LONG-TERM VISUAL UNDERSTANDING WITH LATENT TOKEN MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Multimodal Models (LMMs) now reason well over static images, but long visual streams still expose a basic systems problem: the model is asked to remember more pixels than its context can afford. Existing approaches either concatenate visual tokens, which grows attention cost with history length, or prune tokens statelessly, which discards evidence before the model knows whether it will matter. We introduce **MemVLM**, a fixed-budget visual memory mechanism that treats visual history as online state estimation. MemVLM uses a dual-head projector to separate immediate visual grounding from long-term writes, and a Gated Recurrent Memory Update (GRMU) to maintain a compact latent belief state. A Minimal Write objective makes memory updates sparse, turning long visual history into a stateful constant-memory representation rather than an ever-growing token buffer. Across long-context video benchmarks, dense-event stress tests, and matched trainable controls, MemVLM retains more temporal evidence than retrieval, streaming, and pruning baselines under the same token budget. On a newer LLaVA-Video-7B backbone, adding MemVLM improves Video-MME (Long) from 69.7 to 73.1 and LongVideoBench from 62.7 to 66.8 over a sliding-window variant of the same backbone, suggesting that the memory mechanism transfers beyond the original implementation backbone.

1 INTRODUCTION

The integration of Large Language Models (LLMs) (Radford et al., 2019; Touvron et al., 2023; Bai et al., 2023) with visual encoders has rapidly expanded Vision Language Models (VLMs) (Bordes et al., 2024; Li et al., 2022; 2023). By projecting visual features into a language-aligned space, models such as LLaVA (Liu et al., 2023a; Li et al., 2024) and GPT-4V (OpenAI, 2024) have moved from image-text alignment to instruction following and multimodal reasoning. Long visual streams, however, still break the default recipe. Multi-image documents, high-resolution imagery, and long videos generate visual token histories that cannot be naively concatenated into the context window. The result is both a systems bottleneck, from quadratic attention cost, and a reasoning bottleneck, from the "lost-in-the-middle" effect (Liu et al., 2023b).

To mitigate context limitations in LLMs, strategies such as sparse attention (Yuan et al., 2025; Child et al., 2019), recurrence (Qin et al., 2019), and retrieval-augmented generation (RAG) (Lewis et al., 2021) have been widely adopted. However, transferring these techniques to the visual domain is non-trivial due to the continuous and high-redundancy nature of visual representations. Prevalent VLM approaches often resort to aggressive down-sampling strategies—such as token merging (Shang et al., 2025; Islam & Rochan, 2025) or keyframe selection (Zhu et al., 2025)—to fit within the budget. While effective for simple tasks, these "lossy" compression methods inevitably discard fine-grained details necessary for precise grounding, effectively rendering the model "myopic" to dense visual documents or subtle video cues.

Our work draws inspiration from explicit memory mechanisms (Wu et al., 2025; Zhang et al., 2025a; Thoma et al., 2023) in NLP, but the visual setting changes the problem. A video is not just a long list of tokens; it is a stream with heavy redundancy, occasional high-value events, and a strict serving budget. We therefore ask a narrower question: can a VLM learn a fixed-budget latent state that preserves the task-relevant visual history while ignoring redundant frames?

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.

We introduce **MemVLM**, a framework for scalable multimodal reasoning with a fixed-size latent memory pool. MemVLM processes visual tokens, whether from video frames or multiple images, through a dual-head architecture. A *Read Head* handles immediate local grounding, while a *Write Head* distills each observation into compact memory updates. These updates are integrated by a Gated Recurrent Memory Update (GRMU). The Minimal Write Principle regularizes the gate activations, so the memory state is updated only when the stream contains informative evidence. The model is not a new Vision SSM backbone; it is a plug-in memory mechanism that gives a standard VLM a stateful, constant-memory visual history.

We validate MemVLM on LLaVA-1.5-7B (Liu et al., 2024) and Video-LLaVA (Lin et al., 2023) across image, video, and synthetic long-context benchmarks. Beyond the headline tables, we include the failure-mode tests that matter for this setting: dense-event videos, capacity-density scaling, a matched trainable VisPruner control, TBPTT-window ablations, and transfer to a newer LLaVA-Video-7B backbone. These tests show both sides of the method: fixed-budget memory remains lossy under extremely dense evidence, but it delays this failure mode more effectively than stateless pruning, retrieval, or sliding-window baselines.

Our contributions are threefold. **(i)** We formulate long visual understanding as fixed-budget online state estimation and introduce MemVLM, a plug-in memory mechanism that couples a budgeted dual-head projector with a gated recurrent memory update, so that the LLM-visible cache stays constant regardless of stream length. **(ii)** We derive a Minimal Write objective from a drift bound on the gated update, which turns memory writes into sparse, event-triggered operations and keeps gradient flow near identity through redundant segments. **(iii)** Under matched visual-token budgets, MemVLM retains 91.1% of oracle capability at 128 tokens versus 86.5% for the strongest pruning baseline, doubles VNBench accuracy with a $60\times$ smaller token footprint, and runs up to 10,000 frames on a single A100 where the concatenation baseline fails at about 220.

2 Related Work

Large Multimodal Models and Video Reasoning. The evolution of Large Multimodal Models (LMMs) (An et al., 2025; Bai et al., 2025b) tracks the progression from foundational visual encoders, such as ViT (Dosovitskiy et al., 2021) and contrastive learners like CLIP (Radford et al., 2021) or SigLIP (Zhai et al., 2023), toward deep semantic alignment. Early efforts like BLIP (Li et al., 2022) and BLIP-2 (Li et al., 2023) introduced bridging mechanisms to connect these visual representations with language decoders. This paradigm matured with architectures like LLaVA (Liu et al., 2023a) and Qwen-VL (Bai et al., 2023), which directly project visual tokens into the input space of Large Language Models, unlocking complex reasoning capabilities. However, extending these image-centric designs to the temporal domain remains challenging. The standard practice of concatenating frame sequences into a flat context window incurs prohibitive computational costs due to the quadratic complexity of self-attention. Consequently, while these models excel at static image tasks, they face severe scalability bottlenecks in long-form video understanding, where memory explosion prevents the retention of fine-grained details across extensive histories.

Long-Context Modeling, Latent Memory, and State-Space Models. Long-context language modeling has explored sparse attention, recurrence, explicit memory, and structured state-space models (Wu et al., 2025; Zhang et al., 2025a; Thoma et al., 2023; Gu et al., 2022; Gu & Dao, 2024; Dao & Gu, 2024). These methods share a common goal: avoid storing the full history while preserving the evidence needed for future prediction. MemVLM follows this fixed-state viewpoint, but it does not replace the visual backbone with a Vision SSM. Instead, it adds a recurrent latent memory to an existing LMM and uses visual-token gates to decide when the state should change. This distinction matters empirically: our comparisons target efficient long-context inference mechanisms under matched visual budgets, while recent backbones such as Qwen2.5-VL (Bai et al., 2025b), Qwen3-VL (Bai et al., 2025a), and LLaVA-OneVision (An et al., 2025) remain complementary foundations.

