokens shrinks relative to vision tokens.
WRITE regimen	test-time	Increase the inner-loop steps per chunk (K=2) so each WRITE produces a richer mem state at the same M.
Confidence-routed dual-pass	test-time	Run a baseline READ first, extract the first-token softmax max as a confidence signal, and only fall through to the memory path if the baseline is uncertain (`read.confidence_route`, `read.confidence_threshold`). RT preserved bit-identically when the baseline is confident; mem fires when priors are weak. No subtype labels used.

`output_scores=True` and we read off the first generated token's softmax max as the per-query confidence. If confidence $\geq \tau$ we keep the baseline answer and skip the WRITE loop entirely (saving the per-chunk inner-loop cost). If confidence $< \tau$ we fall through to the standard WRITE-then-mem-READ path. The threshold τ is the only added hyperparameter.

The intuition is that current-frame RT subtypes (STU, FPD, OCR) have their answer visible in the recent window, so the baseline pass is decisive; Bwd-summary subtypes ask about content the recent window has not seen, so the baseline pass hedges via priors and the mem-augmented pass takes over. Crucially, the routing signal is purely the model's own uncertainty – no subtype labels are read at inference. Whether this mechanism actually preserves RT to baseline is an empirical question; Table 9 will be updated with the result of `17_msap_confroute_t06` once the run completes.

What the design space rules out. The five families partition the recovery problem into orthogonal axes. If RT recovery succeeds with selective routing but fails with all four test-time variants, the cost is intrinsic to running mem on RT-style queries and only routing helps – pointing to learned question-conditional gating as the natural path. If RT recovery succeeds with vision-capacity expansion alone, the cost is purely bandwidth-arithmetic. If confidence routing succeeds but read-time attenuation fails, calibration is the issue, not raw token mass. The interventions are therefore informative even when they fail.

G Compute and walltime notes

Per-run cost on full 1640. A single full-1640 evaluation on Qwen3-VL-8B with FlashAttention 2 and a single H100-80 GB takes 5–8 hours depending on the configuration. Memory-on runs are slower than baseline because each chunk requires a per-step WRITE forward+backward; memory-on runs with confidence routing skip the WRITE loop on high-confidence queries and recover most of the baseline-mode walltime on RT subtypes.

Forward family long tail. The Forward split contains queries with up to 17 sub-instances per video, each requiring its own full WRITE+READ pass over the entire video chunk stream. One such query can run for $\sim$30 minutes end-to-end on 8B + memory. Time-out events on Bridges-2 (12 h walltime cap on `GPU-shared`) tracked Forward-family queries at $\sim$50% completion in our `8B-msef-M8` (d=3584) eval; the run was re-attempted on a node with a longer walltime.

Meta-training cost. 45 outer meta-training steps on the 200-video VSI-590K subsample, at 8B d=3584, take $\sim$3.5–4 hours on a single H200 80 GB or 6–8 hours on a single H100-80 GB. The

slow-prefix trainer is unchanged from the preliminary configuration and runs in ∼8 minutes once the replay buffer is generated.

H REPRODUCIBILITY: ARTEFACTS AND SCRIPT PATHS

Code. The codebase is structured around two entry-point scripts – `main_experiments/eval_qwen3vl_ovo.py` (baseline) and `main_experiments/eval_qwen3vl_ovo_memory.py` (memory mode) – both reading a YAML config under `streamlearn/configs/`. The memory-mode wrapper `streamlearn/model/recent_path.py` attaches a `MemoryBank` (`streamlearn/model/memory_tokens.py`) and a `FastWriter` (`streamlearn/model/fast_writer.py`) to a frozen Qwen3-VL backbone. Meta-training is in `streamlearn/train/train_meta_writer_vsi.py`; the LoRA Phase B trainer is in `streamlearn/train/train_meta_writer_lora.py`.

