MEMORY AS WEIGHTS: INTERNALIZING LONG-TERM HISTORY FOR STREAMING VIDEOS

Anonymous authors
Paper under double-blind review

ABSTRACT

Streaming vision-language models (VLMs) must continuously integrate information from the past while remaining responsive to current observations and user queries. Existing memory mechanisms typically compress, retrieve, or cache historical observations and reintroduce them into the input context, forcing historical memory and current observations to compete for limited context. We propose *MaLoW*, a streaming memory framework that instead internalizes historical context into evolving lightweight parameter updates. MaLoW uses a recurrent HyperNetwork to incrementally consolidate incoming video observations into fixed-size LoRA weights, allowing historical information to be carried forward without revisiting the full video stream. In doing so, MaLoW represents *history as weights, while keeping the current observation as input*. To support training, we further construct *MaLoW-Instruct*, a large-scale streaming video dataset containing 381K question-answer pairs over 56K videos. We conduct extensive experiments to demonstrate that MaLoW achieves strong performance on OVO-Bench, StreamingBench, and StreamArena. On the episodic memory task of OVO-Bench, MaLoW outperforms prior approaches by 3.3%–6.7% across different video backbones while incurring only modest latency overhead. Beyond streaming video understanding, MaLoW also generalizes to vision-language-action models, improving the action success rate on Simpler WidowX by 9.3%.

1 INTRODUCTION

By continuously processing an input video stream and responding to user queries, a streaming vision-language model (VLM) must reason over what has already happened, interpret what is happening now, and remain attentive to what may happen next. Such capabilities are central to key tasks ranging from streaming video understanding (Niu et al., 2025) to interactive AI assistants (Lab, 2026) and robotic planning (Li et al., 2025), where responses and decisions must be made online under latency and memory constraints. Streaming VLMs have thus emerged as an important research frontier. Recent work has shown that offline VLMs can be adapted to describe ongoing activities in streaming videos (Shen et al., 2026; Lu et al., 2026), while instruction fine-tuning can further enable models to monitor future events (Li et al., 2026; Wang et al., 2026).

Despite these efforts, *a fundamental open problem remains: how should a streaming VLM remember?* As in the offline setting, an effective memory mechanism must preserve important visual information within a bounded memory footprint. The streaming setting, however, introduces additional challenges. *First*, memory must evolve incrementally as new observations arrive, rather than being repeatedly recomputed from the full history. *Second*, historical information must be incorporated without crowding out the current observation (see Fig. 1-2). *Finally*, a practical memory mechanism should introduce minimal inference overhead.

To address these challenges, we propose **MaLoW** (Memory as LoRA Weights), a framework that internalizes historical context into evolving lightweight parameter updates of a streaming VLM, as illustrated in Fig. 1-1. MaLoW incrementally consolidates incoming observations into fix-sized LoRA weights, allowing accumulated history to be carried forward without revisiting the full video stream. More fundamentally, rather than placing memory and current observations in competition within the same context window, as in prior work (Guan et al., 2026; Liu et al., 2026; Zhang et al., 2026a), *MaLoW represents history as weights while keeping the current observation as input*. At

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.

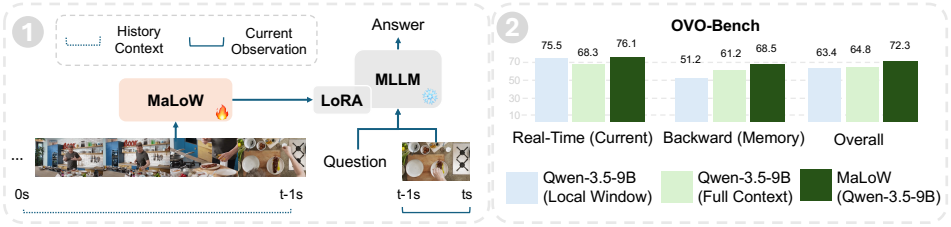


Figure 1: **(1)**: **MaLoW** internalizes the history context into time-evolving, lightweight parameter updates (LoRA) of a streaming VLM. At each step, the streaming VLM takes the current observation and user query as input, and performs inference using the updated LoRA weights. **(2)**: By separating memory from the input, MaLoW preserves the performance on real-time perception while significantly improves the performance of memory-critical, backward tracing tasks.

each step, the model takes the current observation and user query as input and performs inference using the updated LoRA weights that encode historical information. By separating historical memory from the input context, MaLoW preserves the model's capacity to attend to current observations while retaining information from the past (Fig. 1-2), with only modest inference overhead.

Specifically, MaLoW employs a recurrent HyperNetwork (Ha et al., 2017) to efficiently generate the LoRA weights that encode historical information. At each time step, the incoming video clip and its textual context are first compressed into a compact representation, which is then used to update a set of recurrent hidden states. To model the temporal dynamics of these hidden states, MaLoW maintains two parallel branches: a state-space model (Yang et al., 2024) for modeling local temporal dependencies, and a gated summation branch for preserving global historical semantics. The resulting states are mapped through a linear projection to the LoRA weights used by the VLM. To facilitate training, we further construct **MaLoW-Instruct**, a comprehensive streaming video dataset containing 381K question-answer (QA) pairs over 56K videos, by adapting public offline video understanding datasets to the streaming setting.

To evaluate our framework, we conduct extensive experiments across benchmarks and settings. MaLoW demonstrates state-of-the-art results on streaming video benchmarks, including OVO-Bench (Niu et al., 2025) and StreamArena (Zhang et al., 2026d). On the episodic memory (EPM) task of OVO-Bench, for example, MaLoW outperforms prior memory approaches by 3.3%–6.7% across different video backbones, while incurring modest latency overhead. Across base VLMs ranging from 0.8B to 11B parameters, MaLoW yields consistent performance gains. Beyond streaming video understanding, we further extend MaLoW to the Vision-Language-Action (VLA) setting, where incorporating historical observations and action trajectories improves the action success rate on Simpler (Li et al., 2025) WidowX by 9.3%.

Our **contributions** are summarized as follows:

- We propose MaLoW, a new *streaming memory framework that internalizes historical context* into evolving LoRA weights, introducing a distinct design in which history is represented as weights while current observations remain as input.
- We develop a recurrent HyperNetwork that incrementally updates LoRA memory from streaming video and other context. We also construct MaLoW-Instruct, a large-scale training dataset for streaming video understanding, which *we will release publicly*.
- We demonstrate that MaLoW achieves competitive results across streaming video benchmarks, attains consistent gains over diverse base VLMs, and generalizes to VLA settings.

2 Related Work

Vision-Language Models. Vision-language models (VLMs) (Liu et al., 2023; Alayrac et al., 2022; Zhang et al., 2025b; Clark et al., 2026; Qwen Team, 2026) unify diverse visual understanding and reasoning tasks through a natural-language interface. Among these tasks, long-video understanding (Zhou et al., 2025; Wang et al., 2025) is particularly challenging because models must preserve and reason over information spanning extended temporal contexts.

Prior works (Qian et al., 2024; He et al., 2024; Song et al., 2024) have explored various memory and context-management mechanisms for this offline long video understanding setting. Our work builds on pretrained VLMs and equips them with an evolving long-term memory that is needed in the streaming video setting and incrementally updated as new observations arrive.

Streaming VLMs with Memory. Recent works on streaming VLMs (Xia et al., 2026; Chen et al., 2024a; Li et al., 2026; Wang et al., 2026) explore several complementary directions, including constructing higher-quality instruction-tuning data, designing architectures tailored to streaming inputs, and incorporating stronger video encoders. More recent studies (Lu et al., 2026; Shen et al., 2026) show that strong VLMs with a small local video window yield surprisingly strong performance on real-time questions on existing streaming video benchmarks.

Another line of work focuses on the memory mechanisms. Some methods compress historical visual observations into textual summaries or memories (Liu et al., 2026; Guan et al., 2026; Yao et al., 2026), while others reduce the accumulated visual tokens through clustering or token merging (Zhang et al., 2024). Additional approaches organize historical information into hierarchical memory structures (Zeng et al., 2026; Zhang et al., 2026a), or maintain a fixed-size memory buffer that selectively preserves useful historical observations (Qian et al., 2025; Yao et al., 2025). Despite their differences, these methods generally address the growing streaming context by compressing historical information into the token budget in working memory.

In contrast, our work handles the history in streaming video by internalizing historical information directly into low-rank representations of the VLM parameters, rather than maintaining a separate memory pathway or keeping history in the working memory.

Low-Rank Adaptation and Context Internalization. Low-rank adaptation (LoRA) (Hu et al., 2022) has been widely adopted as a parameter-efficient approach for adapting pretrained vision-language models (VLMs) and multi-modal large language models (VLMs) to downstream tasks or new domains (Dettmers et al., 2023; Chen et al., 2024b; Liu et al., 2025). Recently, several works have explored using LoRA as a representation for encoding data-specific patterns (Shrestha et al., 2025; Yin et al., 2026) or contextual information (Charakorn et al., 2026; 2025). These approaches demonstrate that model parameters can be a good representation for storing information beyond conventional token-based context. However, they typically require additional optimization when adapting to a new pattern, making it difficult to apply them directly to evolving inputs. δ-mem (Lei et al., 2026) explores dynamically representing textual information through LoRA updates without test-time training. Nevertheless, it still relies on access to the historical context during inference.

Different from these prior approaches, our work specifically handles the streaming setting, and introduces a context internalization mechanism that progressively incorporates incoming visual context into low-rank model representations.

3 Internalizing Long-Term History for Streaming Videos

3.1 Problem Formulation: Streaming Video and Internalizing History

A streaming VLM M_θ, parameterized by θ, must continuously perceive incoming visual content, respond to user queries or instructions I_t, and generate an appropriate response or decision R_t at time step t, using information available before t. Formally, we define this streaming setting as

$$R_t = M_\theta(I_t, C_{0:t-1}, C_t), \tag{1}$$

where $C_{0:t-1}$ denotes the historical context accumulated before time t, and C_t denotes the current observation. Each C may contain both visual and textual information. This formulation encompasses streaming video QA, VLA, and other real-time interactive settings.

Explicitly retaining C_0, however, leads to an ever-growing input context. We instead propose to *internalize* this history into lightweight, evolving model parameters using LoRA weights $\Delta\theta_t$:

$$\Delta\theta_t = \Phi(C_{0:t-1}; h_0), \qquad R_t = M_{\theta+\Delta\theta_t}(I_t, C_t), \tag{2}$$

where Φ is a recurrent HyperNetwork initialized with a fixed hidden state h_0, and it incrementally encodes historical context into LoRA updates. With the history-dependent $\Delta\theta_t$ applied to M_θ, the

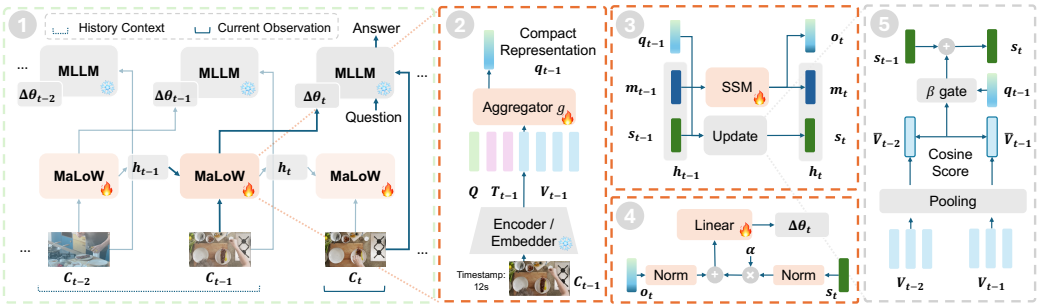


Figure 2: **Overview of MaLoW.** (1) MaLoW internalizes history into LoRA weights applied to a base VLM that processes only the current observation. (2) Context encoding compresses visual tokens V_{t-1} and timestamp tokens T_{t-1} into a compact representation q_{t-1} using a learnable query Q. (3) Hidden state updates m_t that captures local dynamics and s_t that preserves global semantics. (4) LoRA weight is decoded from the hidden states. (5) s_t is updated through gated summation.

VLM only needs to explicitly process the current observation C_t and user query I_t at inference time. Thus, *history is represented as weights, while the current observation remains as input*.

Importantly, we define Φ as the recurrent application of a single-step update f over the history context $C_{0:t-1}$, as shown in Fig. 2-1. f is itself also a HyperNetwork that updates a hidden state h_t and outputs the LoRA weights $\Delta\theta_t$, given by

$$h_t,\ \Delta\theta_t = f(h_{t-1}, C_{t-1}). \tag{3}$$

h_t offers a compact representation of $C_{0:t-1}$ and repeated application of f realizes Φ.

3.2 Our Design: MaLoW

We now describe the design of MaLoW, focusing on the single-step update f. Conceptually, f consists of three parts: (1) the encoding of context C_{t-1}; (2) the update of the hidden state h_t; and (3) the decoding of the LoRA weights $\Delta\theta_t$. This is illustrated in Fig. 2-(2-5).