Token Pruning and Efficient VLMs. A prominent direction for efficient multimodal processing focuses on stateless token reduction strategies to fit visual inputs within a limited budget. Approaches such as Token Merging (ToMe) (Bolya et al., 2023) and heuristic pruning methods like FastV (Chen et al., 2024), VisPruner (Zhang et al., 2025b), and SparseVLM (Zhang et al., 2025c) aggressively down-sample visual tokens based on attention scores or similarity metrics. While these methods

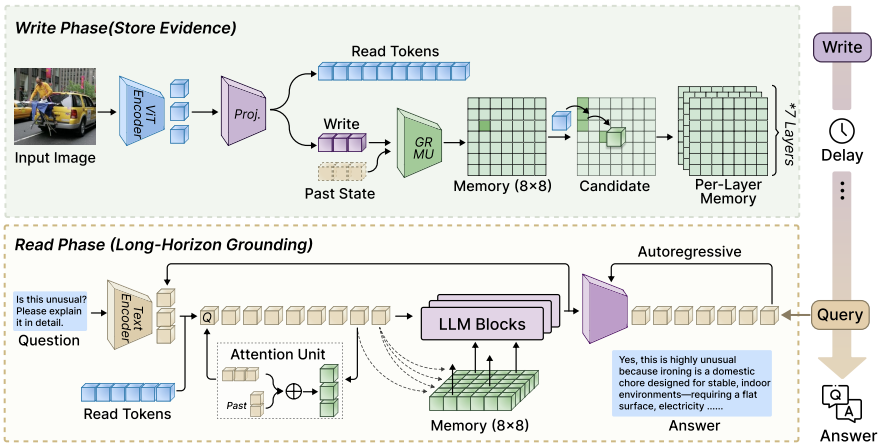


Figure 1: Overview of the MemVLM Framework. The architecture operates in two decoupled phases: **(Top) Write Phase:** Visual inputs are processed via a dual-head projector. The *Write Tokens* are distilled into a compact, fixed-size belief state (M_t) via the **Gated Recurrent Memory Update (GRMU)**, efficiently compressing history. **(Bottom) Read Phase:** At query time, the model performs long-horizon grounding by combining immediate evidence (*Read Tokens* prepended to text) with long-term context (historical memory states injected as read-only KV prefixes) to generate the final response.

(Wang et al., 2025; Anonymous, 2026) reduce computational costs, they operate on a "lossy" selection principle, inevitably discarding tokens that may appear locally redundant but contain critical semantic details for downstream reasoning (e.g., small objects or subtly distinct cues). In contrast to these selection-based paradigms, our framework employs a memory-based distillation approach, preserving semantic integrity in a continuous latent state rather than selecting a discrete subset of raw tokens.

3 METHOD

We introduce **MemVLM**, a framework designed for streaming multimodal reasoning under strict computational constraints. Unlike standard Large Multimodal Models (LMMs) that rely on buffering an ever-growing context of visual tokens, MemVLM learns a fixed-size latent belief state that acts as a sufficient statistic for the history. We instantiate our method using LLaVA-1.5 and Video-LLaVA as the backbone.

3.1 PROBLEM FORMULATION: VARIATIONAL ONLINE STATE ESTIMATION

Consider a continuous stream of visual observations $v_{1:t} = \{v_1, \dots, v_t\}$ and a query q_t at time t. The optimal response y_t is distributed according to the posterior $P(y_t \mid q_t, v_{1:t})$. Maintaining the full history $v_{1:t}$ incurs linear memory cost $\mathcal{O}(t)$ and quadratic attention cost $\mathcal{O}(t^2)$.

We formulate the problem as learning a recursive state update operator ϕ_θ that maps the previous state $\mathcal{M}_{t-1}$ and current observation v_t to a new state $\mathcal{M}_t$, subject to a capacity constraint $\dim(\mathcal{M}_t) \leq B \times D$, where B is the fixed number of memory slots.

ε-Sufficient Statistics. We aim to learn a state $\mathcal{M}_t$ that captures all task-relevant information. Formally, let $\mathcal{Q}$ be the family of admissible queries. We seek $\mathcal{M}_t$ such that it is ε-sufficient for $v_{1:t}$ with respect to $\mathcal{Q}$:

$$\sup_{q \in \mathcal{Q}} \mathrm{TV}(P(\cdot \mid q, v_{1:t}),\ P(\cdot \mid q, \mathcal{M}_t)) \leq \varepsilon, \tag{1}$$

where $\mathrm{TV}(P, Q)$ is the Total Variation distance.

3.2 ARCHITECTURE: BUDGETED DUAL-HEAD VISUAL PROJECTION

To reconcile high-fidelity immediate grounding with fixed-budget long-term storage, we employ a **Budgeted Dual-Head Projection** on frozen vision encoder features $H_t = \mathrm{Enc}(v_t) \in \mathbb{R}^{N \times D_v}$. Crucially, both heads expose a *controllable* number of visual tokens to the LLM.

Read Head (Budgeted Immediate Grounding). Instead of passing all N spatial tokens, we compress H_t into a small set of R *read tokens*, where R is a user-specified budget (e.g., $R \in \{32, 64, 128\}$) at inference time. We first project features to the LLM hidden size via $\bar{H}_t = \mathrm{MLP}(H_t) \in \mathbb{R}^{N\times D}$. We then apply a Perceiver-style resampler with R latent queries $\mathbf{Q}^{(R)}_{\mathrm{read}} \in \mathbb{R}^{R\times D}$:

$$\mathbf{X}^{\mathrm{read}}_t = \mathrm{MultiHeadAttn}\Big(\mathbf{Q}^{(R)}_{\mathrm{read}},\ \bar{H}_t,\ \bar{H}_t\Big) \in \mathbb{R}^{R\times D}. \tag{2}$$

This provides a *budget knob* for immediate visual grounding: increasing R improves per-frame fidelity, while keeping the LLM-visible tokens bounded.

Write Head (Fixed-Budget Memory Distillation). To distill observations into the long-term belief state, we generate K write tokens ($K \ll N$) using a separate resampler with K latent queries $\mathbf{Q}_{\mathrm{lat}} \in \mathbb{R}^{K\times D}$:

$$\mathbf{X}^{\mathrm{write}}_t = \mathrm{MultiHeadAttn}\big(\mathbf{Q}_{\mathrm{lat}},\ \bar{H}_t,\ \bar{H}_t\big) \in \mathbb{R}^{K\times D}. \tag{3}$$

These write tokens serve as the compressed "observation" for the memory update mechanism described below.

3.3 Gated Recurrent Memory Update (GRMU)

We maintain a set of layer-wise belief states $\mathcal{M}_t = \{\mathbf{M}^\ell_t\}_{\ell\in\mathcal{S}}$, where $\mathcal{S}$ denotes the subset of transformer layers augmented with memory (e.g., top-k layers). Each $\mathbf{M}^\ell_t \in \mathbb{R}^{B\times D}$ captures temporal dynamics at a specific semantic depth.