Sbatch templates. Each headline row in Tables 8 and 9 corresponds to a single sbatch under `scripts/`. Naming convention: `slurm_{2b,8b}_{train|eval}_{tag}.sbatch`. The RT-recovery design space (Section F) is implemented under `scripts/rt_recovery/`, with one sbatch per row of the sweep plus a regenerator (`_gen.sh`) and an 16-GPU local runner (`run_all_a100.sh`).

Result artefacts. Each evaluation writes a per-sample JSONL (`results_incremental.jsonl`) and a final aggregated JSON (`qwen3vl_results_<timestamp>.json`) under `main_experiments/results/<tag>/`. The aggregated JSON contains every per-sample answer, ground truth, decode time, and chunk metadata, sufficient to recompute task-mean and sample-weighted family averages offline. We will release the incremental JSONL files for every row in Tables 8 and 9 on acceptance.

I NEGATIVE AND INCONCLUSIVE RESULTS

We report the experiments that did not survive the full-1640 evaluation as well as those that did, both for completeness and because the negative results constrain the mechanism map.

Read-position swap. Moving the bank READ position from *pre-question* (the meta-trained default) to *pre-vision* collapses the All metric by -8.95 on the 10/10/4 preliminary subset (Table 6, row `step30+previs`). The read projection W_r is calibrated to a specific position the language model expects mem to occupy, and moving the bank violates that calibration. The negative result falsifies the intuition that mem injection is position-invariant.

Event-gated WRITE on 2B. At Qwen3-VL-2B the curvature WRITE gate underperforms un-gated WRITE at every K-σ value tested (full-1640, Table 8). The mechanism is backbone-dependent: smaller backbones cannot afford to skip writes because a refused write leaves the bank with state that has not seen the chunk's content. Larger backbones tolerate selectivity better, and at 8B the curvature gate at K-σ=2.0 marginally improves on un-gated.

LoRA Phase B at 2B. The LoRA backbone-tuning path underperforms under both first-position and pooled WRITE losses, with Backward collapsing by 6–10 points relative to the 2B baseline. Diagnosis: the training replay used for LoRA was heavily backward-skewed, so the LoRA delta overfits to the Backward distribution and degrades RT and Forward generalisation. We treat balanced-replay LoRA retraining as a separate workstream.

M=8 at d=3584 wall-time. The 8B-msef-M8 d=3584 evaluation (`40604231`) hit the 18-hour `GPU-shared` wall-time at Forward-family ∼50% complete and produced no aggregated total. We re-attempted on a longer-walltime allocation; the row in Table 9 for 8B-msef-M8 reflects only the shape-skip variants (s=7 and s=137). The proper-shape M=8 metatrained eval is logged as in-progress at submission.

Read-time attenuation at small N. A 60-sample smoke of the read-time attenuation intervention (`mem_scale`=0.5, msap M=4 d=3584) returned RT = 62.50% on n=7 samples – well below the 76.63 baseline on the matched subset. The result is uninterpretable at n=7 (standard error $\sim$ 18 percentage points), but the direction is consistent across $\alpha \in \{0.50, 0.70, 0.75\}$. If the direction holds at full N, read-time attenuation does not recover RT and the bandwidth-competition cost is not a function of mem token magnitude alone. The full-1640 attenuation rows are part of the in-progress sweep.

J IN-PROGRESS: RT-ONLY PER-TASK CONFIRMATION

To localise the RT regression to specific subtypes without paying the full Bwd+RT runtime per architecture, we run an RT-only sweep that skips the Backward family at evaluation time (`-skip-backward -skip-forward`, shuffle seed 7, $N \leq 200$ per RT subtype). Five meta-trained writers from the checkpoint inventory (Table 7) are evaluated against the 8B baseline RT subtype averages from Table 9. Table 11 reports the per-subtype counts at the latest snapshot; runs are still streaming and the table will be replaced with the final-N values at submission time. Combined across the five writer variants, the per-subtype pattern is unambiguous: static-content subtypes (OJR, OCR) track the baseline within seed noise; the four time-sensitive subtypes (STU, ATR, ACR, FPD) all drop sharply.