Context Encoding. Videos contain substantial redundancy (Shang et al., 2025; Zhong et al., 2025). Given the observation C_{t-1} consisting of a video clip and its textual context (e.g., timestamps in text format), we first compress it into a compact representation. This is done by first encoding the video and text into their embeddings V_{t-1} (video) and T_{t-1} (text) using the vision encoder and the text tokenizer of the VLM. We then learn a set of queries Q to compress V_{t-1} and T_{t-1}

$$q_{t-1} = g([V_{t-1}; T_{t-1}; Q]) \tag{4}$$

where q_{t-1} is the resulting compact representation of C_{t-1}. g is realized using a few self-attention layers, with the concatenated $[V_{t-1}, T_{t-1}, Q]$ as the input (see Fig. 2-2).

Hidden-State Update. In interactive settings, new observations arrive continuously over time. An effective memory mechanism should therefore capture both the temporal dependencies among observations and the global semantics accumulated throughout the interaction. To this end, we decompose the hidden state h_t into two components: $h_t = [m_t; s_t]$, where m_t models local temporal dynamics in the stream, while s_t captures longer-term semantic information.

Specifically, m_t is modeled by a State-Space Model (SSM) using DeltaNet from Yang et al. (2024). Given q_{t-1} and the previous state m_{t-1}, the SSM is updated as

$$m_t, o_t = \mathrm{SSM}(q_{t-1}, m_{t-1}), \tag{5}$$

where m_t denotes the updated state and o_t the corresponding output. Through recurrent updates, m_t captures the temporal dependencies across the observed stream (see Fig. 2-3).

In parallel, s_t is simply a gated summation of q_{t-1}, acting as a skip connection to allow longer-term memory (Hochreiter & Schmidhuber, 1997). This is given by

$$s_t = s_{t-1} + \beta_{t-1} q_{t-1} \quad \text{with} \quad \beta_{t-1} = \mathbb{I}\big[\mathrm{sim}(\bar{V}_{t-1}, \bar{V}_{t-2}) \le \beta\big], \tag{6}$$

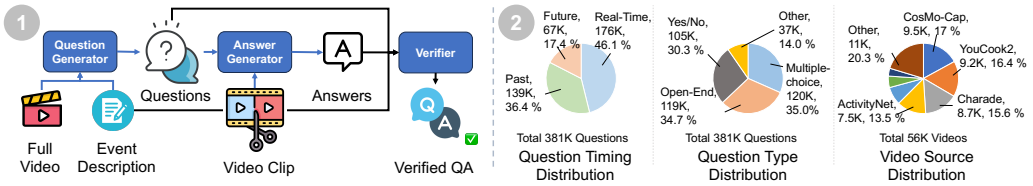


Figure 3: Overview of the MaLoW data pipeline and statistics of **MaLoW-Instruct**. **(1)**: Our data pipeline consists of a Question Generator, an Answer Generator, and a Verifier. **(2)**: The distribution of the streaming question timing, the question type, and video source in MaLoW-Instruct.

where β_t is a binary gate, $\mathbb{I}$ is the indicator function, and $\text{sim}(\bar{V}_{t-1}, \bar{V}_{t-2})$ is the cosine similarity on the averaged pooled V_{t-1}, V_{t-2}. This is shown in Fig. 2-5.

LoRA Decoding. Finally, we convert the accumulated historical information, represented by o_t and s_t, into LoRA weights $\Delta\theta_t$ using a readout function h

$$\Delta\theta_t = h(o_t, s_t) \tag{7}$$

h is realized as a linear mapping on top of o_t and s_t normalized with RMSNorm (Zhang & Sennrich, 2019), with an additional learnable gating parameter α applied to the normalized version of s_t (see Fig. 2-4). The resulting LoRA weights $\Delta\theta_t$ will be applied to the VLM at the next time step.

3.3 Model Training and Theoretic Analysis

Loss Function. We optimize MaLoW using the loss function

$$\mathcal{L} = \mathcal{L}_{\text{ntp}} + \gamma\, \mathcal{L}_{\text{LoRA}}. \tag{8}$$

$\mathcal{L}_{ntp}$ is the standard next-token prediction objective

$$\mathcal{L}_{ntp} = -\mathbb{E}_{(C_{0:t-1},\, C_t,\, I_t,\, y_t)\sim\mathcal{D}} \sum_{k=1}^{|y_t|} \log p_{\theta+\Delta\theta_t}\left(y_{t,k} \mid I_t, C_t, y_{t,<k}\right). \tag{9}$$

$\mathcal{L}_{\text{LoRA}}$ regularizes the magnitude of the generated LoRA weights following the previous work (Charakorn et al., 2025):

$$\mathcal{L}_{\text{LoRA}} = \Sigma_{s\in\mathcal{S}} \left(\text{mean}\,|\mathbf{A}_s| + \text{mean}\,|\mathbf{B}_s|\right) / |\mathcal{S}|, \tag{10}$$

where $\mathcal{S}$ indexes LoRA weights attached to different parameters of the VLM, $\mathbf{A}_s$ and $\mathbf{B}_s$ are the factorized low-rank matrices in the corresponding LoRA weights.

Training Scheme. During training, VLM parameters θ is frozen and the historical context $C_{0:t-1}$ are not directly provided to the VLM when predicting y_t. Instead, $C_{0:t-1}$ can only influence the prediction through the generated LoRA weights $\Delta\theta_t$. The prediction loss is backpropagated through $\Delta\theta_t$ to optimize the parameters in the HyperNetwork f, encouraging MaLoW to preserve historical information that is useful for future interactions while discarding redundant context.

Theoretical Analysis. We also provide theoretical analysis in Appendix A to establish the feasibility of representing history-induced feature transformations through low-rank parameter updates.

3.4 Training Data Construction: MaLoW-Instruct

To train MaLoW to enable adapted VLMs to retrospect over past observations, preserve relevant information for understanding the present, and support future responses or actions, we construct a comprehensive training dataset for streaming video understanding, named as **MaLoW-Instruct**.

Data Sources and Dataset Construction. Our data construction pipeline consists of three main components: (1) a *question generator*, (2) an *answer generator*, and (3) an *verifier* (Figure 3-1). We build on existing video temporal grounding and procedural action datasets that provide event/action timestamps and corresponding descriptions. For each annotated event, the question generator takes the video together with the event description and generates candidate questions about either the target event itself or its surrounding context, an *expected answer*, along with the question timestamps t.

We consider diverse question formats, including open-ended, multiple-choice, and yes/no questions. To prevent data leakage, we filter out the samples used in OVO-Bench (Niu et al., 2025).

Given a generated question, the answer generator then processes the video up to timestamp t and produces a *model answer*. Finally, the verifier takes the generated question, *expected answer*, *model answer*, and the video context surrounding the target event to assess both the validity of the question and the correctness of the answer. An unreasonable question or an inconsistent answer typically results in disagreement between the expected answer, generated answer, and supporting video context. We therefore retain only QA pairs that pass the verification stage to form the final training dataset. We use Qwen3.5-122B-A10B (Qwen Team, 2026) as both the question and answer generator, and Qwen3.5-397B-A17B as the verifier. Please see more details in the Appendix B.1 and B.2.

Dataset Statistics. MaLoW-Instruct contains 381K instruction samples on 56K videos. We summarize the distributions and statistics of the videos and questions in Figure 3-2. Approximately 36% of the QA pairs have query timestamps that occur after the corresponding event has ended, providing explicit supervision for learning the MaLoW module to retain and retrieve historical information. Moreover, for many real-time questions, part of the relevant event falls outside the current observation window, further encouraging the model to leverage historical context through MaLoW. The question types are approximately evenly distributed among multiple-choice, open-ended, and yes/no questions. The videos are collected from 14 datasets, among which CosMo-Cap, YouCook2, Charades-STA (Gao et al., 2017), and ActivityNet-Captions (Krishna et al., 2017) together account for 62.5% of all videos. More dataset statistics are presented in the Appendix B.3.

4 Experiments and Results

4.1 Streaming Video Understanding

Benchmarks. Following the prior works (Shen et al., 2026; Zhang et al., 2026a), We evaluate MaLoW on the *Real-Time* split of StreamingBench (Lin et al., 2026), as well as the *Real-Time* and *Backward* splits of OVO-Bench (Niu et al., 2025). The *Backward* splits of OVO-Bench comprise three tasks: Episodic Memory (EPM), Action Sequence Identification (ASI), and Hallucination Detection (HLD). Among them, EPM is particularly well suited for assessing the effectiveness of the memory mechanism introduced by MaLoW.

Implementation Details. We build MaLoW on top of VideoChat3-4B (Li et al., 2026), Qwen3.5-9B (Qwen Team, 2026) and MOSS-VL-Realtime-11B (Wang et al., 2026). We apply rank-8 LoRA to the MLP in each attention layer and *v_proj* layers in each self-attention layer. For Qwen3.5-9B, we additionally apply LoRA to the *in_proj_qkv_proj* layers in its linear-attention blocks. For MOSS-VL, we additional apply *v_proj* layers in each cross-attention layer. We use a four-layer self-attention network as the aggregator, with a hidden dimension of 512 and eight attention heads. For each layer of the base model, we learn an independent query and maintain a corresponding hidden state. We instantiate the SSM using a four-layer DeltaNet (Yang et al., 2024) with a hidden dimension of 512. To generate the LoRA parameters, we learn separate linear projection layers for each target module and LoRA rank component. We train MaLoW on the training data introduced in Sec. 3.4 for 2,800 steps with a total batch size of 64. We use the Adam optimizer with a peak learning rate of 8×10^{-5} and a cosine learning-rate schedule. The videos are sampled at 1 fps during training and inference of MaLoW. Each C_t is a 2-second video clip, and the most recent C_t is used as the current observation. Additional implementation details are provided in the Appendix C.1.

Baselines. We categorize the baselines into three groups according to how they manage historical visual context during streaming inference. **1. Streaming Video LLMs with Full Context.** This category includes Streamo (Xia et al., 2026), VideoChat3 (Li et al., 2026), and MOSS-VL-Realtime (Wang et al., 2026). These methods retain the historical context during inference, resulting in an ever-growing context as the video stream progresses and, consequently, substantial memory and computational overhead for long-duration streams. In particular, VideoChat3 uses a local context window for OVO-Bench and the full history for StreamingBench. **2. Streaming Video LLMs with a Local Context Window.** This category includes Mage-VL (Yang et al., 2026), SimpleStream (Shen et al., 2026), and AURA (Lu et al., 2026). These methods retain only the most

Table 1: **Results in streaming video understanding**. *VideoChat3 uses 2 fps on OVO-Bench and 4 fps on StreamingBench. Gray numbers denote results reported in the original paper that differ from our re-evaluation using the official implementation and reported evaluation hyperparameters. Please refer to *Baselines Re-evaluation* in Appendix Section C.1 for more information. MaLoW (MOSS-VL-Real-11B) achieves state-of-the-art performance on OVO-Bench, while MaLoW (Qwen3.5-9B) performs best on the episodic memory (EPM) task.

Model	#Frames	Streaming Bench (R)	*OVO-Bench*											Avg.
			Real-Time Visual Perception							*Backward Tracing*				
			OCR	ACR	ATR	STU	FPD	OJR	Avg.	**EPM**	ASI	HLD	Avg.	
Offline Video LLMs														
LLaVA-Video-7B (Zhang et al., 2025b)	64	–	69.1	58.7	68.8	49.4	74.3	59.8	63.5	56.2	57.4	7.5	40.4	52.0
LongVU-7B (Shen et al., 2025)	1 fps	–	55.7	49.5	59.5	48.3	68.3	63.0	57.4	43.1	66.2	9.1	39.5	48.5
Qwen3.5-9B (Qwen Team, 2026)	1 fps	81.9	83.9	62.4	70.7	46.1	76.2	70.6	68.3	61.6	62.8	59.1	61.2	64.8
Streaming Video LLMs with full context														
Streamo-7B (Xia et al., 2026)	1 fps	–	79.2	57.8	75.0	49.4	64.4	70.1	66.0	54.6	52.0	31.7	46.1	56.1
VideoChat3-4B (Li et al., 2026)	2-4 fps*	79.0 (83.0)	92.6	78.0	77.6	59.0	74.3	71.7	75.5	58.8	58.6	46.8	54.7	65.1
MOSS-VL-Real-11B (Wang et al., 2026)	1 fps	80.6 (82.9)	86.6	70.6	84.8	64.6	71.2	73.9	75.2	60.6	64.5	92.4	**72.5**	73.8
Streaming Video LLMs with Local Context Window														
Mage-VL-4B (Yang et al., 2026)	4 (1 fps)	–	94.6	83.5	79.3	74.2	70.3	77.2	79.8	50.2	61.5	32.8	48.2	64.0
Simplestream-8B (Shen et al., 2026)	4 (1 fps)	78.3	89.3	77.1	83.6	68.5	76.2	81.0	**79.3**	49.5	56.1	54.8	53.5	66.4
AURA-8B (Lu et al., 2026)	30 (1 fps)	**83.2**	89.9	79.8	80.2	70.2	77.2	81.5	79.8	54.9	67.6	58.6	60.4	70.1
Streaming Video LLMs with Local Context Window and Memory Mechanism														
HERMES-7B (Zhang et al., 2026a)	1 fps	79.44	85.2	64.2	71.6	53.4	74.3	65.2	69.0	48.5	62.2	37.6	49.4	59.2
ThinkStream-3B (Liu et al., 2026)	2 fps	75.0	85.2	64.2	69.8	49.4	69.3	64.1	67.0	53.9	59.5	43.6	52.3	59.7
VST-7B (Guan et al., 2026)	2 fps	79.5	80.5	55.1	72.4	55.1	76.2	64.1	67.2	56.9	64.9	48.4	56.7	62.0
Ours														
MaLoW (VideoChat3-4B) - 197.6M trainable (2.2%)	1 fps	81.2	89.9	70.6	80.2	60.7	76.2	75.3	75.5	60.2	58.1	54.8	57.7	66.6
MaLoW (Qwen3.5-9B) - 308.9M trainable (3.0%)	1 fps	82.3	87.9	66.1	81.0	60.1	84.2	77.7	76.2	**63.6**	66.2	69.4	66.4	71.3
MaLoW (MOSS-VL-Real-11B) - 321.1M trainable (2.8%)	1 fps	81.0	91.9	74.3	85.3	62.9	73.3	76.6	77.4	62.0	62.8	90.3	71.7	**74.5**

recent observations within a fixed local window, ranging from approximately 4 to 30 seconds. Historical observations that fall outside the local window are directly evicted. **3. Streaming Video LLMs with a Local Context Window and Memory Mechanism.** This category includes HERMES (Zhang et al., 2026a), ThinkStream (Liu et al., 2026), and VST (Guan et al., 2026). These methods retain the most recent observations in a local context window while compressing historical information outside the window into a more compact memory representation, such as textual summaries or condensed visual features. Due to space limitations, we report the top three methods in each category here and provide the full comparison in Appendix Section D.3.