Parallel Update Mechanism. The update is computed outside the autoregressive decoding loop, avoiding the need to buffer historical visual tokens. At time t, for each layer $\ell \in \mathcal{S}$, the memory slots actively query the new visual write tokens via Multi-Head Cross-Attention (MHCA):

$$\mathbf{C}^\ell_t = \mathrm{MHCA}(Q = \mathbf{M}^\ell_{t-1}, K = \mathbf{X}^{\mathrm{write}}_t, V = \mathbf{X}^{\mathrm{write}}_t). \tag{4}$$

We then generate a candidate update state $\tilde{\mathbf{M}}^\ell_t$:

$$\tilde{\mathbf{M}}^\ell_t = \mathrm{FFN}(\mathrm{LN}(\mathbf{M}^\ell_{t-1} + \mathbf{C}^\ell_t)). \tag{5}$$

To control plasticity, we compute a scalar write gate $g^\ell_t[b] \in (0, 1)$ for each slot b, conditioned on the concatenated state and context:

$$\mathbf{g}^\ell_t = \sigma\left(\mathrm{MLP}_{\mathrm{gate}}([\mathbf{M}^\ell_{t-1}\,;\,\mathbf{C}^\ell_t])\right). \tag{6}$$

The memory is updated via soft interpolation, akin to a GRU cell:

$$\mathbf{M}^\ell_t = (1 - \mathbf{g}^\ell_t) \odot \mathbf{M}^\ell_{t-1} + \mathbf{g}^\ell_t \odot \tilde{\mathbf{M}}^\ell_t. \tag{7}$$

Memory-Conditioned Inference. At query time, we inject visual information into the LLM via two pathways: (1) **Immediate Context:** The read tokens $\mathbf{X}^{\mathrm{read}}_t$ are prepended to the text embeddings as standard input tokens. (2) **Long-term Context:** The updated memory states $\mathbf{M}^\ell_t$ are injected as *read-only prefixes* into the Key-Value cache of the corresponding transformer layers $\ell \in \mathcal{S}$. Importantly, MemVLM never concatenates historical frame tokens $v_{1:t-1}$ into the context, ensuring $\mathcal{O}(1)$ inference cost.

We define the total memory budget as $B_{\mathrm{tot}} = |\mathcal{S}|B$. Since the memory prefix is *stateful* (replaced at each step), the KV cache size remains bounded independent of the stream length t.

3.4 Theoretical Analysis: Stability and Forgetting

We analyze the stability properties to ensure our fixed-budget memory can retain information over $T \to \infty$.

Assumption 3.1 (Bounded Candidate Update)**.** For all ℓ, t, there exists a constant $\Gamma > 0$ such that the update magnitude is bounded: $\|\tilde{\mathbf{M}}^\ell_t - \mathbf{M}^\ell_{t-1}\|_F \le \Gamma$.

Proposition 3.2 (Drift Bound via Gate Sparsity)**.** *Under Assumption 3.1, the total drift of the memory state over a horizon T is upper-bounded by the cumulative L1-norm of the write gates.*

Proof sketch. The result follows by writing the one-step drift as $\Delta_t = \mathbf{g}_t^\ell \odot (\tilde{\mathbf{M}}_t^\ell - \mathbf{M}_{t-1}^\ell)$, applying Assumption 3.1, and summing the resulting per-step bound across the horizon. The full derivation is in Appendix C.2. **Remark (Stability vs. Sparsity):** While the tightest stability bound depends on the ℓ_∞-norm, we utilize the looser ℓ_1-bound for our objective function. Minimizing the ℓ_1-norm implicitly controls the ℓ_∞-norm while simultaneously promoting *sparsity*. This ensures the memory update is *event-triggered*—updating only specific slots when necessary.

Assumption 3.3 (First-Order Gradient Approximation)**.** Within the background segments (which constitute the majority of the stream), the gate values vary slowly with respect to the state magnitude, i.e., we neglect terms involving $\frac{\partial \mathbf{g}_t^\ell}{\partial \mathbf{M}_{t-1}^\ell}$.

Proposition 3.4 (Exponential Decay of Influence)**.** *Under Assumption 3.3, the gradient of the state at time t with respect to a past state at time $\tau < t$ satisfies:*

$$\frac{\partial \mathbf{M}_t^\ell}{\partial \mathbf{M}_\tau^\ell} \approx \prod_{k=\tau+1}^{t} \mathit{diag}(\mathbf{1} - \mathbf{g}_k^\ell). \tag{8}$$

Proof sketch. Differentiating Eq. (7) gives a dominant $\mathrm{diag}(\mathbf{1} - \mathbf{g}_t)$ term plus gate-scaled update terms. Under Assumption 3.3, inactive slots preserve near-identity gradient flow; chaining these Jacobians yields the product form. The full derivation is in Appendix C.2. Crucially, for inactive slots ($g_k^\ell[b] \approx 0$), the gradient flow remains close to identity, preventing the vanishing gradient problem common in long-horizon reasoning.

3.5 Objective: The Minimal Write Principle

We train MemVLM end-to-end. The primary loss is the autoregressive modeling of the answer y:

$$\mathcal{L}_{\text{task}} = -\mathbb{E}\left[\sum_{j=1}^{|y|} \log P(y_j \mid y_{<j}, q, \mathbf{X}_t^{\text{read}}, \{\mathbf{M}_t^\ell\})\right]. \tag{9}$$

Minimal Write Regularization. Guided by Proposition 3.2, we impose a regularization term on the gate magnitudes. We interpret this as minimizing the *Information Cost* of the update:

$$\mathcal{L}_{\text{reg}} = \frac{1}{|\mathcal{S}|B} \sum_{\ell \in \mathcal{S}} \sum_{t=1}^{T} \|\mathbf{g}_t^\ell\|_1. \tag{10}$$

The final objective is $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \lambda \mathcal{L}_{\text{reg}}$. This forces the model to treat memory updates as expensive operations, preserving the "sufficient statistic" state $\mathcal{M}_t$ against irrelevant visual noise. The full streaming inference procedure is provided in Appendix C.3; in brief, each incoming frame is passed through the read/write heads, the write tokens update the recurrent memory, and the final memory state is injected as read-only KV prefixes during generation.

4 Experiments

4.1 Main Results

Experimental Setup: Evaluating Capability Retention. We evaluate whether a fixed memory can preserve VLM capability under severe visual-token budgets. Table 1 covers eight image and video benchmarks and compares three efficient-inference families: streaming architectures such as SlowFast-LLaVA (Xu et al., 2024), retrieval methods such as VideoRAG (Ren et al., 2025), and pruning methods including VisPruner (Zhang et al., 2025b), PruMerge+ (Shang et al., 2024), and SparseVLM (Zhang et al., 2025c). This split is intentional: streaming baselines test recency-biased compression, retrieval baselines test query-conditioned frame selection, and pruning baselines test stateless token selection. Trainable baselines use the same Video-LLaVA-7B backbone, LLaVA-Video-178K data (Zhang et al., 2024), and budgets $\mathcal{B}_{vis} \in \{128, 64, 32\}$. We report retention relative to the uncompressed LLaVA-1.5-7B or Video-LLaVA oracle, so the metric asks how much original capability survives compression rather than how high each backbone can score in isolation.