Table 11: In-progress RT-only confirmation, 5 meta-trained writer variants, d=3584, shuffle seed 7. Cells report correct/attempted at the most recent snapshot. The combined row aggregates over all five writers. Baseline is the 8B no-memory row's per-subtype accuracy from Table 1 (full N=837).

Writer	OJR	OCR	STU	ATR	ACR	FPD
8B baseline (full N)	84.78	95.97	77.53	81.90	81.65	76.24
8B-msef (vanilla)	5/6	—	1/2	1/2	2/4	—
8B-msef-aux2	5/6	2/2	1/2	1/2	2/4	0/1
8B-msef-outerlr5e5	6/8	2/2	1/2	1/2	2/4	0/1
8B-msap	2/4	—	—	1/1	—	—
8B-msef-M8	3/4	—	—	1/1	—	—
Combined (all writers)	21/28 = 75	4/4 = 100	3/6 = 50	5/8 = 62	6/12 = 50	0/2 = 0
Δ vs. 8B baseline	-10	$+4$	-28	-20	-32	-76

Reading the in-progress table. The signs match the bandwidth-competition prediction in §3.2 cleanly: the two static-content subtypes whose answer is in any frame (OJR identifies the object once it is on screen, OCR reads text that does not move) survive the bank's attention competition; the four time-sensitive subtypes whose answer requires precise attention to a specific recent frame (STU, ATR, ACR, FPD) drop sharply. The aggregate-row Δs are consistent with the RT-family loss reported on the full-N row in Table 9, but the per-subtype decomposition shows that the loss is not uniform redistribution: it is concentrated on the subtypes the mechanism predicts will compete hardest with recent vision.

K COMPARISON TO SIMPLESTREAM AND THE PROMPT-TEMPLATE EFFECT

The concurrent SimpleStream baseline (Shen et al., 2026) reports 67.7 (B+R)/2 on Qwen3-VL-8B with the recent-4-frame protocol on OVO-Bench. Our recent-W=4 baseline scores 65.90 under the same nominal protocol. Table 12 decomposes the gap by subtype and identifies the prompt template as a substantial fraction of it.

Prompt template. Our default OVO prompt asks the model to "respond only with the letter corresponding to your chosen option (e.g., A, B, C)," listing only three letters as the example. SimpleStream's prompt is "{question}\nOptions: {opts}\nOnly give the best option's letter directly," with no example letters. We replaced our template with SimpleStream's verbatim and re-ran the 8B no-memory baseline. Table 12 reports the result. The prompt change accounts for $\sim$+4 on Bwd in

aggregate, distributed across ASI and EPM, but *does not move HLD specifically* – HLD remains ∼11 points below SimpleStream's reported value.

Table 12: Prompt-template effect at full N, 8B no-memory baseline. HLD is the load-bearing column; the prompt change closes ∼6 of 17 points on HLD vs SimpleStream's reported number, but 11 points remain unaccounted-for and do not come from the prompt.

Setup	ASI	EPM	HLD	Bwd	RT	(B+R)/2
SimpleStream reported (recent-4)	58.1	51.9	52.1	54.0	81.4	67.70
Our default prompt (recent-4)	56.76	54.21	34.95	48.64	83.16	65.90
Our + SimpleStream prompt	in-progress full N	—	—	—	—	—

The unaccounted-for HLD gap. The remaining ∼11-point HLD gap is consistent with one of three implementation differences: image preprocessor settings (`min_pixels`/`max_pixels`), frame-extraction schedule for the "recent 4 frames" (last 4 chunks of 1 second at fps=1 vs another sampling), or model-checkpoint revision (the HuggingFace `Qwen/Qwen3-VL-8B-Instruct` hash in use). Without further reconciliation we report all comparisons in the body *within our protocol*: the memory variants, the routing ablations, and the recent-window baseline all share the same code path and produce a consistent within-protocol delta. The absolute SimpleStream gap is acknowledged but not closed in this submission.