Results and Discussion. Our results are presented in Table 1. MaLoW (MOSS-VL-Real-11B) achieves a new state-of-the-art score of 74.5 on OVO-Bench, while MaLoW (Qwen3.5-9B) attains the highest EPM score of 63.6 among all compared methods. All three MaLoW variants consistently improve upon their respective base models across the two benchmarks. In particular, MaLoW (Qwen3.5-9B) improves the base model by 7.9, 5.2, and 2.0 points on the Real-Time split, Backward split, and EPM task, respectively. MaLoW (MOSS-VL-Real-11B) improves the Real-Time split and EPM by 2.2 and 1.4 points, while MaLoW (VideoChat3-4B) improves the Backward split and EPM by 3.0 and 1.4 points. Moreover, the three variants outperform the strongest prior streaming Video LLMs with explicit memory mechanisms by 4.6, 9.3, and 12.5 points, respectively. These results demonstrate the effectiveness of MaLoW's memory internalization mechanism. Notably, these gains are achieved with less than 3% additional parameters, highlighting the efficiency of MaLoW.

Inference Overhead. We profile the streaming inference latency of MaLoW and four baselines using 100 deterministically sampled OVO-Bench questions. At each query timestamp t, we measure the time required to process the current observation, prefill the KV-cache, and decode the first tokens. In particular, for MaLoW, we also include the time to generate the LoRA and apply the LoRA. The figure is presented in Fig. 4. The MaLoW strikes a balance between latency and memory-related performance. Appendix C.5 provides more experimental details, and Table 15 in the Appendix provides further analysis on the latency breakdown.

Long-Horizon Memory Tasks. To further evaluate the long-term memory capability of MaLoW, we conduct a memory stress test on the Historical Retrospection (HR) split of StreamArena (Zhang et al., 2026d). This split specifically contains questions for which the query occurs substantially later than the time at which the relevant evidence appears, with a median query–evidence temporal gap of 12.1 minutes. We compare MaLoW (Qwen3.5-9B) against strong streaming video understanding models: MOSS-VL-Realtime-11B (Wang et al., 2026), VideoChat3-4B (Li et al., 2026), AURA-

Table 2: **VLA results** on SimplerEnv-Bridge (Li et al., 2025) with the WidowX robot. SGB: Stack Green Block, PC: Put Carrot, PS: Put Spoon, PE: Put Eggplant.

Method	SGB	PC	PS	PE	AVG
Q3.5-2B-MLP	20.8	75.0	70.8	70.8	59.4
MaLoW (Q3.5-2B-MLP)	**29.2**	**75.0**	**87.5**	**83.3**	**68.8**

Table 3: **Results in long-horizon memory tasks** on the HR split of StreamArena.

Method	Avg.	L1	L2	L3	L4
VideoChat3-4B	18.0	16.6	21.8	17.9	12.6
MOSS-VL-Realtime-11B	18.3	17.1	22.7	16.3	13.7
VST-7B	20.6	**24.3**	21.8	**19.6**	13.2
AURA-8B	21.4	21.8	26.4	19.3	14.0
MaLoW (Qwen3.5-9B)	**22.8**	24.0	**28.1**	19.2	**15.7**

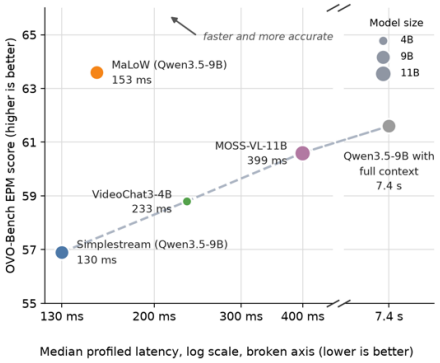


Figure 4: **Latency–accuracy tradeoffs** of MaLoW and baselines on the Episodic Memory (EPM) task of OVO-Bench.

8B (Lu et al., 2026), and VST-7B (Guan et al., 2026), which compresses the historical observation as text-based memory. The results are reported in Table 3. MaLoW achieves an average score of 22.8, outperforming VST and MOSS by 2.2% and 4.5%, respectively, demonstrating the effectiveness of its memory mechanism over long temporal horizons. See implementation details in Appendix C.4.

4.2 Additional Demonstration and Analysis

Generalization Across Different Model Scales and Families. We build MaLoW on the base models with different scales, ranging from 0.8B to 4B parameters, as well as different model families, including Qwen3.5 and Gemma4 (Gemma Team, 2026), demonstrating its generalizability. Please refer to Appendix C.3 for more implementation details. We evaluate all three splits of OVO-Bench (Backward, Real-Time, and Forward) and all four splits of StreamingBench (Real-Time, Omni, Proactive, and SQA), following the official benchmark organization. We compare MaLoW against: 1. the corresponding base model with full context and 2. the base model with a local context window. Table 4 shows the results. Across different model scales and families, MaLoW consistently improves the overall performance. In particular, MaLoW generally outperforms the corresponding full-context baseline on the real-time perception splits (cells in green), while outperforming the local-window baseline on memory-intensive splits (cells in blue). These results demonstrate that MaLoW can effectively preserve historical information while maintaining strong perception of the current observation, leading to improved overall performance over the baselines.

Extension to Vision-Language-Action (VLA) Models. We further evaluate MaLoW in the VLA setting, where it is used to internalize historical visual observations and the action trajectory for action prediction. Following VLM4VLA (Zhang et al., 2026b), we adapt Qwen3.5-2B (Qwen Team, 2026) to the VLA setting by attaching a lightweight MLP action decoder and fine-tuning the VLM backbone, resulting in our baseline model, **Q3.5-2B-MLP**. The model is trained on the Bridge and RT-1 subsets of the OXE dataset (O'Neill et al., 2024) to predict future actions based solely on the current visual observation. We then train MaLoW on top of Q3.5-2B-MLP using the Bridge subset of OXE. During this stage, the VLM backbone is frozen and receives only the current observation, while MaLoW and the MLP action decoder are optimized to internalize historical observations and action trajectories and produce more accurate action trajectories. We refer to the resulting model as **MaLoW (Q3.5-2B-MLP)**. We evaluate both models on the Simpler-Bridge with WidowX robot, as shown in Table 2. In particular, MaLoW improves the average success rate from 59.4% to 68.7%, corresponding to a gain of **9.4%**. These results demonstrate that MaLoW can effectively internalize historical information and leverage it to refine action trajectories without explicitly providing them to the base VLM. Appendix C.2 provides more implementation details.

Qualitative Results. We present qualitative results in Fig. 5, with examples from StreamArena (HR) and OVO-Bench (EPM) shown in the top and bottom rows, respectively. We compare the results from MaLoW (Qwen3.5-9B) with VST-7B, VideoChat3-4B, and MOSS-VL-realtime-11B. Overall, MaLoW produces more accurate answers than the baselines on memory-intensive tasks, demonstrating the effectiveness of its history internalization mechanism. Notably, Sample 5 shows that MaLoW can continuously update its memory and avoid being misled by obsolete cues. Sample

Table 4: **Performance across base-model scales and families**. Green and blue cells indicate improvements over full-context baselines on real-time perception and local-window baselines on memory tasks, respectively. MaLoW shows consistent performance gains.

Method	OVO-Bench (Niu et al., 2025)				StreamingBench (Lin et al., 2026)				
	Back	Real	Forw	Avg.	Real	Omni	Pro	SQA	Avg.
Base model with 1 fps, full-context baseline									
Gemma4-E4B (Gemma Team, 2026)	56.8	63.1	37.9	52.6	68.6	37.0	24.4	50.0	54.6
Qwen3.5-0.8B	34.2	60.8	48.0	47.7	69.1	32.1	22.4	39.6	52.5
Qwen3.5-2B	44.6	62.5	46.3	51.2	75.1	38.1	6.4	42.8	57.2
Qwen3.5-4B	60.1	66.9	51.2	59.4	80.4	40.7	54.8	46.0	63.8
Base model with the last 2 frames from the last 2 seconds, local-window baseline									
Gemma4-E4B	54.9	65.7	39.6	53.4	67.1	51.1	20.4	46.8	58.1
Qwen3.5-0.8B	32.3	69.9	45.1	49.1	70.1	53.5	18.8	48.0	60.5
Qwen3.5-2B	39.2	72.8	45.1	52.4	76.7	54.2	5.2	56.8	64.1
Qwen3.5-4B	50.6	75.1	39.3	55.1	79.4	60.9	45.6	52.8	69.8
MaLoW models with 1 fps, maximum 768 frames									
MaLoW (Gemma4-E4B) – 190.4M trainable (1.2%)	51.1	65.4	50.2	**55.6**	73.9	41.8	35.6	47.6	**59.6**
MaLoW (Qwen3.5-0.8B) – 67.5M trainable (6.6%)	55.4	65.4	49.7	**56.9**	76.9	46.5	29.6	46.8	**62.5**
MaLoW (Qwen3.5-2B) – 95.3M trainable (3.6%)	59.5	65.3	50.2	**58.3**	80.1	49.3	56.0	53.6	**67.0**
MaLoW (Qwen3.5-4B) – 127.6M trainable (2.5%)	61.2	70.6	50.8	**60.8**	80.8	60.2	56.8	55.2	**71.8**

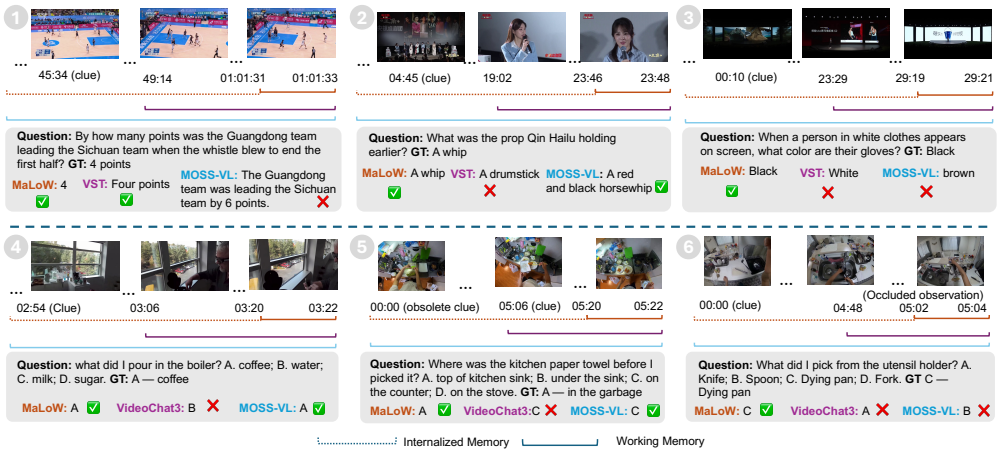


Figure 5: **Visualization of MaLoW's results** (using Qwen3.5-9B). Top: result from StreamArena (HR) split. Bottom: result from OVO-Bench (EPM) task.

6 further demonstrates that the internalized memory complements the current observation, enabling the model to recover relevant information even when the current visual evidence is occluded.

Additional Ablation Studies and Analysis. Due to space limitations, we provide ablation studies in Section D.1 of the Appendix. Similarly, additional analyses of latency, generated LoRA adapters, and qualitative failure cases can be found in Section D.2 of the Appendix.