Method	EgoSchema	Video-MME (Long)	Video-MME (All)	ActivityNet QA	LongVideo Bench	NExT-QA	MSRVTT-QA	MVBench	Avg. Ret.(%)
Upper Bound (Oracle)	56.2	52.5	58.4	54.8	48.6	68.2	72.5	55.1	100.0
Budget: $\mathcal{B}_{vis} = 128$ tokens									
SlowFast-LLaVA	45.2	41.5	48.5	49.2	38.5	62.5	66.2	47.5	85.1
VideoRAG	44.1	40.8	47.2	48.5	39.2	61.8	**68.5**	46.8	84.6
SparseVLM	43.8	40.5	47.1	47.5	38.2	60.5	64.2	45.1	83.2
PruMerge+	44.5	41.2	47.8	48.2	38.8	61.2	64.9	45.8	84.1
VisPruner	46.5	43.2	49.8	49.5	40.1	62.8	66.5	47.6	86.5
Sliding Window (Baseline)	38.5	34.2	42.1	44.5	33.8	58.2	64.5	43.2	77.8
MemVLM (Ours)	**48.8**	**45.6**	**52.1**	**51.8**	**42.5**	**65.1**	68.9	**50.2**	**91.1**
Budget: $\mathcal{B}_{vis} = 64$ tokens									
SlowFast-LLaVA	41.5	37.2	44.5	45.6	35.5	58.5	62.4	43.5	78.8
VideoRAG	40.8	36.5	43.8	45.1	36.1	57.2	64.2	42.8	77.5
SparseVLM	40.2	37.5	43.5	45.2	35.1	57.5	61.2	42.5	76.8
PruMerge+	41.5	38.2	44.8	45.8	36.2	58.2	62.1	43.1	78.5
VisPruner	43.2	40.1	46.5	46.8	37.2	59.5	63.2	44.5	80.8
Sliding Window (Baseline)	32.4	27.5	36.8	40.1	29.2	53.5	59.8	36.5	66.2
MemVLM (Ours)	**45.8**	**42.6**	**48.9**	**49.2**	**39.8**	**61.8**	**65.6**	**46.9**	**85.4**
Budget: $\mathcal{B}_{vis} = 32$ tokens									
SlowFast-LLaVA	35.2	31.5	38.8	40.2	30.5	52.8	56.5	38.1	69.2
VideoRAG	34.5	30.2	37.5	39.5	31.8	51.5	58.2	37.5	68.5
SparseVLM	35.8	32.5	38.2	40.5	30.5	52.4	56.1	37.2	69.5
PruMerge+	36.5	33.2	39.5	41.2	31.2	53.1	56.8	38.5	70.8
VisPruner	38.5	35.4	41.2	42.5	32.6	54.5	58.2	39.8	72.1
Sliding Window (Baseline)	24.1	19.8	28.5	34.2	21.5	44.8	53.2	28.8	52.4
MemVLM (Ours)	**40.9**	**37.8**	**43.6**	**45.1**	**35.2**	**56.8**	**60.5**	**42.1**	**76.5**

Table 1: Benchmarking Capability Retention against Advanced Baselines. We evaluate MemVLM against streaming models (SlowFast), retrieval-based methods (VideoRAG), and token pruning. MemVLM consistently outperforms these baselines, particularly in long-context tasks (e.g., Video-MME Long), maintaining high retention rates where RAG and Pruning methods degrade.

Implementation Details. We instantiate MemVLM on Video-LLaVA-7B and LLaVA-1.5-7B. The visual encoder is frozen, the LLM uses LoRA, and the Dual-Head Projector and GRMU are trained end-to-end. Unless stated otherwise, $B = 64$, $R = 64$, and $K = 32$. Training uses TBPTT with 8-frame chunks; the memory state $\mathcal{M}_t$ persists across chunk boundaries, so the model learns a stationary update policy rather than an 8-frame local smoother. This distinction is important: gradients are truncated for efficiency, but the forward state is not reset, and the updater is trained to operate over the full stream. VideoRAG uses the same frozen CLIP-ViT-L/14 features for retrieval, removing the visual encoder as a confounder.

Analysis: Superiority over RAG and Streaming Baselines. Table 1 shows that retrieval and pruning are competitive when the answer depends on a few salient frames, but degrade on long-horizon reasoning. VideoRAG nearly matches the oracle on retrieval-heavy MSRVTT-QA, yet drops sharply on Video-MME (Long), where evidence must be integrated across time. Sliding-window and SlowFast variants suffer from a different failure: they preserve recent or sparse context, but cannot keep older fine-grained evidence once the budget tightens. At 128 visual tokens, MemVLM improves average retention from 86.5% for VisPruner to 91.1%; the advantage remains at 64 and 32 tokens. The main gain is not more compression, but preserving a state that can be queried after the stream has passed.

4.2 The Needle in a Haystack Test

VNBench (Zhao et al., 2024) isolates the sampling dilemma with Retrieval, Ordering, and Counting tasks. We compare uniform sampling with 8 or 32 frames against MemVLM, which processes the stream frame-by-frame with a fixed 128-token visual budget on Video-LLaVA-7B (Lin et al., 2023).

Result Analysis. Table 2 shows that 32-frame Video-LLaVA uses 8,192 tokens yet reaches only 3.5% Ordering and 18.1% overall accuracy. MemVLM uses 128 tokens and reaches 18.2% Ordering, 29.5% Counting, and 35.4% overall accuracy, a roughly $2\times$ gain in overall accuracy with more than a $60\times$ smaller token footprint. The gains are largest on Ordering and Counting, the two tasks that require temporal state rather than single-frame recognition. Figure 2a shows the corresponding length trend: uniform sampling collapses as the haystack grows because the probability of sampling the relevant frame decreases, while MemVLM remains stable through the long-video regime.

Table 2: VNBench Results. We compare token efficiency and task accuracy. Metrics: **Ret.**(rieval), **Ord.**(ering), **Cnt.**(ounting). † denotes results reported in the original benchmark paper.

Model (Strategy)	**Tok.**	**VNBench Scores (%)** **Ret.**	**Ord.**	**Cnt.**	**Ov.**
Video-LLaVA† 8 fr. (Unif.)	2,048	23.8	1.1	12.4	12.4
Video-LLaVA 32 fr. (Dense)	8,192	35.2	3.5	15.8	18.1
MemVLM (Ours) Streaming	**128**	**58.4**	**18.2**	**29.5**	**35.4**
w/o Min. Write Streaming	128	45.1	10.3	20.1	25.1

Table 3: Dense-event ordering stress test. We keep the visual-token budget fixed and increase event density. Full Retrieval/Ordering/Counting results, capacity-density scaling, and protocol details are provided in Appendix C.1.

Method	**Low**	**Medium**	**High**
Sliding Window	13.2	5.4	1.8
VideoRAG	10.5	6.1	2.5
VisPruner	15.1	8.8	4.2
MemVLM (Ours)	**18.2**	**14.5**	**9.8**

4.3 Stress Tests and Matched Controls

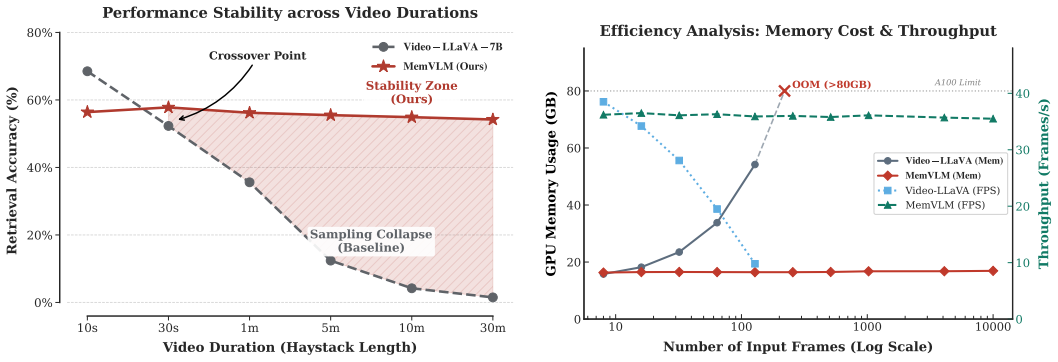
The hard case for fixed memory is a dense stream where many events matter. We vary event density while keeping the token budget fixed and report Ordering in Table 3. All methods degrade as density increases, but MemVLM retains 9.8% Ordering under high density, compared with 4.2% for VisPruner, 2.5% for VideoRAG, and 1.8% for a sliding window. Fixed memory is lossy; the evidence is that a recurrent latent state loses less temporal structure at the same budget.