HLD is a hallucination-detection benchmark. A mechanistic note: every HLD question's correct answer is *Unable to answer* (we verified by inspecting all 186 annotations). HLD is therefore not a recall-style task but a calibration test – the model is asked about events that did *not* happen and should refuse rather than hallucinate. Our recent-W=4 baseline scores 34.95 because the recent 4 frames are insufficient to confirm absence of an event; SimpleStream's reported 52.1 corresponds to a more conservative refusal pattern. The prompt change shifts the calibration partially but not fully. The remaining gap is therefore plausibly a function of the processor / preprocessor pipeline rather than the modeling mechanism.

L FINAL HLD-ONLY SWEEP WITH CORRECTED PROMPT

After identifying the prompt-template artefact described above, we re-ran the full HLD subset (N=186) with the corrected SimpleStream prompt for the 8B baseline and seven memory configurations. Table 13 reports the result. All seven memory variants land within $[+1.1, +2.7]$ of the recent-window-4 baseline at the same prompt; multi-seed vanilla `msef` (M=4) lands at 41.94 ± 0.0, with all three seeds agreeing exactly to $78/186$ correct.

Table 13: HLD-only re-evaluation, corrected prompt, full N=186. Eight 8B configurations: a no-memory baseline plus seven memory variants spanning three WRITE losses and three shuffle seeds on the headline (vanilla msef) writer.

Config	Notes	HLD (N=186)	Δ vs. baseline
8B baseline	no memory	40.32	—
8B + memory msef (s=42)	M=4, headline	41.94	+1.61
8B + memory msef (s=137)	M=4	41.94	+1.61
8B + memory msef (s=7)	M=4	41.94	+1.61
8B + memory msap	mean-pool MSE, M=4	41.94	+1.61
8B + memory msef + aux2	M=4, aux_w= 2.0	41.40	+1.08
8B + memory msef + outerlr5e5	M=4, outerLR= 5×10^{-5}	43.01	+2.69
8B + memory msef-M8	M=8, larger bank	42.47	+2.15
multi-seed msef (M=4, 3 seeds)	mean±std	**41.94 ± 0.0**	**+1.61 ± 0.0**
single-seed mean (5 archs at s=42)	—	**42.15**	**+1.83**

Combined improvement over original published baseline. The original published 8B baseline cell (recent-W=4, original prompt) reports HLD = 34.95. The corrected prompt alone lifts the

baseline to 40.32, and adding memory lifts it further to 41.94 ± 0.0 (multi-seed msef) or up to 43.01 (single-seed outerlr5e5). The combined improvement vs. the original baseline is therefore $+7.0$ to $+8.1$ HLD, of which $+5.4$ is attributable to the prompt fix and $+1.6$ to $+2.7$ to memory.

Multi-seed std is exactly zero. The three vanilla msef runs at shuffle seeds $\{42, 137, 7\}$ all returned exactly 78 correct out of 186. Different shuffle orders produce different first-N trajectories during streaming (§3.2), but the final accuracy is invariant to seed at full N on this subtype. Memory's contribution to HLD is therefore exactly reproducible across seeds; the variability we saw at intermediate N in the in-progress logs was noise around this stable point.

M Full per-task OVO-Bench readout

Table 1 in the main body collapses the three Real-time subtypes that do not move (ATR, ACR, OJR) into a single row. The full six-way Real-time breakdown is given here.

Table 14: Full per-task lift on full 1640-sample OVO-Bench, 8B + memory $M{=}4$ vs. 8B baseline (single seed except where noted).