5 Conclusion

In this work, we presented **MaLoW**, a new memory framework for streaming VLMs. Rather than repeatedly injecting compressed or retrieved history into the input context, MaLoW represents *history as weights while keeping the current observation as input*. This separation provides a natural way to carry long-term information forward without revisiting the full video stream or continually expanding the context window. To realize this design, we developed a recurrent HyperNetwork that incrementally updates LoRA memory, together with **MaLoW-Instruct**, a large-scale streaming video dataset containing 381K QA pairs over 56K videos. MaLoW achieves strong results across streaming video understanding benchmarks, with substantial gains on memory-critical tasks and only modest inference overhead. We further show that the same memory mechanism transfers to VLA models, where incorporating historical context improves action success. We hope that MaLoW offers a useful perspective on long-term memory for streaming VLMs: beyond asking what information should be stored or retrieved, we can also reconsider *where memory should reside*, in this case, directly within evolving model weights.

AI USE STATEMENT

In this work, we used generative AI tools to assist with writing refinement, literature research, experimental implementation and execution, synthetic data generation, mathematical analysis, visualization, brainstorming, and information retrieval. We did not use generative AI tools to draft sections of the paper.

Specifically, ChatGPT was used to polish the writing throughout the paper and to assist with retrieving, discovering, and summarizing related work. Codex and Claude Code were used to implement the training and evaluation code used in Section 4, execute and monitor training jobs, and assist with interpreting experimental results. Models from the Qwen3.5 series were used to generate synthetic data, as described in Section 3.4. Codex and ChatGPT were also used to assist in formulating the mathematical claims presented in Appendix A. In addition, Codex was used to assist with visualization creation, brainstorming, and information retrieval and search.

We carefully reviewed all AI-assisted outputs. For writing assistance, we reviewed all revised text and verified that retrieved references and factual information were accurate and not hallucinated. For experimental implementation, we inspected the generated code, monitored the execution of training jobs, and independently verified the interpretation of the reported results. For synthetic data generation, we manually reviewed the generated data and assessed its quality. For mathematical claims, we manually verified the correctness of the formulations and derivations. For visualizations, we confirmed that the generated figures accurately reflected the underlying data.

We take full responsibility for the final content of this work, including all text, claims, results, code, data, and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work does not involve newly recruited human subjects or new data collection from human participants. Our training data are constructed from existing video datasets as mentioned in section B, and we rely on the curation and privacy practices of the original dataset providers. As a result, our dataset and models may inherit biases, privacy concerns, or other limitations present in the underlying data and pretrained VLMs, including demographic, geographic, cultural, and content biases. We encourage users to consider these limitations when applying the resulting models in real-world settings.

Our method is intended to advance research on streaming multimodal understanding and long-term memory. At the same time, the ability to retain and utilize information from extended visual histories could potentially be applied in privacy-sensitive scenarios, such as continuous monitoring or surveillance. Such applications should be conducted with appropriate consent, privacy protections, and compliance with applicable laws and regulations. We plan to release the model and dataset artifacts under the Apache License 2.0 where permitted, while respecting the licenses and usage restrictions of the underlying datasets and source videos. We encourage responsible use of the released resources and discourage applications that violate individual privacy or enable harmful surveillance.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the model architecture, training procedure, implementation settings, and evaluation protocols in Section C. The construction process of MaLoW-Instruct, including the source datasets, data processing pipeline, and question generation procedure, is documented in Section B. Additional mathematical interpretations and supporting derivations are provided in Section A. To further support reproducibility, we plan to publicly release the training and evaluation code, trained model checkpoints, and processed MaLoW-Instruct data after publication.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. Localizing moments in video with natural language. In *Proceedings of the IEEE international conference on computer vision*, pp. 5803–5812, 2017.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert T. Lange. Text-to-lora: instant transformer adaption. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025.

Rujikorn Charakorn, Edoardo Cetin, Shinnosuke Uesaka, and Robert Tjarko Lange. Doc-to-loRA: Learning to instantly internalize contexts. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=iW1oBBO72S.

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 *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18407–18418. IEEE, 2024a.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. Longlora: Efficient fine-tuning of long-context large language models. In *International Conference on Learning Representations*, volume 2024, pp. 8220–8238, 2024b.

Christopher Clark, Jieyu Zhang, Zixian Ma, Jae Sung Park, Rohun Tripathi, Sangho Lee, Mohammadreza Salehi, Jason Ren, Chris Dongjoo Kim, Yinuo Yang, et al. Molmo2: Open weights and data for vision-language models with video understanding and grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 28652–28668, 2026.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. Tall: Temporal activity localization via language query. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5277–5285. IEEE, 2017.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 18995–19012, 2022.

Yiran Guan, Liang Yin, Dingkang Liang, Jianzhong Ju, Zhenbo Luo, Jian Luan, Yuliang Liu, and Xiang Bai. Video streaming thinking: Videollms can watch and think simultaneously. In *European Conference on Computer Vision*, pp. 80–99. Springer, 2026.

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations*, 2017.

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 *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13504–13514. IEEE, 2024.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Haroon Idrees, Amir R Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The thumos challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding*, 155:1–23, 2017.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 5156–5165, 2020.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *Proceedings of the IEEE international conference on computer vision*, pp. 706–715, 2017.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Thinking Machines Lab. Interaction models: A scalable approach to human-ai collaboration. *Thinking Machines Lab: Connectionism*, May 2026. doi: 10.64434/tml.20260511. https://thinkingmachines.ai/blog/interaction-models/.

Jingdi Lei, Di Zhang, Junxian Li, Weida Wang, Kaixuan Fan, Xiang Liu, Qihan Liu, Xiaoteng Ma, Baian Chen, and Soujanya Poria. δ-mem: Efficient online memory for large language models. *arXiv preprint arXiv:2605.12357*, 2026.

Xinhao Li, Yuhan Zhu, Xiangyu Zeng, Yuhao Dong, Haoning Wu, Zhiqiu Zhang, Yuandong Yang, Changlian Ma, Qingyu Zhang, Yansong Shi, et al. Videochat3: Fully open video mllm for efficient and generalist video understanding. *arXiv preprint arXiv:2607.14935*, 2026.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Oier Mees, Karl Pertsch, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3705–3728. PMLR, 06–09 Nov 2025. URL `https://proceedings.mlr.press/v270/li25c.html`.

Junming Lin, Zheng Fang, Chi Chen, Haoxuan Cheng, Zihao Wan, Fuwen Luo, Ziyue Wang, Peng Li, Yang Liu, and Maosong Sun. Streamingbench: Assessing the gap for mllms to achieve streaming video understanding. In *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 12147–12151. IEEE, 2026.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

Ye Liu, Zongyang Ma, Zhongang Qi, Yang Wu, Ying Shan, and Chang W Chen. Et bench: Towards open-ended event-level video-language understanding. *Advances in Neural Information Processing Systems*, 37:32076–32110, 2024.

Zhuoming Liu, Yiquan Li, Khoi Duc Nguyen, Yiwu Zhong, and Yin Li. Pave: Patching and adapting video large language models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3306–3317. IEEE, 2025.

Zikang Liu, Longteng Guo, Handong Li, Ru Zhen, Xingjian He, Ruyi Ji, Xiaoming Ren, Yanhao Zhang, Haonan Lu, and Jing Liu. Thinking in streaming video. In *European Conference on Computer Vision*, pp. 21–38. Springer, 2026.

Xudong Lu, Yang Bo, Jinpeng Chen, Shuhan Li, Xintong Guo, Huankang Guan, Fang Liu, Dunyuan Xu, Peiwen Sun, Heyang Sun, et al. Aura: Always-on understanding and real-time assistance via video streams. *arXiv preprint arXiv:2604.04184*, 2026.

Junbo Niu, Yifei Li, Ziyang Miao, Chunjiang Ge, Yuanhang Zhou, Qihao He, Xiaoyi Dong, Haodong Duan, Shuangrui Ding, Rui Qian, et al. Ovo-bench: How far is your video-llms from real-world online video understanding? In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18902–18913. IEEE, 2025.

Andreea-Maria Oncescu, Joao F Henriques, Yang Liu, Andrew Zisserman, and Samuel Albanie. Queryd: A video dataset with high-quality text and audio narrations. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2265–2269. IEEE, 2021.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Rui Qian, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Shuangrui Ding, Dahua Lin, and Jiaqi Wang. Streaming long video understanding with large language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 119336–119360. Curran Associates, Inc., 2024. doi: 10.52202/079017-3792. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/d7ce06e9293c3d8e6cb3f80b4157f875-Paper-Conference.pdf.

Rui Qian, Shuangrui Ding, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Dahua Lin, and Jiaqi Wang. Dispider: Enabling video llms with active real-time interaction via disentangled perception, decision, and reaction. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24045–24055. IEEE, 2025.

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

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 9355–9366, 2021.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22857–22867. IEEE, 2025.

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 Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 54582–54599. PMLR, 13–19 Jul 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.

Sagar Shrestha, Gopal Sharma, Luowei Zhou, and Suren Kumar. Finetuning-free personalization of text to image generation via hypernetworks. *arXiv preprint arXiv:2511.03156*, 2025.

Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Haozhe Chi, Xun Guo, Tian Ye, Yanting Zhang, et al. Moviechat: From dense token to sparse memory for long video understanding. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18221–18232. IEEE, 2024.

Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. Coin: A large-scale dataset for comprehensive instructional video analysis. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1207–1216. IEEE, 2019.

Alex Jinpeng Wang, Linjie Li, Kevin Qinghong Lin, Jianfeng Wang, Kevin Lin, Zhengyuan Yang, Lijuan Wang, and Mike Zheng Shou. Cosmo: Contrastive streamlined multimodal model with interleaved pre-training. *arXiv preprint arXiv:2401.00849*, 2024a.

Pengyu Wang, Chenkun Tan, Shaojun Zhou, Qirui Zhou, Yanxin Chen, Xingyang He, Huazheng Zeng, Jijun Cheng, Chenghao Wang, Xiaomeng Qian, et al. Moss-vl technical report. *arXiv preprint arXiv:2608.15045*, 2026.

Weihan Wang, Zehai He, Wenyi Hong, Yean Cheng, Xiaohan Zhang, Ji Qi, Ming Ding, Xiaotao Gu, Shiyu Huang, Bin Xu, et al. Lvbench: An extreme long video understanding benchmark. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22958–22967. IEEE, 2025.

Yi Wang, Yinan He, Yizhuo Li, Kunchang Li, Jiashuo Yu, Xin Ma, Xinhao Li, Guo Chen, Xinyuan Chen, Yaohui Wang, et al. Internvid: A large-scale video-text dataset for multimodal understanding and generation. In *International Conference on Learning Representations*, volume 2024, pp. 42055–42079, 2024b.

Bo Wu, Shoubin Yu, Zhenfang Chen, Joshua B Tenenbaum, and Chuang Gan. Star: A benchmark for situated reasoning in real-world videos. *arXiv preprint arXiv:2405.09711*, 2024.

Jiaer Xia, Peixian Chen, Mengdan Zhang, Xing Sun, and Kaiyang Zhou. Streaming video instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 31219–31229, 2026.

Junbin Xiao, Xindi Shang, Angela Yao, and Tat-Seng Chua. Next-qa: Next phase of question-answering to explaining temporal actions. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9772–9781. IEEE, 2021.

Senqiao Yang, Kaichen Zhang, Zhaoyang Jia, Jinghao Guo, Yifei Shen, Xinjie Zhang, Xiaoyi Zhang, Haoqing Wang, Xiao Li, Peng Zhang, et al. Mage-vl: An efficient codec-native streaming multimodal foundation model. *arXiv preprint arXiv:2607.24904*, 2026.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems*, 37:115491–115522, 2024.

Dingyu Yao, Junhao Zhou, Chenxu Yang, Chuanyu Qin, Haowen Hou, Zheming Liang, Congcong Wang, Yuhang Cao, Shenglong Ye, Shuai Xie, et al. Joyai-vl-interaction: Real-time vision-language interaction intelligence. *arXiv preprint arXiv:2606.14777*, 2026.

Linli Yao, Yicheng Li, Yuancheng Wei, Lei Li, Shuhuai Ren, Yuanxin Liu, Kun Ouyang, Lean Wang, Shicheng Li, Sida Li, et al. Timechat-online: 80% visual tokens are naturally redundant in streaming videos. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 10807–10816, 2025.

Shengming Yin, Zekai Zhang, Zecheng Tang, Kaiyuan Gao, Xiao Xu, Kun Yan, Jiahao Li, Yilei Chen, Yuxiang Chen, Heung-Yeung Shum, et al. Qwen-image-layered: Towards inherent editability via layer decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16196–16205, 2026.

Abhay Zala, Jaemin Cho, Satwik Kottur, Xilun Chen, Barlas Oguz, Yashar Mehdad, and Mohit Bansal. Hierarchical video-moment retrieval and step-captioning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23056–23065. IEEE, 2023.

Xiangyu Zeng, Kefan Qiu, Qingyu Zhang, Xinhao Li, Jing Wang, Jiaxin Li, Ziang Yan, Kun Tian, Meng Tian, Xinhai Zhao, et al. Streamforest: Efficient online video understanding with persistent event memory. *Advances in Neural Information Processing Systems*, 38:75804–75835, 2026.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

Haoji Zhang, Yiqin Wang, Yansong Tang, Yong Liu, Jiashi Feng, Jifeng Dai, and Xiaojie Jin. Flash-vstream: Memory-based real-time understanding for long video streams. *arXiv preprint arXiv:2406.08085*, 2024.