We emphasize Ordering in the main text because it is the most direct probe of long-range state retention. A retrieval query can often be answered from a single salient frame, and a counting query can sometimes be approximated from coarse frequency statistics. Ordering is stricter: the model must preserve which events occurred, retain their relative positions, and answer after the relevant frames are no longer visible. This is where stateless compression is most exposed. A sliding window forgets old events by construction; retrieval can miss frames that are not individually similar to the final query; pruning commits to irreversible token removal before the future question is known. MemVLM does not make the fixed budget unlimited, but the recurrent state gives the model a chance to revise and protect evidence as the stream unfolds.

We also control for training data. A trainable VisPruner using the same Video-LLaVA-7B backbone, LLaVA-Video-178K data, and 128-token budget improves from 49.8 to 50.8 on Video-MME (All) and from 49.5 to 50.6 on ActivityNet-QA; MemVLM under the same setting reaches 52.1 and 51.8. This isolates the effect of the memory mechanism from the effect of video instruction tuning: matched finetuning helps pruning, but does not close the gap to stateful memory. On a newer LLaVA-Video-7B backbone (Zhang et al., 2024), MemVLM improves Video-MME (Long) from 69.7 to 73.1 and LongVideoBench from 62.7 to 66.8. We treat this as a transfer check, not a leaderboard claim. Full capacity, TBPTT, length, and gate statistics are in Appendix C.1.

Two additional checks clarify how to read these stress tests. First, the capacity-density sweep shows a monotonic improvement from $B = 32$ to $B = 128$ at every density level. This supports a simple interpretation: when dense streams fail, the bottleneck is not that GRMU stops updating correctly, but that the memory budget is smaller than the amount of evidence the task asks the model to preserve. Second, the TBPTT sweep shows that increasing the training window from 4 to 8 frames improves both VNBench and Video-MME (Long), while 16 frames gives a smaller gain at higher cost. We therefore use 8 frames as a practical training point, and rely on the persistent memory state, rather than full backpropagation through every frame, to carry information over long horizons.

Finally, the extended length scaling in Appendix C.1 shows a gradual degradation rather than an abrupt collapse: performance is stable through 10 minutes and drops more visibly at 30 and 60 minutes. This is exactly the behavior expected from a fixed-state representation. MemVLM is not



(a) VNBench stability. **(b)** Scalability on A100-80GB.

Figure 2: Long-context stability and scalability. (a) Video-LLaVA degrades as video duration increases, while MemVLM maintains robust retrieval accuracy. (b) Standard VLM inference triggers an **OOM** error at ≈220 frames, whereas MemVLM maintains constant memory and stable throughput up to 10,000 frames.

lossless; it shifts the failure boundary by preserving a compact belief state instead of repeatedly resampling or pruning the stream from scratch.

4.4 Efficiency and Scalability Analysis

We stress test scalability on one NVIDIA A100-80GB by increasing video length from 8 to 10,000 frames. We compare peak GPU memory and inference throughput between MemVLM with a 128-token memory budget and standard Video-LLaVA with full causal attention over visual tokens. This setting directly measures the systems consequence of the design: concatenation grows the KV cache with the stream, while MemVLM replaces the historical prefix with a fixed-size state.

Figure 2b shows the expected systems failure: Video-LLaVA reaches OOM at about 220 frames and its throughput drops from ≈38 FPS to ≈9 FPS as attention is computed over an expanding visual history. MemVLM keeps memory near 16.5GB and throughput near 36 FPS up to 10,000 frames. This is the practical benefit of the recurrent latent state: the model can continue reading the stream without paying a larger memory bill for every past frame.

4.5 Ablation Studies and Visualization

We ablate three design choices on VNBench: the Dual-Head architecture, the GRMU update rule, and the Minimal Write regularizer. The ablations are deliberately simple; each replaces the proposed component with a common alternative that preserves the same high-level budget.

Table 4: Ablation Analysis on VNBench. We report accuracy (%) on Retrieval (Ret.), Ordering (Ord.), and Counting (Cnt.) tasks using the 128-token budget setting. The default MemVLM configuration is highlighted in gray.

Model Variant	Change	Ret.	Ord.	Cnt.	Overall
MemVLM (Default)	—	**58.4**	**18.2**	**29.5**	**35.4**
Architecture Design					
(a) Shared Head	Dual → Single	52.1	14.5	26.8	31.1 (-4.3)
Update Mechanism					
(b) FIFO Queue	$M_t \leftarrow [M_{t-1}[1:], x_t]$	48.5	14.2	22.1	28.3 (-7.1)
(c) Average Pooling	$M_t \leftarrow \text{Mean}(M_{t-1}, x_t)$	41.2	5.6	18.4	21.7 (-13.7)
(d) Vanilla RNN	w/o Gate ($g_t \equiv 1$)	46.8	9.4	21.3	25.8 (-9.6)
(e) Momentum	$M_t \leftarrow 0.9M_{t-1} + \ldots$	49.5	11.2	23.0	27.9 (-7.5)
Objective Function					
(f) w/o Sparsity	$\lambda_{reg} = 0$	54.3	15.1	27.6	32.3 (-3.1)
(g) High Sparsity	$\lambda_{reg} = 1.0$	38.6	4.2	15.9	19.6 (-15.8)

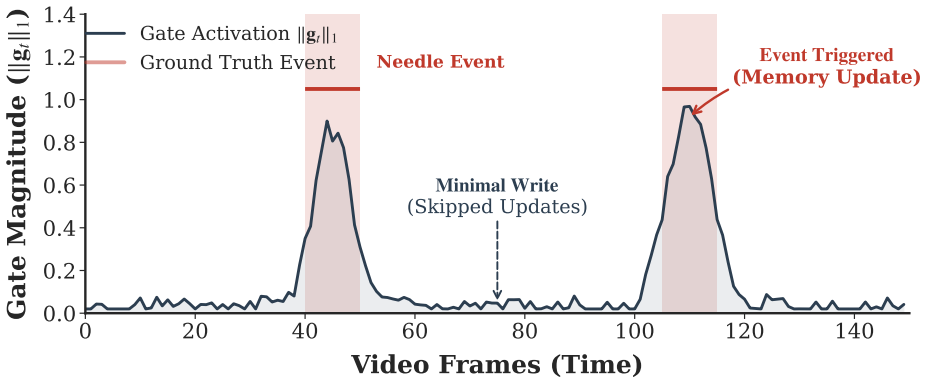


Figure 3: Visualization of Memory Gates. We visualize the gate activation norm $\|\mathbf{g}_t\|_1$ over a VNBench sequence. The gates stay suppressed during repetitive segments and spike when semantically novel "needle" events appear.

Effect of Dual-Head Projection. Table 4 shows that sharing the read/write projector costs 4.3 points. This supports the design choice that immediate grounding and long-term storage should not compete for the same projection: the read path must preserve high-frequency visual details for the current query, while the write path should distill the observation into update-friendly memory evidence.

Necessity of Gated Updates. Replacing GRMU with FIFO, average pooling, vanilla RNN, or momentum updates loses 7.1 to 13.7 points. FIFO behaves like a latent sliding window and fails when relevance is not aligned with recency. Average pooling performs worst because it washes rare events into the background. The learned gate is therefore not just a parameter increase; it is the mechanism that lets the model protect old evidence and selectively admit new evidence.