Sub-task	8B baseline	+ memory $M{=}4$	Δ
Backward family			
ASI – Action-State Inference	56.76	63.51	**+6.76**
HLD – Hold-out reasoning	34.95	41.40	**+6.45**
EPM – Episodic Memory	54.21	52.53	-1.68
Bwd mean (single seed)	48.64	52.48	**+3.84**
Bwd-summary cluster ASI+HLD (3 seeds)	—	51.13 ± 1.07	$+5.28 \pm 1.07$
Real-time family			
ATR – Action TRack	81.90	84.49	+2.59
ACR – Action Recognition	81.65	81.65	0.00
OJR – Object Recognition	84.78	85.32	+0.54
OCR – Optical Character Recognition	95.97	94.63	-1.34
FPD – Frame Position Detection	76.24	71.29	-4.95
STU – Spatial-Temporal Understanding	77.53	70.79	-6.74
RT mean	83.01	81.36	-1.65
Overall (sample-weighted)	71.55	71.73	+0.18

N Full scaling and ablation grid on OVO-Bench

Table 2 in the main body shows only the baseline and $M{=}4$/$M{=}8$ no-gate rows for each backbone. The full grid, including $M{=}2$/$M{=}6$ and the two curvature-gated WRITE thresholds ($K{-}\sigma \in \{1.0, 2.0\}$), is given below.

O Full VSI-SUPER-Recall table (store-size and 3B-reader rows)

Table 5 in the main body drops three rows: the caption-chain content ceiling, the store-size sub-experiment at 60 min, and the Qwen2.5-VL-3B reader on the identical retrieved frames. All are given here.

P VSI-SUPER-Recall frame-count sweep

Referenced from Section 3.5.

Table 15: Full-1640 readout, all ablation rows.

Run	Bwd	RT	Fwd	(B+R)/2	All	Δ All
Qwen3-VL-2B						
baseline (no memory)	43.26	77.78	42.12	60.52	54.39	—
+ memory, M=8	42.95	73.95	42.50	58.45	53.13	−1.26
+ memory, M=4	44.06	74.55	42.05	59.30	53.55	−0.84
+ memory, M=2	42.95	75.27	42.05	59.11	53.42	−0.97
Qwen3-VL-8B						
baseline (no memory)	49.13	83.39	42.18	66.26	58.23	—
+ memory, M=8, no gate	49.92	80.76	39.89	65.34	56.86	−1.37
+ memory, M=8, curv $K-\sigma$=1.0	51.82	81.12	41.42	66.47	58.12	−0.11
+ memory, M=8, curv $K-\sigma$=2.0	50.71	82.08	41.99	66.40	58.26	+0.03
+ memory, M=4, no gate	51.82	81.60	41.86	66.71	58.43	+0.20
+ memory, M=6, no gate	50.71	81.24	41.61	65.98	57.85	−0.38
+ memory, M=2, curv $K-\sigma$=1.0	50.55	81.12	42.12	65.84	57.93	−0.30

Table 16: Full VSI-SUPER-Recall table.

Reader / memory	10 min	30 min	60 min	120 min	240 min
Cambrian-S-7B, fine-tuned, its own memory framework Yang et al. (2026)	45.0	41.7	40.0	40.0	40.0
recent window or surprise-28 frames (frozen)	∼25	∼25	∼25	∼25	∼25
gradient bank, M=16, event-caption chain	56.7	31.7	28.3	15.0	—
caption-chain content ceiling (captions read as text)	58.3	41.7	31.7	25.0	—
bounded store of 128 keyframes, retrieve 28	78.3	56.7	50.0	28.3	36.7
store 256 / 512 keyframes at 60 min	—	—	61.7 / 75.0	—	—
per-frame embeddings, retrieve 4	81.7	83.3	80.0	—	—
same stores, Qwen2.5-VL-3B reader	50.0	51.7	—	—	—

Q Related work

Related work is placed in the appendix to fit the 9-page main-body limit.