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)*, pp. 8411–8430, 2026a.

Jianke Zhang, Xiaoyu Chen, Yanjiang Guo, Yucheng Hu, and Jianyu Chen. VLM4VLA: Revisiting vision-language-models in vision-language-action models. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=tc2UsBeODW.

Jun Zhang, Teng Wang, Yuying Ge, Yixiao Ge, Xinhao Li, and Limin Wang. Timelens: Rethinking video temporal grounding with multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10419–10429, 2026c.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, et al. Lmms-eval: Reality check on the evaluation of large multimodal models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 881–916, 2025a.

Xichen Zhang, Guankai Li, Yinghao Zhu, Shijian Wang, Sitong Wu, Shaozuo Yu, Meng Chu, Yuan Lu, and Jiaya Jia. Streamarena: Toward continuous, interactive, and long-horizon agentic streaming video understanding. *arXiv preprint arXiv:2608.05703*, 2026d.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun MA, Ziwei Liu, and Chunyuan Li. LLaVA-video: Video instruction tuning with synthetic data. *Transactions on Machine Learning Research*, 2025b. ISSN 2835-8856. URL https://openreview.net/forum?id=EElFGvt39K.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand, 2024. Association for Computational Linguistics. URL http://arxiv.org/abs/2403.13372.

Yiwu Zhong, Zhuoming Liu, Yin Li, and Liwei Wang. Aim: Adaptive inference of multi-modal llms via token merging and pruning. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20180–20192. IEEE, 2025.

Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Zhengyang Liang, Shitao Xiao, Minghao Qin, Xi Yang, Yongping Xiong, Bo Zhang, et al. Mlvu: Benchmarking multi-task long video understanding. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13691–13701. IEEE, 2025.

Luowei Zhou, Chenliang Xu, and Jason J Corso. Towards automatic learning of procedures from web instructional videos. In *AAAI Conference on Artificial Intelligence*, pp. 7590–7598, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17344.

Dimitri Zhukov, Jean-Baptiste Alayrac, Ramazan Gokberk Cinbis, David Fouhey, Ivan Laptev, and Josef Sivic. Cross-task weakly supervised learning from instructional videos. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3532–3540. IEEE, 2019.

A MATHEMATICAL INTERPRETATION

MaLoW carries information from earlier video clips through history-dependent parameter updates. This appendix provides a mathematical interpretation of the feasibility of encoding the effect of historical context into low-rank updates of the MLP projections within a Transformer block. Specifically, we trace how a query reads historical information in section A.1, how this read induces a transformation of the features entering the MLP in section A.2, and how such a transformation can be realized or approximated by LoRA-style weight updates in section A.3. The analysis does not claim that MaLoW explicitly reconstructs the analytical history matrix M_H or is equivalent to full-context attention. Rather, it establishes that historical information can, under the assumptions described below, be represented in parameter space through low-rank updates. MaLoW can be viewed as one of the learned instantiations of this principle, with our empirical results demonstrating that its recurrent state and generated adapters effectively preserve history-dependent information in practice. The exact identities below first assume fixed historical keys and values and an unnormalized dot-product read, while Section A.4 and Section A.5 extend the interpretation to the online setting, normalized attention, and the normalization operations in the backbone.

A.1 HISTORICAL CONTEXT AS A READ OPERATOR

A streaming video arrives as a sequence of clips. At step t, let $H = (C_1, \ldots, C_{t-1})$ denote the clips already observed. Suppose their visual tokens have produced n_H key–value pairs $k_i \in \mathbb{R}^{d_k}$ and $v_i \in \mathbb{R}^{d_v}$. A current query $q \in \mathbb{R}^{d_k}$ reads this history through

$$O_H(q) = \sum_{i=1}^{n_H} v_i (k_i^\top q). \tag{A.1}$$

All vectors are columns. Keys and values are fixed while the query varies.

Each term first measures a match, $k_i^\top q$, and then weights the corresponding value v_i. Collecting the terms gives

$$O_H(q) = M_H q, \qquad M_H = \sum_{i=1}^{n_H} v_i k_i^\top \in \mathbb{R}^{d_v \times d_k}. \tag{A.2}$$

Thus the same read can be computed either from historical tokens or from one matrix. The matrix stores how the history responds to a query: M_H is shared by all queries for that fixed history. Its dimensions depend on the feature widths, while the number of stored tokens grows with the video. This is the associative-memory interpretation of linear attention (Katharopoulos et al., 2020; Schlag et al., 2021).

A.2 FROM HISTORICAL READS TO FEATURE TRANSFORMATIONS

To connect the read to an MLP, consider a block with current feature $x \in \mathbb{R}^d$, query projection $W_Q \in \mathbb{R}^{d_k \times d}$, and read-output projection $W_O \in \mathbb{R}^{d \times d_v}$. The same feature supplies the query and the residual stream:

$$q = W_Q x, \qquad x_H = x + W_O O_H(q), \qquad y_H = F(x_H), \tag{A.3}$$

where F is the MLP. The feature entering the MLP changes from x to x_H when the historical read is added.

Substituting Eq. (A.2) into this block gives

$$x_H = (I + P_H)x, \qquad \boxed{P_H = W_O M_H W_Q} \in \mathbb{R}^{d \times d}. \tag{A.4}$$

M_H maps a query to a historical read. P_H expresses that same read as a change to the MLP input. The projections W_Q and W_O determine their relation. Consequently, the output to reproduce is

$$y_H = F\big((I + P_H)x\big). \tag{A.5}$$

This identifies a concrete role for an MLP update: reproduce how the MLP would process the feature after history has changed it.

A.3 Low-Rank Realizations in MLP Projections

Let a rank-r factorization approximate the history matrix:

$$\widehat{M}_H = B_H A_H, \qquad B_H \in \mathbb{R}^{d_v \times r}, \quad A_H \in \mathbb{R}^{r \times d_k}. \tag{A.6}$$

The corresponding feature transformation is

$$\widehat{P}_H = W_O B_H A_H W_Q, \qquad \operatorname{rank}(\widehat{P}_H) \le r. \tag{A.7}$$

These factors are shared across queries for the same history. The required rank depends on how accurately they reproduce the reads used by the model.

Gate and up Projections. Consider the gated MLP

$$F(x) = W_d\big[\sigma(W_g x) \odot (W_u x)\big], \tag{A.8}$$

where $W_g, W_u \in \mathbb{R}^{m \times d}$, $W_d \in \mathbb{R}^{d_o \times m}$, σ acts elementwise, and $\odot$ is the elementwise product. Biases are omitted; fixed biases can be retained in the same identities.

Proposition A.1 (Realizing the History-Induced Input Change). Keep W_d fixed and set

$$\Delta W_g = W_g \widehat{P}_H, \qquad \Delta W_u = W_u \widehat{P}_H. \tag{A.9}$$

Then both updates have rank at most r, and the updated MLP satisfies

$$F_\Delta(x) = F\big((I + \widehat{P}_H)x\big) \quad \text{for every } x. \tag{A.10}$$

Proof. The updated gate preactivation is $(W_g + \Delta W_g)x = W_g(I + \widehat{P}_H)x$; the same identity holds for the up projection. Substituting both expressions into Eq. (A.8) proves Eq. (A.10). The explicit factorizations are

$$\begin{aligned} \Delta W_g &= (W_g W_O B_H)(A_H W_Q), \\ \Delta W_u &= (W_u W_O B_H)(A_H W_Q). \end{aligned} \tag{A.11}$$

Each product has an inner dimension of r, so each update has rank at most r. □

Equation (A.11) has the form used by LoRA (Hu et al., 2022); any fixed nonzero adapter scale can be absorbed into a factor. The MLP nonlinearity is preserved exactly because the update changes its inputs before the activation and multiplication. In this construction, gate and up updates work together to apply the same history-induced input change.

How Read Error Reaches the MLP Output. Write $E_H = M_H - \widehat{M}_H$. The feature error is

$$x_H - (I + \widehat{P}_H)x = W_O E_H W_Q x. \tag{A.12}$$

If F is L_F-Lipschitz on the segment between these two inputs, then

$$\|y_H - F_\Delta(x)\|_2 \le L_F \, \|W_O\|_2 \, \|E_H W_Q x\|_2 . \tag{A.13}$$

Here matrix norms are spectral norms. For a truncated rank-r singular value decomposition, $\|E_H\|_2 = \sigma_{r+1}(M_H)$, and therefore

$$\|y_H - F_\Delta(x)\|_2 \le L_F \, \|W_O\|_2 \, \sigma_{r+1}(M_H) \, \|W_Q x\|_2 . \tag{A.14}$$

When $r \ge \operatorname{rank}(M_H)$, the factorization is exact and the error is zero. For smaller ranks, Eq. (A.13) ties the approximation to the queries actually encountered, which can be more informative than a worst-case bound.

The Down Projection. The same target can also be expressed at the MLP's intermediate features. Define

$$z(x) = \sigma(W_g x) \odot (W_u x), \qquad z_H(x) = z\big((I + P_H)x\big). \tag{A.15}$$

For a fixed history, suppose one shared matrix $S_H \in \mathbb{R}^{m \times m}$, with rank at most s, approximates the change across current inputs:

$$z_H(x) = (I + S_H)z(x) + e_H(x). \tag{A.16}$$

Keeping gate and up fixed and setting $\Delta W_d = W_d S_H$ gives

$$\begin{aligned} F_{\Delta d}(x) &= (W_d + \Delta W_d) z(x), \\ \|y_H - F_{\Delta d}(x)\|_2 &\le \|W_d\|_2 \, \|e_H(x)\|_2 , \qquad \operatorname{rank}(\Delta W_d) \le s. \end{aligned} \tag{A.17}$$

For $S_H = D_H C_H$, the update is $\Delta W_d = (W_d D_H) C_H$, another LoRA factorization. A useful low-rank S_H is an additional property of the intermediate features to measure; the nonlinear map z makes it a separate condition from low rank of M_H.

The gate/up construction and the down construction describe two ways to reproduce the same target y_H. Jointly trained adapters can distribute the correction across these projections. MaLoW learns module-specific factors; the identities above exhibit computations available within that parameterization.

A.4 Updating the Historical Effect Online

When a new clip C arrives, its key–value pairs add a new contribution:

$$M_{H \cup C} = M_H + \Delta M_C, \qquad \Delta M_C = \sum_{i \in C} v_i k_i^\top . \tag{A.18}$$

Here $H \cup C$ denotes appending the clip to the history. With fixed W_Q and W_O, the feature transformation follows the same update:

$$P_{H \cup C} = P_H + W_O \Delta M_C W_Q . \tag{A.19}$$

The update therefore extends the same effect traced from the historical read to the MLP input. Previously produced keys and values are assumed to stay fixed; the matrix retains their summed outer-product contribution.

Cost and Rank. For n tokens per clip, adding the outer products costs $O(n d_v d_k)$ per clip and stores $d_v d_k$ matrix entries. Over T clips, incremental accumulation costs $O(T n d_v d_k)$. Rebuilding the matrix from all available key–value pairs at every step costs $O(T^2 n d_v d_k)$. These costs cover dense matrix accumulation given the keys and values. Video encoding, compression, adapter generation, and responses add their respective costs.

Each new clip can increase the matrix rank. Maintaining a fixed rank can be written as an update followed by compression:

$$\widehat{M}_t = \mathcal{C}_r \left(\widehat{M}_{t-1} + \Delta M_{C_t} \right), \tag{A.20}$$

where $\mathcal{C}_r$ returns a matrix of rank at most r. The compression algorithm determines both its additional cost and which historical responses the bounded-rank memory preserves.

Connection to MaLoW. MaLoW implements an online memory with a learned state update U_ϕ and adapter generator D_ϕ:

$$\Delta\theta_t = D_\phi(h_{t-1}), \qquad h_t = U_\phi\big(h_{t-1}, E(C_t)\big). \tag{A.21}$$

Here $E(C_t)$ encodes the incoming clip. The response at step t uses $\Delta\theta_t$ together with the current clip; writing that clip into h_t prepares the memory for later responses. Equations (A.18)–(A.20) describe an analytical update-and-compression process. MaLoW learns a surrogate from the task objective: its recurrent state and generated adapters aim to preserve the historical effects useful for later responses. Their behavior can be compared with the matrix read and feature transformation derived above.

A.5 Extensions to the Backbone Computation

The same reasoning can include the remaining operations by retaining their normalization terms or using a local affine approximation.

Normalized Kernel Reads. For a finite feature map $\phi : \mathbb{R}^{d_k} \to \mathbb{R}^p$ and weights $\kappa(q,k) = \phi(k)^\top \phi(q)$, define

$$N_H = \sum_{i \in H} v_i \phi(k_i)^\top, \qquad b_H = \sum_{i \in H} \phi(k_i). \tag{A.22}$$

If the denominator is positive, the normalized read is exactly

$$R_H(q) = \frac{N_H \phi(q)}{b_H^\top \phi(q)}. \tag{A.23}$$

Both N_H and b_H update additively by clip. When a read includes current tokens C, their contributions are combined before division:

$$R_{H \cup C}(q) = \frac{(N_H + N_C)\phi(q)}{(b_H + b_C)^\top \phi(q)}. \tag{A.24}$$

If N_H is approximated by $\widehat{N}_H$, b_H is kept exact, and $b_H^\top \phi(q) \geq \eta > 0$, then

$$\left\| R_H(q) - \widehat{R}_H(q) \right\|_2 \leq \frac{\left\| (N_H - \widehat{N}_H)\phi(q) \right\|_2}{\eta}. \tag{A.25}$$

The denominator state is therefore part of the historical read, alongside its numerator matrix.