Impact of the Minimal Write Principle. Removing Minimal Write regularization loses 3.1 points, while excessive sparsity loses 15.8 points. The first result shows that an always-open memory gate admits redundant frames and overwrites useful state; the second shows that the information bottleneck cannot be made arbitrarily tight. The default setting is the empirical middle ground between noise rejection and event retention.

Qualitative Analysis: Gate as an Information Detector. Figure 3 visualizes the update gate over time. The gate remains near zero on redundant background frames and spikes around target events, matching the Minimal Write interpretation. This also clarifies the role of the local gradient assumption in Section 3: the slow-gate approximation is intended for inactive background segments, not for the sparse event frames where the gate should change sharply. Appendix C.1 reports the corresponding gate statistics.

5 Conclusion

MemVLM reframes long visual understanding as fixed-budget state estimation. A dual-head projector separates immediate grounding from long-term writes, and GRMU maintains a sparse latent memory that can be queried after the stream has passed. Across standard benchmarks, VNBench, dense-event stress tests, matched pruning controls, and a newer-backbone transfer check, MemVLM preserves more temporal evidence than stateless compression at the same budget.

The larger point is that long visual context should not be treated as an ever-growing token list. It should be treated as a state that is continuously revised, protected when the stream is redundant, and updated when new evidence appears. This does not remove the limits of fixed memory, and our dense-event analysis makes that bottleneck explicit. It does, however, give a scalable operating point for long-form multimodal understanding: remember what matters, keep the cache bounded, and let the visual history survive after the frames are gone.

This view also separates the proposed mechanism from a new backbone claim. MemVLM is meant to sit on top of existing VLMs, inherit their recognition and language ability, and add a bounded memory path for long streams. The newer-backbone transfer result supports this role, while the dense-event results make clear where larger or adaptive memory budgets will still be needed.

AI USE STATEMENT

Large language models were used as writing assistants for language polishing, LaTeX formatting, and conversion of the manuscript to the ICLR template. They were not used to design the method, run experiments, or generate results. All AI-assisted text was reviewed and edited by the authors, who take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

Section 3 specifies the architecture, the GRMU update rule, and the Minimal Write objective; Section 4.1 lists the backbones, token budgets, baselines, and benchmarks; Appendix B summarizes the backbone, frozen and trainable modules, training data, token budgets, TBPTT setting, baseline controls, compute, and evaluation protocol; Appendix C.1 provides the additional stress tests and controls, and Appendix C.2 contains the proofs. We will release the code and trained memory modules upon publication.

REFERENCES

Xiang An, Yin Xie, Kaicheng Yang, Wenkang Zhang, Xiuwei Zhao, Zheng Cheng, Yirui Wang, Songcen Xu, Changrui Chen, Chunsheng Wu, Huajie Tan, Chunyuan Li, Jing Yang, Jie Yu, Xiyao Wang, Bin Qin, Yumeng Wang, Zizhen Yan, Ziyong Feng, Ziwei Liu, Bo Li, and Jiankang Deng. Llava-onevision-1.5: Fully open framework for democratized multimodal training. In *arXiv*, 2025.

Anonymous. S-ATM: Self-boosting visual reasoning via adaptive token merging, 2026. URL https://openreview.net/forum?id=YpPMb1tC5N.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report, 2023. URL https://arxiv.org/abs/2309.16609.

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, 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, 2025b. URL https://arxiv.org/abs/2502.13923.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster, 2023. URL https://arxiv.org/abs/2210.09461.

Florian Bordes, Richard Yuanzhe Pang, Anurag Ajay, Alexander C. Li, Adrien Bardes, Suzanne Petryk, Oscar Mañas, Zhiqiu Lin, Anas Mahmoud, Bargav Jayaraman, Mark Ibrahim, Melissa Hall, Yunyang Xiong, Jonathan Lebensold, Candace Ross, Srihari Jayakumar, Chuan Guo, Diane Bouchacourt, Haider Al-Tahan, Karthik Padthe, Vasu Sharma, Hu Xu, Xiaoqing Ellen Tan, Megan

Richards, Samuel Lavoie, Pietro Astolfi, Reyhane Askari Hemmat, Jun Chen, Kushal Tirumala, Rim Assouel, Mazda Moayeri, Arjang Talattof, Kamalika Chaudhuri, Zechun Liu, Xilun Chen, Quentin Garrido, Karen Ullrich, Aishwarya Agrawal, Kate Saenko, Asli Celikyilmaz, and Vikas Chandra. An introduction to vision-language modeling, 2024. URL https://arxiv.org/abs/2405.17247.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models, 2024. URL https://arxiv.org/abs/2403.06764.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *URL https://openai.com/blog/sparse-transformers*, 2019.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality, 2024. URL https://arxiv.org/abs/2405.21060.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL https://arxiv.org/abs/2312.00752.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces, 2022. URL https://arxiv.org/abs/2111.00396.

Zahidul Islam and Mrigank Rochan. AdaptMerge: Inference time adaptive visual and language-guided token merging for efficient large multimodal models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 7352–7361, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.387. URL https://aclanthology.org/2025.findings-emnlp.387/.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021. URL https://arxiv.org/abs/2005.11401.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer, 2024. URL https://arxiv.org/abs/2408.03326.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation, 2022. URL https://arxiv.org/abs/2201.12086.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023. URL https://arxiv.org/abs/2301.12597.

Bin Lin, Bin Zhu, Yang Ye, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*, 2023.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023a. URL https://arxiv.org/abs/2304.08485.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning, 2024. URL https://arxiv.org/abs/2310.03744.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023b. URL https://arxiv.org/abs/2307.03172.

OpenAI. Gpt-4o system card, 2024. URL https://arxiv.org/abs/2410.21276.

Shenghao Qin, Jiacheng Zhu, Jimmy Qin, Wenshuo Wang, and Ding Zhao. Recurrent attentive neural process for sequential data, 2019. URL https://arxiv.org/abs/1910.09323.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Xubin Ren, Lingrui Xu, Long Xia, Shuaiqiang Wang, Dawei Yin, and Chao Huang. Videorag: Retrieval-augmented generation with extreme long-context videos, 2025. URL https://arxiv.org/abs/2502.01549.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models, 2024. URL https://arxiv.org/abs/2403.15388.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. *ICCV*, 2025.

Lukas Thoma, Ivonne Weyers, Erion Çano, Stefan Schweter, Jutta L Mueller, and Benjamin Roth. CogMemLM: Human-like memory mechanisms improve performance and cognitive plausibility of LLMs. In Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell (eds.), *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pp. 180–185, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.conll-babylm.15. URL https://aclanthology.org/2023.conll-babylm.15/.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Zhenhailong Wang, Senthil Purushwalkam, Caiming Xiong, Silvio Savarese, Heng Ji, and Ran Xu. Dymu: Dynamic merging and virtual unmerging for efficient vlms, 2025. URL https://arxiv.org/abs/2504.17040.

Yaxiong Wu, Sheng Liang, Chen Zhang, Yichao Wang, Yongyue Zhang, Huifeng Guo, Ruiming Tang, and Yong Liu. From human memory to ai memory: A survey on memory mechanisms in the era of llms, 2025. URL https://arxiv.org/abs/2504.15965.

Mingze Xu, Mingfei Gao, Zhe Gan, Hong-You Chen, Zhengfeng Lai, Haiming Gang, Kai Kang, and Afshin Dehghan. Slowfast-llava: A strong training-free baseline for video large language models, 2024. URL https://arxiv.org/abs/2407.15841.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention, 2025. URL https://arxiv.org/abs/2502.11089.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training, 2023. URL https://arxiv.org/abs/2303.15343.