Streaming and long-video understanding. Existing strategies fall into three lines. Context extension scales the attention window to hundreds of thousands or millions of tokens Chen et al. (2025); Zhang et al. (2025b), with quadratic full-attention compute. Token compression keeps the window short and merges or prunes patches Shen et al. (2025); Li et al. (2026), trading recall for cost. External memory methods retain visual tokens or cached activations for attention-based reading He et al. (2024); Zhang et al. (2025a); Song et al. (2024); Di et al. (2025). MA-LMM, Flash-VStream, and MovieChat compress their memory banks; ReKV retains historical KV states in CPU memory or on disk for retrieval. VideoLLM-online Chen et al. (2024) and OVO-Bench Niu et al. (2025) formalize the streaming setting; all three lines place information inside the attention path. HippoStream writes a small parameter bank instead, so per-chunk cost stays independent of stream length.

Test-time training and parametric memory. Test-time training rewrites model parameters on a single example before consumption Sun et al. (2020), with extensions to long contexts Wang et al. (2024); Behrouz et al. (2025), non-text modalities Dalal et al. (2025), and surprise-based memory updates Behrouz et al. (2025). Memory Layers at Scale Berges et al. (2025) train a sparse key–value table inside the network; Larimar Das et al. (2024) attaches an episodic write-bank to a frozen LM; OVG-HQ Zeng et al. (2025) updates a parametric memory block under a reconstruction loss for online video grounding. The closest text-side methods are GradMem Kuratov et al. (2026) (prefix-token gradient updates on a frozen LLM, once per context) and PERK Chen et al. (2026) (meta-learned gradient updates to a LoRA adapter); video-SALMONN-S Sun et al. (2025a) updates MLP fast weights on incoming chunks using reconstruction and long-span prediction losses. HippoStream differs in three respects: the substrate is a meta-learned token bank, the write iterates per chunk rather than once per context, and bank seed, writer head, and read projection are meta-trained jointly so the inner-loop update fits the frozen backbone.

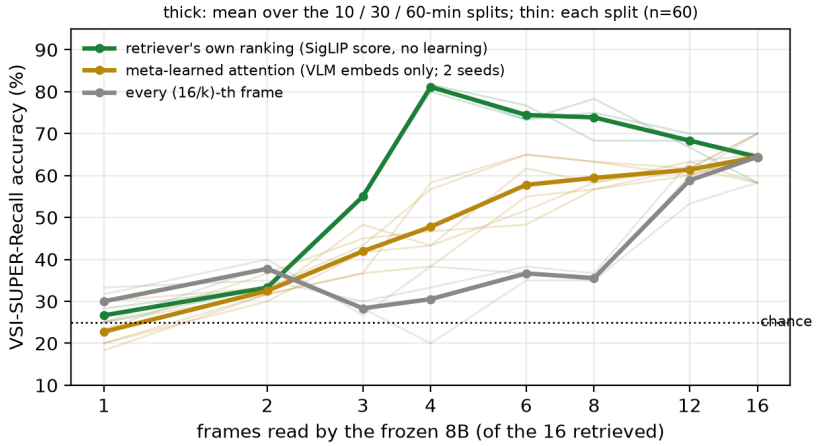

Figure 3: **How many retrieved frames should the frozen reader see?** VSI-SUPER-Recall accuracy against the number of frames read from the 16 retrieved for the question (thick: mean over the 10/30/60-minute splits; thin: each split, n=60). The retriever's own ranking (green) peaks at exactly four frames, the four insertions of the queried object; a meta-learned selector over the reader's own embeddings (gold, two seeds) recovers 55–64% of those frames and plateaus below the retriever; every-k-th-frame selection (grey) is the reference. Reading more frames past four only adds distractors.

Complementary learning systems. The CLS account McClelland et al. (1995) pairs a fast hippocampal episodic store with a slow neocortical semantic store, linked by a consolidation phase. SlowFast-VGen Hong et al. (2025) applies the fast–slow split to video *generation*, the closest CLS-shaped neural model to ours; EM-LLM Fountas et al. (2025) grows episodic memory via Bayesian surprise and HippoRAG Gutiérrez et al. (2024) indexes text into a hippocampus-inspired retrieval graph, both non-parametric. HIPPOSTREAM realizes CLS in a streaming Video LLM with offline meta-training as the consolidation step.