Softmax Reads. For fixed historical keys and values and temperature $\tau > 0$, let

$$R_H(q) = \sum_{i \in H} p_i(q) v_i, \qquad p_i(q) = \frac{\exp(k_i^\top q/\tau)}{\sum_{j \in H} \exp(k_j^\top q/\tau)}. \tag{A.26}$$

Its local dependence on a query is described by an offset and a Jacobian. Writing $\mu_k(q) = \sum_i p_i(q) k_i$, direct differentiation gives

$$J_H(q) = \frac{1}{\tau} \sum_{i \in H} p_i(q) \big(v_i - R_H(q)\big) \big(k_i - \mu_k(q)\big)^\top. \tag{A.27}$$

Around a reference query q_0, with $\delta = q - q_0$,

$$R_H(q) = R_H(q_0) + J_H(q_0)\delta + \rho_H(q). \tag{A.28}$$

If J_H is L_J-Lipschitz along the segment from q_0 to q, the integral form of the Taylor remainder gives

$$\|\rho_H(q)\|_2 \leq \tfrac{1}{2} L_J \|\delta\|_2^2 .$$

For a rank-r matrix $\widehat{J}_H$ satisfying $\left\| J_H(q_0) - \widehat{J}_H \right\|_2 \leq \epsilon_r$, define the affine read as $\widehat{R}_H(q) = R_H(q_0) + \widehat{J}_H \delta$. It satisfies

$$\left\| R_H(q) - \widehat{R}_H(q) \right\|_2 \leq \epsilon_r \|\delta\|_2 + \tfrac{1}{2} L_J \|\delta\|_2^2 . \tag{A.29}$$

This gives a local low-rank interpretation with the reference response $R_H(q_0)$ retained explicitly.

When softmax attention reads both history and current tokens, define

$$Z_H(q) = \sum_{i \in H} \exp(k_i^\top q/\tau), \qquad Z_C(q) = \sum_{i \in C} \exp(k_i^\top q/\tau).$$

For nonempty H and C,

$$R_{H \cup C}(q) = w_H(q) R_H(q) + \big(1 - w_H(q)\big) R_C(q), \quad w_H(q) = \frac{Z_H(q)}{Z_H(q) + Z_C(q)}. \tag{A.30}$$

Thus adding history changes both the available values and their relative weights. A reconstruction of the joint read must retain this mixing weight together with the historical response.

Table 5: Distribution of training videos across underlying datasets and data sources.

Underlying source	TimeLens	LLaVA-Video	ET-Instruct	Others	Unique Wideos	Weight
CosMo-Cap (Wang et al., 2024a)	9,524	–	–	–	**9,524**	**17.05%**
YouCook2 (Zhou et al., 2018)	–	7,451	–	1,704	**9,155**	**16.39%**
Charades (Gao et al., 2017)	–	8,692	–	–	**8,692**	**15.56%**
ActivityNet (Krishna et al., 2017)	–	7,543	–	–	**7,543**	**13.50%**
InternVid-VTime (Wang et al., 2024b)	4,644	–	–	–	**4,644**	**8.31%**
NextQA (Xiao et al., 2021)	–	3,704	–	–	**3,704**	**6.63%**
DiDeMo (Anne Hendricks et al., 2017)	2,948	–	–	–	**2,948**	**5.28%**
STAR (Wu et al., 2024)	–	2,008	–	–	**2,008**	**3.59%**
COIN (Tang et al., 2019)	–	–	1,995	–	**1,995**	**3.57%**
CrossTask (Zhukov et al., 2019)	–	–	–	1,640	**1,640**	**2.94%**
QuerYD (Oncescu et al., 2021)	1,511	–	–	–	**1,511**	**2.70%**
HiREST (Zala et al., 2023)	1,204	–	–	–	**1,204**	**2.16%**
Ego4D (Grauman et al., 2022)	–	905	–	–	**905**	**1.62%**
THUMOS14 (Idrees et al., 2017)	–	–	–	394	**394**	**0.71%**
Total	**19,831**	**30,303**	**1,995**	**4,155**	**55,867**	**100.00%**

Normalization and the MLP Input. In a pre-normalized backbone, the query and MLP each receive normalized features. A controlled comparison can isolate the historical read within one block. Fix the key–value sets, and let both paths receive the same residual feature u. With attention input normalization N_a, MLP input normalization N_m, and $q(u) = W_Q N_a(u)$, the two MLP inputs are

$$\begin{aligned} T_H(u) &= N_m\left(u + W_O R_{H\cup C}(q(u))\right), \\ T_0(u) &= N_m\left(u + W_O R_C(q(u))\right). \end{aligned} \tag{A.31}$$

These expressions place both normalization operations and softmax mixing inside the feature map being compared. For $D_H(u) = T_H(u) - T_0(u)$, a local expansion at u_0 is

$$D_H(u) = D_H(u_0) + J_D(u_0)(u - u_0) + r_D(u). \tag{A.32}$$

The constant term records the historical change at the reference feature; the Jacobian records how that change varies nearby. If J_D is L_D-Lipschitz on the segment, $\|r_D(u)\|_2 \le \frac{1}{2} L_D \|u - u_0\|_2^2$.

Comparing Complete Forward Passes. Removing history can also change earlier layers' outputs and the current-token keys and values. Let x_H^{act} and x_0^{act} be the normalized MLP inputs for the same current token in retained-history and current-only runs of the frozen backbone. For a fixed history, fit one rank-r matrix P across these pairs and measure the residual:

$$x_H^{\mathrm{act}} = (I + P)x_0^{\mathrm{act}} + e_H^{\mathrm{in}}. \tag{A.33}$$

The gate/up construction with $\Delta W_g = W_g P$ and $\Delta W_u = W_u P$ gives

$$\left\| F(x_H^{\mathrm{act}}) - F_\Delta(x_0^{\mathrm{act}}) \right\|_2 \le L_F \left\| e_H^{\mathrm{in}} \right\|_2, \tag{A.34}$$

provided F is L_F-Lipschitz on the segment from x_H^{act} to $(I + P)x_0^{\mathrm{act}}$. The residual measures how well the shared P captures the full history-induced input change, including normalization and changes from earlier layers. This connects the block interpretation to a measurable approximation in the backbone.

In conclusion, this analysis establishes the theoretical feasibility of compressing the effect of historical context into low-rank parameter updates of the MLP modules within a Transformer block. In particular, it shows that, under the assumptions described above, a history-dependent feature transformation induced by historical context can be realized or approximated through LoRA-style updates to the MLP projections. Importantly, this analysis does not claim that MaLoW exactly reconstructs the analytical history matrix M_H, nor that its learned recurrent state and generated adapters are equivalent to full-context attention. Rather, the derivation provides a mathematical interpretation demonstrating that such a parameter-space representation of historical information is feasible. Our empirical results further show that MaLoW provides an effective learned instantiation of this approximation in practice.

B Additional Dataset Details

In this section, we provide more information related to the preparation of MaLoW-Instruct. We share the video source details in section B.1, the data pipeline and the annotation construction details in section B.2, and finally the dataset statistic in section B.3.

B.1 Video Sources

We begin the dataset construction by collecting videos from multiple existing sources, including TimeLens-100K (Zhang et al., 2026c), the academic subset of LLaVA-Video-178K (Zhang et al., 2025b), ET-Instruct (Liu et al., 2024), and several additional video datasets, including YouCook2 (Zhou et al., 2018), CrossTask (Zhukov et al., 2019), and THUMOS14 (Idrees et al., 2017). We primarily build upon these large-scale collections rather than gathering every underlying dataset independently, as they already consolidate videos from diverse academic sources and have undergone prior curation and preprocessing, providing a more scalable and reliable starting point for dataset construction. To prevent evaluation contamination, we further filter out videos that overlap with those used in OVO-Bench (Niu et al., 2025). In contrast, StreamingBench and StreamArena do not overlap with any of our video sources and therefore require no additional filtering. After aggregating all sources and removing duplicate and overlapping videos, the resulting dataset contains 55,867 unique videos drawn from 14 underlying video datasets. A detailed breakdown of the video distribution is provided in Table 5.

B.2 Data Pipeline and Annotation Construction

As illustrated in Figure 3-1, our data construction pipeline consists of three components: a *Question Generator*, an *Answer Generator*, and a *Verifier*. To construct the streaming video training dataset, we begin with an event description associated with a specific temporal interval. These descriptions are obtained either from the original annotations of the 14 datasets described in Section B.1, or from the question–answer pairs provided by TimeLens and ET-Instruct.

Question Generation. We first prompt the *Question Generator* to generate a new question for a target event, conditioned on the full video, the event description, and its corresponding temporal interval. By observing the full video, the *Question Generator* can leverage both the target event and its surrounding visual context to construct a question that is grounded in the video content. The prompt explicitly controls the question type, which is sampled from four categories: open-ended, multiple-choice, Yes/No, and counting questions. Along with each generated question, the *Question Generator* also produces an *expected answer*, which is subsequently used during the verification stage. To cover different temporal reasoning scenarios, we further assign each question a randomly sampled question-timestamp type from three categories: *past*, *real-time*, and *future*. Given a target event spanning the interval $[t_{\text{start}}, t_{\text{end}}]$, a *past* question is assigned a timestamp $t > t_{\text{end}}$, requiring the model to retrospectively recall an event that has already occurred. A *real-time* question is assigned a timestamp $t \in [t_{\text{start}}, t_{\text{end}}]$, such that the target event is ongoing when the question is asked. A *future* question is assigned a timestamp $t < t_{\text{start}}$, requiring the model either to anticipate a future event from the available context or to abstain from answering when the observed evidence is insufficient. The exact question timestamp is then randomly sampled from the temporal range corresponding to the selected timestamp type. We use Qwen3.5-122B-A10B (Qwen Team, 2026) as the *Question Generator*. For question generation, we launch four parallel workers, each serving one Qwen3.5-122B-A10B instance with vLLM (Kwon et al., 2023) on two H200 GPUs.

Answer Generation. After obtaining the generated questions and their corresponding timestamps t, we prompt the *Answer Generator* to produce a *model answer* using only the video content observable up to time t. This setup ensures that the generated answer respects the streaming setting and does not access visual information occurring after the question timestamp. We use Qwen3.5-122B-A10B (Qwen Team, 2026) as the *Answer Generator*. For answer generation, we launch four parallel workers, each serving one Qwen3.5-122B-A10B instance with vLLM on two H200 GPUs.

Verification. After obtaining the generated question and its corresponding *model answer*, we prompt the *Verifier* to assess the quality of the generated question–answer pair. The *Verifier* takes as input the generated question, the *expected answer*, the *model answer*, and the original event description used during question generation. It evaluates both whether the generated question is valid and well

Table 6: Distribution of questions across timestamp classes.

Timestamp class	Questions	Raw corpus share
Realtime	175,773	46.1%
Past	138,600	36.4%
Future	66,547	17.4%
Total	**380,920**	**100.00%**

Table 7: Distribution of questions across question types.

Question type	Questions	Raw corpus share
Multiple-choice	120,416	35.0%
Open-ended	119,371	34.7%
Yes/no	104,508	30.0%
Counting	36,625	14.0%
Total	**380,920**	**100.00%**

Table 8: Statistics of questions, answers, and video durations in the dataset.

	Min	Max	Mean
Question tokens	33	263	70.4
Answer tokens	1	129	3.8
Question timestamp (s)	0.07	1,015.8	57.2
Full video length (s)	5.04	1,673.4	89.5

grounded in the target event and whether the *model answer* is consistent with the available evidence. In particular, a substantial mismatch between the *expected answer* and the *model answer* may indicate either that the generated question is ambiguous or unreasonable, or that the *model answer* is incorrect. The original event description provides additional grounding information that allows the *Verifier* to distinguish between these cases and assess the overall validity of the question–answer pair. For verification, we launch two parallel workers, each serving one Qwen3.5-397B-A17B instance with vLLM on four H200 GPUs.

Filtering and Post-Processing. We discard question–answer pairs for which the *Verifier* identifies an inconsistency between the *expected answer* and the *model answer*, determines that the *model answer* is incorrect, or judges the generated question to be invalid, ambiguous, or otherwise unreasonable. After filtering, we group questions associated with the same video and question timestamp into multi-turn conversations, which are subsequently used for training. We also include annotations from the offline video understanding from LLaVA-Video-178K to further strengthen the memory mechanism.

B.3 Dataset Statistic

MaLoW-Instruct contains 381K instruction samples constructed from 56K videos. Table 6 and Table 7 summarize the distribution of questions by timestamp class and question type. In terms of temporal categories, real-time questions constitute the largest proportion (46.1%), followed by past (36.4%) and future (17.4%) questions. These categories correspond to questions posed while the target event is occurring, after the event has occurred, and before the event occurs, respectively. Regarding question types, multiple-choice questions account for the largest proportion (35.0%), while open-ended (34.7%) and Yes/No (30.0%) questions are comparably represented, with counting questions comprising the remaining 14.0%.