Dianxing Zhang, Wendong Li, Kani Song, Jiaye Lu, Gang Li, Liuchun Yang, and Sheng Li. Memory in large language models: Mechanisms, evaluation and evolution, 2025a. URL https://arxiv.org/abs/2509.18868.

Qizhe Zhang, Aosong Cheng, Ming Lu, Renrui Zhang, Zhiyong Zhuo, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. Beyond text-visual attention: Exploiting visual cues for effective token pruning in vlms, 2025b. URL https://arxiv.org/abs/2412.01818.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. Sparsevlm: Visual token sparsification for efficient vision-language model inference, 2025c. URL https://arxiv.org/abs/2410.04417.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. Video instruction tuning with synthetic data, 2024. URL https://arxiv.org/abs/2410.02713.

Zijia Zhao, Haoyu Lu, Yuqi Huo, Yifan Du, Tongtian Yue, Longteng Guo, Bingning Wang, Weipeng Chen, and Jing Liu. Needle in a video haystack: A scalable synthetic framework for benchmarking video mllms. *arXiv preprint*, 2024.

Zirui Zhu, Hailun Xu, Yang Luo, Yong Liu, Kanchan Sarkar, Zhenheng Yang, and Yang You. Focus: Efficient keyframe selection for long video understanding, 2025. URL https://arxiv.org/abs/2510.27280.

A Limitations and Future Work

Fixed memory is lossy. MemVLM replaces an ever-growing visual-token history with a fixed-size latent state. This is the intended design, but it is also the main limitation: when the stream contains many fine-grained events with little redundancy, the memory state must compress or discard some evidence. The dense-event study in Appendix C.1 makes this explicit. MemVLM degrades more slowly than sliding-window, retrieval, and pruning baselines, but it does not remove the information bottleneck.

Capacity should match event density. The default memory budget is chosen as a practical operating point, not as a universal optimum. Increasing the memory budget improves Ordering across low-, medium-, and high-density settings, suggesting that the failure mode is better understood as a capacity-demand mismatch than as an abrupt failure of GRMU. In deployment, streams with dense state changes should use a larger memory budget or an adaptive budget policy.

Serving integration is still engineering work. The algorithm keeps the memory prefix bounded, but efficient integration into production inference engines may require custom KV-prefix replacement rather than ordinary append-only caching. Our experiments validate the algorithmic memory profile and A100 scalability; production systems such as batched video serving would still need careful cache management, batching, and kernel-level optimization.

Scope. The current implementation focuses on visual-only streams and evaluates open VLM backbones. Audio-visual synchronization, multilingual video QA, and stronger recent backbones are natural next steps. We include a newer LLaVA-Video-7B transfer check, but do not claim that the current results exhaust the design space of modern VLMs.

B Reproducibility, Compute, and Responsible Use

Reproducibility scope. Table 5 summarizes the settings needed to reproduce the main claims. The central comparison is budget-controlled: all efficient methods are evaluated under the same visual-token budgets, and the trainable controls use the same backbone and LLaVA-Video-178K training data. The architecture is specified in Section 3, while Algorithm 1 gives the streaming inference procedure.

Compute. The scalability experiment is run on a single NVIDIA A100-80GB GPU and reports both peak memory and throughput. The TBPTT ablation reports relative training cost because the exact wall-clock time depends on cluster scheduling and data-loading throughput. All matched-control comparisons are designed so that the variable under test is the memory/pruning mechanism rather than the backbone, training data, or visual-token budget.

Table 5: Protocol summary for reproducibility. We list the choices that affect the main claims and the matched-control experiments.

Item	Setting
Backbones	LLaVA-1.5-7B and Video-LLaVA-7B for the main experiments; LLaVA-Video-7B for the newer-backbone transfer check.
Vision encoder	Frozen CLIP-ViT-L/14-336px features for MemVLM and the matched retrieval baseline.
Trainable modules	Dual-Head Projector and GRMU are trained end-to-end; the LLM uses LoRA; the visual encoder is frozen.
Training data	LLaVA-Video-178K is used for MemVLM and matched trainable baselines.
Default budgets	Memory capacity $B = 64$, read budget $R = 64$, write budget $K = 32$ unless otherwise stated; budget sweeps use $\mathcal{B}_{vis} \in \{128, 64, 32\}$.
Sequence training	TBPTT uses 8-frame chunks by default; the latent memory state persists across chunks.
Baseline controls	SlowFast and VideoRAG are re-implemented on the same Video-LLaVA-7B backbone; VisPruner is additionally evaluated as a matched trainable control.
Evaluation	Main benchmarks report accuracy or benchmark-native score; VNBench reports Retrieval, Ordering, Counting, and Overall accuracy; retention is normalized to the uncompressed oracle.

Statistical reporting. The current tables report single benchmark scores rather than confidence intervals. This is common for expensive VLM evaluation, but it limits statistical interpretation. We therefore avoid claims based on small differences alone and focus on repeated patterns across budget sweeps, dense-event regimes, matched controls, and transfer checks. Repeated-seed evaluation for the smaller VNBench stress tests is left to future work.

Broader impact. MemVLM is an efficiency method for long-context multimodal reasoning. Positive uses include lower memory cost for long-video analysis, document understanding, and accessibility tools that need to process long streams. Potential risks follow from the same capability: more efficient long-video understanding may also make surveillance-style analysis cheaper. The paper does not release a new model or dataset, and any future model release should include usage guidance consistent with the licenses and policies of the underlying VLM backbones and datasets.

Existing assets. We use existing model backbones, training data, and public benchmarks, and cite their original papers in the main text. We do not redistribute these assets in the manuscript. Our code release will include an asset manifest listing each dataset/model source, version, license or terms of use, and any redistribution restrictions.

C Supplementary Material

C.1 Additional Stress Tests and Controls

This section provides the full results behind the stress-test discussion in Section 4.3. We include the complete dense-event metrics, the capacity-density sweep, a matched trainable VisPruner control, the newer-backbone transfer check, TBPTT-window sensitivity, extended length scaling, and gate statistics. These experiments are designed to answer failure-mode questions rather than to introduce a new benchmark suite.

Dense-event stress test. Table 6 reports all three VNBench metrics as event density increases. The default memory budget is fixed at $B = 128$ visual tokens. Retrieval remains the most stable metric, while Ordering is the first metric to expose capacity pressure.

Table 6: Dense-event VNBench stress test. We increase event density while keeping the MemVLM memory budget fixed.

Event Density	Retrieval	Ordering	Counting
Low	58.4	18.2	29.5
Medium	52.1	14.5	24.8
High	43.6	9.8	18.2

Table 7 gives the matched comparison used in the main text. All methods use the same dense-event protocol and visual-token budget. The absolute drop under high density should be expected: the input contains more relevant events than a fixed memory can represent without loss. The comparison shows that the drop is not specific to MemVLM; it appears across sliding-window, retrieval, and pruning baselines.

Table 7: Matched dense-event comparison on Ordering. MemVLM preserves the strongest temporal-order signal across density regimes.

Method	Low Density	Medium Density	High Density
Sliding Window	13.2	5.4	1.8
VideoRAG	10.5	6.1	2.5
VisPruner	15.1	8.8	4.2
MemVLM (Ours)	**18.2**	**14.5**	**9.8**

Capacity-density analysis. Table 8 separates the failure mode from the update mechanism. Increasing the memory budget improves Ordering at every density level, with the largest absolute gains in the harder regimes. This supports the interpretation that dense-event degradation is a capacity-demand mismatch under fixed memory, not an abrupt failure of the GRMU updater.