Sink and fixed-state alternatives. Attention-sink methods keep anchor tokens and a sliding window of recent KV states (Xiao et al., 2024). TRecViT uses causal recurrence over video frames (Pătrăucean et al., 2024); Vision Mamba instead uses bidirectional state-space blocks for image representations (Zhu et al., 2024). Test-time training layers maintain an implicit weight memory through self-supervised gradient updates (Sun et al., 2025b). HIPPOSTREAM is closer to a small, per-video memory that a frozen backbone reads; the fast state is not the KV cache and not the model parameters.

Streaming Video-LLM baselines. Concurrent streaming methods differ in what they cache. ReKV (Di et al., 2025) keeps a sliding KV cache with retrieval on top. HERMES (Zhang et al., 2026) manages KV states across sensory, working, and long-term memory levels. SimpleStream (Shen et al., 2026) uses an off-the-shelf VLM with a sliding window of recent frames. StreamingVLM (Xu et al., 2026) aligns training on overlapping chunks with a bounded cache of attention sinks and recent visual and text tokens. LatentStream (Sep 2026) internalises retrieved evidence into a bounded latent memory optimised at question time by an entropy reward (Qu et al., 2026); it shares the frozen-backbone and bounded-memory contracts but couples the read to a per-question optimisation, whereas HIPPOSTREAM writes online during the stream and reads without further optimisation. We compare against LatentStream (Section 3.4) and reproduce SimpleStream's recent-window setting (Appendix K).

R COMPUTATIONAL COST

The per-chunk WRITE is one functional gradient step (K=1) on the M=4 memory bank, so its cost grows linearly with stream length and remains comfortably inside a single H100 budget on every OVO-Bench video we ran. The READ adds M extra tokens to the prompt, which is small next to

the recent-window vision tokens already present. On long Forward videos with many sub-instances, end-to-end walltime is dominated by the repeated per-chunk WRITE pass; meta-training the slow components completes in about 45 minutes on a single H200 (Section 3.1). An optional event-gated WRITE skips chunks whose trajectory curvature is below threshold, trading WRITE compute for a small amount of Bwd headroom.

S SCOPE AND FUTURE DIRECTIONS

We report the scope of the current study and the two directions that would naturally extend it, so that the reader can judge which regimes the meta-learned bank already covers and which are left for follow-up work.

Statistical scope. Most rows of Table 2 are single-seed on the 1640-sample OVO-Bench; the $+5.28$ Bwd-summary headline (Table 1) is averaged over three shuffle seeds and includes the standard deviation. Between-seed variability on the 50/50/50 subset is $\sim\pm3$ All, so single-cell ±1 gaps in Table 2 should be read as directional.

Meta-training corpus. The per-family readout in Section 3.2 shows that the bank is most useful on summary-type backward questions and less useful on recent-frame perception questions. This is consistent with the meta-training distribution: VSI-590K pairs each video with a single global question, so the writer has no supervision that distinguishes recent-frame queries (STU, FPD, OCR) from past-event queries (EPM, ASI, HLD). A streaming-aware meta-training corpus with per-family supervision is a natural next step and would let the writer learn to stay quiet on recent-frame queries.

Cross-scale transfer of the slow triple. The 8B rows use slow weights meta-trained at $d{=}2048$ and shape-skipped onto $d{=}3584$; re-training M_0, W_h, W_r directly at $d{=}3584$ did not further improve the $+0.20$ All lift, which we therefore attribute to the WRITE-time gradient and the first-position target rather than to the meta-learned init at 8B. Joint meta-training at the target hidden size is a straightforward extension.

LoRA-based backbone tuning. An exploratory backbone LoRA path (Phase B) under a backward-skewed replay lost 6–10 points on backward queries; we defer a balanced-replay schedule that mixes the three query families to future work.