Table 8 summarizes the statistics of the questions, answers, and video durations in the dataset. Questions contain an average of 70.4 tokens, encoded by the tokenizer from Qwen3.5-9B, while answers are generally concise, with an average of 3.8 tokens. The question timestamps cover a wide temporal range, with an average question video length of 57.2 seconds, while the full videos are longer on average at 89.5 seconds, reaching up to 1,673.4 seconds. The distribution of questions, answers, and video durations in the dataset are presented in Figure 6.

C Additional Experiment details

C.1 Main Experiments

In this section, we provide additional experiment details for the main experiments presented in Table 1 and the full comparison in Table 16.

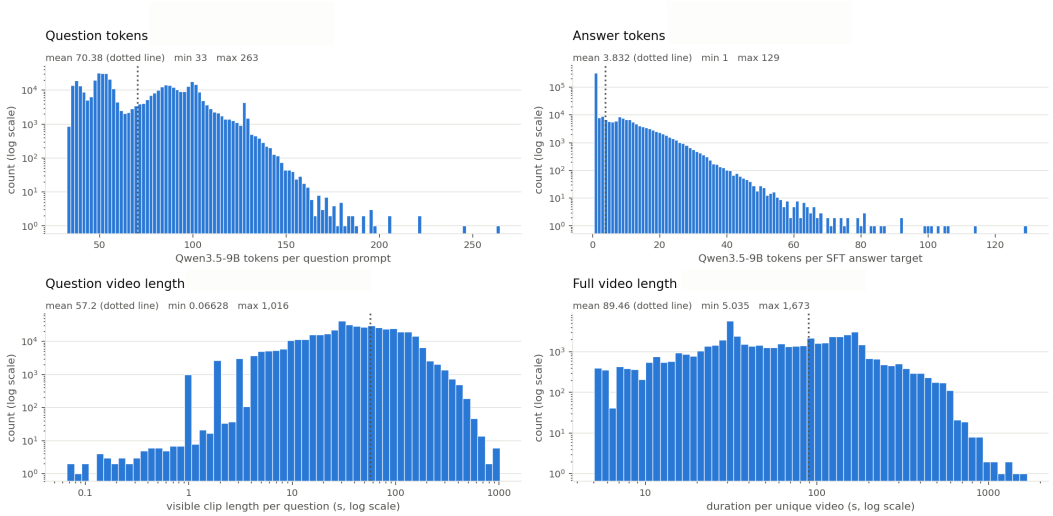


Figure 6: The distribution of questions, answers, and video durations in the dataset.

Base Models. In the main experiments, we build MaLoW on top of three different models, VideoChat3-4B (Li et al., 2026), Qwen3.5-9B (Qwen Team, 2026), and MOSS-VL-Realtime-11B (Wang et al., 2026). VideoChat3 is a video-centric VLM that is further fine-tuned on streaming video understanding data and adopts a full self-attention architecture. Qwen3.5 is a general-purpose VLM based on a hybrid attention architecture, where a self-attention layer is interleaved with every four linear-attention layers. MOSS-VL is a video-centric VLM, which adapts cross-attention layers to inject visual information into the text tokens, and a cross-attention layer is interleaved with every four self-attention layers.

Video Preprocessing. During training and evaluation, we sample the video frames at 1 fps, dividing the video frames into non-overlapping two-frame temporal blocks (each C_t is a 2-second video clip). We regard the most recent 2-second block as the current observation, which will be sent into the VLMs, while all the rest of the history frames will be sent into the MaLoW to be converted into LoRA weights. We preprocess the video frames with the preprocessor from the corresponding VLMs, with the maximum per-frame pixel budget set to 262,144 for all three versions of MaLoW.

Vision Encoder. We use the vision encoder from the corresponding VLMs as the vision encoder to encode each of the temporal blocks into vision tokens. We also use the MLP layers or adaptor from the VLMs to convert the vision tokens into the text semantic space before we send them into the Aggregator. We denote the vision tokens from the temporal block at t as V_t.

Aggregator. Visual tokens V_t in the temporal block at t are concatenated with the learnable query q and the corresponding text embedding T_t of the timestamp of the temporal block. These tokens are first projected to a hidden dimension of 512 and processed by a four-layer Transformer encoder aggregator with eight attention heads (64 dimensions per head), a feed-forward dimension of 2,048, pre-layer normalization, GELU activations. We learn one 512-dimensional query for each of the LLM decoder layers. The query is shared across target modules, and LoRA ranks in the same LLM decoder layer.

State-Space Model. We learn a four-layer DeltaNet (Yang et al., 2024) with a hidden dimension of 512 as the state-space model. Each DeltaNet layer uses four value heads and two key heads, both with head dimension 128.

Cosine-Similarity-Based Gating. We determine the gating threshold by profiling the cosine similarity score between two adjecent block from 1000 subset of video from the training data introduced in section 3.4. Across 64,426 transitions between consecutive temporal blocks, the median raw cosine distance, $1 - \cos(\bar{V}_t, \bar{V}_{t-1})$, is 0.0542. We therefore set $\beta = 0.05$, approximately the median-change operating point. The first temporal block is always retained, while each subsequent block is retained if $\cos(\bar{V}_t, \bar{V}_{t-1}) \leq \beta$. Blocks exhibiting less visual change than this threshold receive zero weight in the residual-history aggregation, whereas the retained blocks are uniformly averaged.

The Converter. For each target-module type and each LoRA rank, a separate linear output head maps the 512-dimensional representation $Norm(o_t) + \alpha Norm(s_t)$ to a vector of dimension $d_{\text{in}} + d_{\text{out}}$. The vector is split into one row of the LoRA factors $\mathbf{A}_t \in \mathbb{R}^{r \times d_{\text{in}}}$ and $\mathbf{B}_t \in \mathbb{R}^{r \times d_{\text{out}}}$. These output heads are shared across decoder layers, while layer specificity is supplied by the layer-specific learned queries. Finally, the LoRA at a particular slot s at timestamp t is calculated with $\Delta\theta_t = \alpha\, B_t^{\mathsf{T}} A_t$, where α is lora_alpha, which we set to 16.

LoRA Application Location. For VideoChat3-4B, we apply rank-8 generated LoRA for the MLP projections (`gate_proj`, `up_proj`, and `down_proj`) in every LLM layer. For Qwen3.5-9B, we apply rank-8 generated LoRA for the MLP projections (`gate_proj`, `up_proj`, and `down_proj`) in every LLM layer, `self_attn.v_proj` in full-attention layers, and `linear_attn.in_proj_qkv` in linear-attention layers. For MOSS-VL-Realtime-11B, we apply rank-8 generated LoRA for the MLP projections (`gate_proj`, `up_proj`, and `down_proj`), and `v_proj` in every LLM layer. We inject the generated LoRA weights into the target linear layers of the frozen base model through customized forward functions.

Training Details. We build our training pipeline on top of the Llama-Factory (Zheng et al., 2024). We train MaLoW on the MaLoW-Instruct dataset introduced in Section 3.4 for 2,800 optimization steps with a total batch size of 64. Training uses four H200s, a per-device batch size of one, and 16 gradient-accumulation steps. We use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, an peak learning rate of 8×10^{-5}, a cosine learning-rate schedule, and a warm-up ratio of 0.1. Training is performed in bfloat16. We set the LoRA regularization factor γ to 0.01.

During training, we use the state-space model in its parallel causal form. For each sample, the compact vision representations q_t are arranged in chronological order and processed by the four-layer DeltaNet using the chunk-parallel gated-delta-rule kernel and an attention mask for padded blocks. We then select the output at the last valid temporal block for each history observation. This parallel computation is causally equivalent to processing the blocks recurrently while carrying the convolutional and recurrent DeltaNet states, which is the form that is used for online streaming inference.

Evaluation Details. Following the evaluation protocols adopted in prior streaming video understanding works (Shen et al., 2026; Zhang et al., 2026a), we evaluate MaLoW on the *Real-Time* split of StreamingBench (Lin et al., 2026), as well as the *Real-Time* and *Backward* splits of OVO-Bench (Niu et al., 2025).

For StreamingBench, we evaluate on the *Real-Time* split, which contains 2,500 multiple-choice questions, and use accuracy as metric. For OVO-Bench, both the *Backward* and *Real-Time* splits are evaluated with the metric accuracy. The *Backward* and *Real-Time* splits consist of 631 and 837 questions. Following prior work, the performance of each split is computed as the macro-average across the tasks belonging to that split, and we additionally report the macro-average performance across the *Backward* and *Real-Time* splits in Table 1 and Table 16.

The *Backward* split of OVO-Bench comprises three tasks: Episodic Memory (EPM), Action Sequence Identification (ASI), and Hallucination Detection (HLD). Among them, EPM is particularly well suited for evaluating the memory mechanism introduced by MaLoW, as it directly measures the model's ability to retain and retrieve information from earlier observations. We further note that HLD is relatively less challenging because all ground-truth responses are *Unable to answer*. As a result, models can achieve high accuracy on this task by consistently predicting this response, making HLD less discriminative for assessing the effectiveness of long-term memory mechanisms.

The trained MaLoW can operate in a fully streaming manner by processing the video history block by block using the cached-forward mode of DeltaNet. (Please see the *Evaluation details.* in the section C.2) To accelerate evaluation, however, we use the same parallel-forward mode employed during training when evaluating on OVO-Bench and StreamingBench. Specifically, for a question associated with a video prefix $[0, T]$, we treat the interval $[0, T-2]$ as historical context and sample it at 1 fps as input to the hypernetwork. The base VLM then receives the question together with two frames sampled from the current observation window $[T-2, T]$. For the evaluation on the OVO-bench, we follow the prompt example used in lmms-eval (Zhang et al., 2025a), and for the evaluation on the StreamingBench we follow the prompt example used in the official repository.

Baselines Re-Evaluation. Among the baselines in Table 1 and the full comparison in Table 16, we re-evaluate MOSS-VL-Realtime-11B (Wang et al., 2026), VideoChat3-4B (Li et al., 2026), and Qwen3.5-9B (Qwen Team, 2026), as these models are also used as the base VLMs for MaLoW. For the remaining baselines, we report the results from their original papers.

For Qwen3.5-9B, we sample video frames at 1 fps with a maximum per-frame pixel budget of 262,144 and use greedy decoding for answer generation.

For VideoChat3-4B, we use the official evaluation code and follow the hyper-parameters reported in the paper. On OVO-Bench, we sample frames at 2 fps and retain the 32 most recent frames, with a per-frame pixel budget of 448^2 and a visual-token budget of 80K. On StreamingBench, we sample frames at 4 fps with at most 4,096 frames, using the same per-frame pixel budget of 448^2 and visual-token budget of 80K.

For MOSS-VL-Realtime-11B, as neither the paper nor the released repository provides an official evaluation script, we evaluate the model on OVO-Bench and StreamingBench using frames sampled at 1 fps with a maximum per-frame pixel budget of 262,144.

These settings successfully reproduce the reported OVO-Bench performance of VideoChat3 and MOSS-VL-Realtime. However, our re-evaluated StreamingBench results are lower than those reported in the corresponding papers. We therefore include both the re-evaluated and paper-reported results in Table 1 and Table 16, with the paper-reported results shown in gray.

C.2 MaLoW in VLA setting

In this section, we provide more implementation details for the MaLoW in VLA setting. The result presented in table 2.

Base VLA Model and Initialization. Inspired by the previous work (Zhang et al., 2026b), we train a simple VLA model on top of the Qwen3.5-2B, named as Q3.5-2B-MLP as the starting point of the MaLoW (Q3.5-2B-MLP).

The Q3.5-2B-MLP is given current observation (an image) and text instruction and predicts a chunk of four seven-dimensional actions, corresponding to the current action and the following three actions. We learn special tokens as action hidden representation and VLMs compress the vision observation and the text instruction into the special tokens. An action decoder which is a two-layer MLP that directly regresses normalized robot actions using mean-squared error base on the special tokens. We use the Qwen3.5 preprocessor to preprocess the input image and the text instruction.

Q3.5-2B-MLP was trained on the Bridge and RT-1 mixture of the OXE LeRobot dataset (O'Neill et al., 2024) for 50,000 optimization steps, with learning rate is 4×10^{-5} on action decoder, and 1×10^{-4} on LoRA. The learning rate is linearly warmed up for 5,000 steps and then cosine-decayed. Training uses AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$,and weight decay 10^{-8}. Training is conducted in bfloat16 on four H200 GPUs with a per-device batch size of 16 and two gradient-accumulation steps, yielding a global batch size of 128. We apply rank-128 LoRA to all linear layers, including the vision tower and multimodal projector, with a scaling factor of 256. Q3.5-2B-MLP achieves 59.4% in the Simpler-Bridge with WidowX robot.

Observation Preprocessing. For a training sample at control step t, the base VLA Model receives the current observation x_t and the language instruction, while the MaLoW receives all preceding observations $x_0, \ldots, x_{t-1}$. We do not impose a maximum history length. Bridge trajectories are recorded at 5 Hz, and timestamps are therefore expressed using a control period of 0.2 seconds.