Table 8: Capacity-density sweep on Ordering. Larger memory budgets consistently improve robustness in dense streams.

Memory Budget	Low Density	Medium Density	High Density
$B = 32$	11.8	7.2	3.5
$B = 64$	15.6	10.8	6.2
$B = 128$ (default)	**18.2**	**14.5**	**9.8**

Matched trainable VisPruner control. Table 9 controls for training data. We compare the original training-free VisPruner, a trainable VisPruner variant fine-tuned on the same LLaVA-Video-178K data, and MemVLM trained under the same backbone and budget. Matched fine-tuning improves VisPruner, but MemVLM remains stronger, indicating that the recurrent memory architecture contributes beyond the data.

Table 9: Matched VisPruner control. All matched rows use the same Video-LLaVA-7B backbone and 128-token visual budget.

Method	Video-MME (All)	ActivityNet-QA	Avg. Ret.
VisPruner (training-free)	49.8	49.5	86.5
VisPruner + matched finetuning	50.8	50.6	88.3
MemVLM (ours)	**52.1**	**51.8**	**91.1**

Newer-backbone transfer. Table 10 reports the transfer check summarized in the main text. We attach the same GRMU and dual-head memory mechanism to a newer LLaVA-Video-7B backbone. The gain is measured against a sliding-window variant of the same backbone.

Table 10: Transfer to a newer LLaVA-Video-7B backbone. The memory mechanism improves long-context scores on a stronger base model.

Backbone and Memory	Video-MME (Long)	LongVideoBench
LLaVA-Video-7B + Sliding Window	69.7	62.7
LLaVA-Video-7B + MemVLM	**73.1**	**66.8**

TBPTT-window sensitivity. Table 11 studies the chunk size used for truncated backpropagation through time. The memory state persists across chunks in all settings; the chunk size controls how far gradients are backpropagated before truncation. A window of 8 frames gives a practical trade-off, while 16 improves accuracy modestly at higher training cost.

Extended length scaling. Table 12 extends the VNBench length stress test beyond the main evaluation range while keeping the memory budget and needle density fixed. We do not observe an abrupt collapse within the tested range. Instead, performance is stable through 10 minutes and then decays gradually at 30 and 60 minutes.

Table 11: TBPTT chunk-size ablation. The default chunk size is 8 frames.

Chunk Size	VNBench Ret.	Video-MME (Long)	Training Cost
4	56.7	42.3	Low
8	58.4	43.6	Medium
16	60.1	45.2	High

Table 12: Length scaling under fixed memory. Relative drop is measured against the 30-second setting.

Duration	Overall Accuracy	Relative Drop vs. 30s
10s	34.8	-1.7%
30s	35.4	0.0%
1m	35.2	-0.6%
5m	34.8	-1.7%
10m	34.5	-2.5%
30m	33.2	-6.2%
60m	31.5	-11.0%

Gate statistics. Table 13 quantifies the qualitative pattern shown in Figure 3. Background segments have small gate magnitude and small temporal variation, matching the local regime used in Assumption 3.3. Event segments are intentionally outside this inactive regime: their gates spike because the memory should be updated.

Table 13: Gate statistics by segment type. Event frames trigger larger gate magnitude and temporal variation than background frames.

Segment Type	Avg. Gate Magnitude $\|\mathbf{g}_t\|$	Avg. Temporal Variation $\|\mathbf{g}_t - \mathbf{g}_{t-1}\|$
Background segments	$\sim$0.04	$\sim$0.01
Event segments	$\sim$0.85	$\sim$0.25

C.2 PROOFS

Proof of Proposition 3.2. Let $\Delta_t = \mathbf{M}_t^\ell - \mathbf{M}_{t-1}^\ell$. From Eq. (7), $\Delta_t = \mathbf{g}_t^\ell \odot (\tilde{\mathbf{M}}_t^\ell - \mathbf{M}_{t-1}^\ell)$. Using the sub-multiplicative property $\|\mathbf{x} \odot \mathbf{y}\|_F \leq \|\mathbf{x}\|_\infty \|\mathbf{y}\|_F$, we have:

$$\|\Delta_t\|_F \leq \|\mathbf{g}_t^\ell\|_\infty \Gamma. \tag{11}$$

Relaxing $\|\mathbf{g}\|_\infty \leq \|\mathbf{g}\|_1$, the total drift is bounded by:

$$\|\mathbf{M}_{t+T}^\ell - \mathbf{M}_t^\ell\|_F \leq \sum_{i=1}^{T} \|\Delta_{t+i}\|_F \leq \Gamma \sum_{i=1}^{T} \|\mathbf{g}_{t+i}^\ell\|_1. \tag{12}$$

Proof of Proposition 3.4. Differentiating the update rule Eq. (7) with respect to $\mathbf{M}_{t-1}^\ell$ gives

$$\frac{\partial \mathbf{M}_t}{\partial \mathbf{M}_{t-1}} = \operatorname{diag}(1 - \mathbf{g}_t) + \mathbf{g}_t \odot \frac{\partial \tilde{\mathbf{M}}_t}{\partial \mathbf{M}_{t-1}} + \underbrace{\frac{\partial \mathbf{g}_t}{\partial \mathbf{M}_{t-1}} \odot (\tilde{\mathbf{M}}_t - \mathbf{M}_{t-1})}_{\approx 0 \text{ via Assump. 3.3}} \tag{13}$$

For sparse updates where $g_t[b] \approx 0$, the second term vanishes because it is scaled by $\mathbf{g}_t$, leaving $\frac{\partial \mathbf{M}_t}{\partial \mathbf{M}_{t-1}} \approx I$. Applying the chain rule across steps yields

$$\frac{\partial \mathbf{M}_t^\ell}{\partial \mathbf{M}_\tau^\ell} \approx \prod_{k=\tau+1}^{t} \operatorname{diag}(\mathbf{1} - \mathbf{g}_k^\ell). \tag{14}$$

C.3 STREAMING INFERENCE PROCEDURE

Algorithm 1 MemVLM Streaming Inference

Require: Video stream $v_{1:T}$, query q_{text}, initial memory $\mathcal{M}_0 = \{M_0^l\}_{l \in \mathcal{S}}$
Require: Budgets: read tokens R, write tokens K
Initialize: $X_{context} \leftarrow \emptyset$
for $t = 1$ to T **do**
  $H_t \leftarrow \text{VisionEncoder}(v_t)$
  $X_t^{read} \leftarrow \text{ReadHead}(H_t, R)$
  $X_t^{write} \leftarrow \text{WriteHead}(H_t, K)$
  for each memory layer $l \in \mathcal{S}$ **do**
    $C_t^l \leftarrow \text{MHCA}(Q = M_{t-1}^l, K = X_t^{write}, V = X_t^{write})$
    $\tilde{M}_t^l \leftarrow \text{FFN}(\text{LN}(M_{t-1}^l + C_t^l))$
    $g_t^l \leftarrow \sigma(\text{MLP}([M_{t-1}^l; C_t^l]))$
    $M_t^l \leftarrow (1 - g_t^l) \odot M_{t-1}^l + g_t^l \odot \tilde{M}_t^l$
  end for
  $X_{context} \leftarrow X_t^{read}$ {only current read tokens; no history buffer}
end for
Inject M_T^l into the LLM KV cache as read-only prefixes.
$y \leftarrow \text{LLM.generate}(X_{context}, q_{text})$
return y