The current observation is processed through the same single-image preprocessing path as the Q3.5-2B-MLP, while all the historical observations are decoded directly from the episode video and be sent into the MaLoW. All the frames are resized to 224×224, and passed to the Qwen3.5-2B preprocessor. This matches the preprocessing of the historical frames received from SimplerEnv during interactive evaluation.

We divide the observation history into consecutive, non-overlapping two-frame blocks, matching the temporal patch size of the Qwen vision encoder. If the history contains an odd number of frames, its final frame is repeated to complete the last block. For example, the history at step $t = 3$ is

represented as $[x_0, x_1]$ and $[x_2, x_2]$. Each block is assigned a timestamp interval using the true Bridge control clock. A committed block j is labeled $[2j\Delta t, (2j+2)\Delta t]$, where $\Delta t = 0.2$ seconds.

Vision Encoder. We use a frozen copy of the vision tower and vision adaptor from the Q3.5-2B-MLP as the MaLoW's vision encoder.

Action-History Conditioning. In addition to visual history, every historical frame is paired with the action that produced it. Specifically, frame x_i is paired with the preceding action a_{i-1}. The first frame x_0 is paired with a learned empty-action embedding representing a_{-1}. Historical actions undergo the same normalization transformations as the action targets.

Each seven-dimensional action is mapped to the visual-token dimension by a two-layer MLP with a GELU activation. Thus, the input sequence for a two-frame temporal block is

$$[V_{2j}, V_{2j+1}, T_j, E(a_{2j-1}), E(a_{2j})],$$

where V_i denotes the visual tokens for frame i, T_j denotes the frozen token embeddings of the block's timestamp interval, and $E(\cdot)$ is the action-history MLP.

Aggregator. For every temporal block, the visual, timestamp, and action-history tokens are projected to a hidden dimension of 512 and processed by a four-layer Transformer encoder. The rest of the aggregator's details are the same as the one used in section C.1.

State-Space Model and the Converter. The state-space model and the converter share the same details as the one used in section C.1.

LoRA Application Locations. Because Qwen3.5 interleaves full self-attention and linear-attention layers, we explicitly construct only valid layer–module combinations. The rank 8 generated LoRA is applied to `gate_proj`, `up_proj`, and `down_proj` in every decoder layer, to `self_attn.v_proj` in full-attention layers, and to `linear_attn.in_proj_qkv` in linear-attention layers.

Training Details. MaLoW (Q3.5-2B-MLP) is trained on the Bridge portion of the OXE LeRobot dataset. Before training, we merge the Q3.5-2B-MLP LoRA adapter into Qwen3.5-2B and load the stage-1 action-decoder weights. The merged Qwen backbone is then frozen. The trainable components in the MaLoW (Q3.5-2B-MLP) are the MaLoW, including the aggregator, action-history encoder, temporal state-space model, and LoRA output heads, together with the action decoder. The complete objective is

$$\mathcal{L} = \mathcal{L}_{\text{action}} + 0.1\,\mathcal{L}_{\text{LoRA}},$$

where

$$\mathcal{L}_{\text{LoRA}} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} (\text{mean}\,|\mathbf{A}_s| + \text{mean}\,|\mathbf{B}_s|)$$

, $\mathcal{S}$ is the set of valid generated LoRA slots, and $\mathcal{L}_{\text{action}}$ is MSE Loss between the predicted actions and the ground truth actions.

We train for 25,000 optimization steps using four H200 GPUs. The per-device batch size is four and gradients are accumulated for four steps, giving a total batch size of 64. We use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, and weight decay 10^{-8} and learning rate of 1×10^{-5} in this experiment. We use a cosine schedule with 1,500 warm-up steps and a final learning rate of 5×10^{-7}. Training is performed in bfloat16 with FlashAttention-2 and ordinary distributed data parallelism.

Evaluation Details. We evaluate MaLoW (Q3.5-2B-MLP) on the four WidowX Bridge tasks in SimplerEnv: Stack Green Block on Yellow Block, Put Carrot on Plate, Put Spoon on Towel, and Put Eggplant in Yellow Basket. Each task contains 24 evaluation episodes, for a total of 96 episodes.

Evaluation is conducted in a fully interactive streaming setting. At environment step zero, no historical context is available, and the model exactly reproduces the Q3.5-2B-MLP policy. At each subsequent step, completed two-frame blocks and the corresponding actions are permanently incorporated into the cached DeltaNet state m_t and updated into the global sematic s_t. If the available history ends with a single unpaired frame, we construct a provisional block by duplicating that frame and evaluate it using a temporary copy of the state; this provisional block is replaced once the next observation becomes available. The h_t state is reset at each episode boundary.

The policy predicts four-action chunks and applies action ensembling over a horizon of four steps. Because ensembling may cause the action executed by the environment to differ from the most recent raw model prediction, we feed the normalized executed action back to the policy server. This executed action, rather than the unensembled model prediction, is paired with the corresponding observation when constructing subsequent history blocks.

C.3 MaLoW with different VLM scales and sources

In this section, we provide additional implementation details for the results presented in Table 4.

Base Models. In this sections, we use the Gemma-E4B (Gemma Team, 2026) and Qwen3.5-0.8B, Qwen3.5-2B and Qwen3.5-4B (Qwen Team, 2026) as the base model.

LoRA Application Location. For Gemma-E4B, we apply rank-8 generated LoRA for the MLP projections (`gate_proj`, `up_proj`, and `down_proj`) in every LLM layer. For Qwen3.5 series, we apply rank-8 generated LoRA for the MLP projections (`gate_proj`, `up_proj`, and `down_proj`) in every LLM layer, `self_attn.v_proj` in full-attention layers, and `linear_attn.in_proj_qkv` in linear-attention layers.

Data, Structural, and Training Details. For the Video Preprocessing, Vision Encoder, Aggregator, State-Space Model, Cosine-similarity-based gating, the Converter, and Training details are share the same details in section C.1.

Evaluation Details. For the evaluation on the OVO-bench, we follow the example used in lmms-eval (Zhang et al., 2025a), and for the evaluation on the StreamingBench we follow the example used in the official repository. For a question associated with video prefix $[0, T]$, the interval $[0, T-2]$ is treated as historical context and sampled at 1 fps for the MaLoW. The base VLM receives the question together with two frames sampled from the current observation $[T-2, T]$.

For OVO-Bench, we evaluate all backward, real-time, and forward splits. Backward and real-time questions are scored by multiple-choice accuracy. For the forward split, repetitive-event counting is scored by exact count matching, while sequential-step recognition and contextual-readiness recognition are scored using exact yes/no accuracy. We follow the benchmark's macro-averaging protocol across tasks for each split, and we further present the macro-averaging number over 3 splits.

For StreamingBench, we evaluate the Real, Omni, Sequential-QA, and Proactive subsets base on how the benchmark annotations are organized. Real, Omni, and Sequential-QA are scored by multiple-choice accuracy; following the original protocol, Sequential-QA includes the gold question–answer history from earlier timestamps in the same video. For Proactive evaluation, the model is queried once per second until four seconds after the annotated trigger time. An example is counted as correct only when the model triggers within this tolerance window, and its generated response contains the expected output. The overall StreamingBench score is the benchmark-weighted average over Real, Omni, Proactive, and Sequential-QA, using weights of 2,500, 1,500, 250, and 250 examples, respectively.

C.4 Memory stress test

In this section, we share additional implementation details for the memory stress test evaluation and the results presented in Table 3.

We use the StreamArena (Zhang et al., 2026d) Historical Retrospection (HR) split as the benchmark, which has 877 questions across L1–L4 over 237 videos. L1 to L4 indicate the evidence-to-query gap of less than 5 min, between 5 and 15 min, between 15 and 30 min, and longer than 30 min, respectively.

We compare MaLoW (Qwen3.5-9B) against strong streaming video understanding models: MOSS-VL-Realtime-11B (Wang et al., 2026), which retains the full historical context; VideoChat3-4B (Li et al., 2026) and AURA-8B (Lu et al., 2026), which operates with a local 16 / 30 -second observation window, respectively; VST-7B (Guan et al., 2026), which compresses the historical observation as text-based memory.

For AURA and VST, we follow the official implementations in StreamArena to evaluate the model performance. For MOSS-VL-Realtime, VideoChat3, and MaLoW, they share the same question

prompt derived from the prompt used by AURA. MaLoW (Qwen3.5-9B) samples video frames at 1fps with a maximum per-frame pixel budget of 262,144. The final two-second window is regarded as the current observation and will be sent to the VLMs, while the rest of the history frames will be sent to the MaLoW and converted into LoRA. The VideoChat3 uses 32 unique frames at 2 FPS from the latest 16 seconds, with an aspect-ratio-preserving maximum of 200,704 pixels per frame (448^2); MOSS-VL samples frames at 1 FPS and with maximum per-frame pixel cap as 262,144 pixels; and VST uses up to 384 causal-prefix frames at 1 FPS with aspect-ratio-preserving resolution determined dynamically by its 8,192-visual-token-per-segment budget. All methods use their own preprocessor to preprocess the video frames.

C.5 Streaming Inference Speed Profiling

We provide additional details on the inference-speed profiling reported in Figure 4. We measure the streaming inference latency of MaLoW and four baselines on 100 deterministically sampled OVO-Bench (Niu et al., 2025) questions spanning a wide range of history lengths. Video durations range from 8 to 1,117 seconds, with a median of 153.5 seconds, a mean of 193.3 seconds, and a 90th percentile of 312 seconds. This set includes both short clips and videos exceeding 10 minutes, enabling evaluation of long-context streaming efficiency. At each query timestamp t, we measure synchronized GPU computation required to process the current observation and answer the query under each method's native streaming protocol. Video decoding, resizing, and other input preprocessing are excluded.

For MaLoW, each streaming step includes encoding the current observation, updating the hidden state and generating the LoRA adapter for the next time step, applying the adapter generated for the current time step, prefilling the model with the question and the two most recent frames from $[t-2, t]$, and decoding the first output token.

For SimpleStream (Qwen3.5-9B), the model receives only the question and two frames sampled from the most recent two-second window, $[t-2, t]$, without earlier visual history. The measured latency includes prefilling with these inputs and decoding the first output token.

For Qwen3.5-9B with full context, the measured latency includes prefilling the question together with the complete video history from the start of the stream to t, sampled at 1 FPS, followed by decoding the first output token.

For VideoChat3, we follow its official streaming implementation. At each query, the model re-prefills the question together with a sliding window of the most recent 32 frames, corresponding to approximately 16 seconds at 2 FPS, before generating the streaming response. The measured latency includes prefilling of the local window video and text query and subsequent decoding.

For MOSS-VL-Realtime, we follow its official streaming implementation, in which incoming observations are continuously incorporated into a growing vision-text KV state. The measured latency includes the forward pass required to ingest the latest two seconds of video and the subsequent decoding.

D Additional Experiments

D.1 Ablation Study

We conduct ablation studies with MaLoW (Qwen3.5-9B) to investigate key design choices: (1) the LoRA target modules, (2) the LoRA rank, (3) the architecture of the MaLoW, (4) the size of the current observation and (5) the refresh rate of the hidden state. All ablations are conducted on a subset of an subset of the training data described in Section 3.4, rather than the final dataset used for the main experiments. Within each ablation, all training hyperparameters are kept fixed following Section 4.1, except for the component being ablated, ensuring a controlled comparison. We report results on OVO-Bench and StreamingBench, together with the Episodic Memory (EPM) results from OVO-Bench.

Where should LoRA be applied? Qwen3.5-9B adopts a hybrid attention architecture in which one self-attention layer is interleaved with every four linear-attention layers, providing a representative setting for studying LoRA placement in modern hybrid-attention VLMs. We evaluate three cumu-

Table 9: Ablation study on the modules to which LoRA is applied. Each row cumulatively adds LoRA to the module indicated in the first column.

Module	OVO-Bench (EPM)	StreamingBench	Avg.
MLP	63.6 (63.3)	72.4	68.0
+ `v_proj`	63.8 (63.6)	72.4	68.1
+ `qkv_proj`	64.1 (63.2)	72.5	**68.3**

Table 10: Ablation study on the learned LoRA rank.

Rank	OVO-Bench (EPM)	StreamingBench	Avg.
4	65.3 (63.0)	71.9	68.6
8	65.7 (63.6)	72.8	**69.2**
16	63.4 (59.3)	68.2	65.8

Table 11: Ablation study on the causal relationship and global semantics.

Variant	OVO-Bench (EPM)	StreamingBench	Avg.
w/ sum	62.5 (63.0)	70.9	66.7
w/ SSM	62.4 (60.9)	72.8	67.6
w/ SSM + sum	64.2 (63.0)	72.7	**68.5**

Table 12: Ablation study on different global semantics aggregation strategies.

Variant	OVO-Bench (EPM)	StreamingBench	Avg.
sum	64.1 (61.3)	72.1	68.1
weighted sum	64.6 (62.0)	72.7	68.6
gated sum	64.8 (64.0)	72.7	**68.8**

lative configurations: (1) applying LoRA only to the MLP layers shared by both self-attention and linear-attention blocks; (2) additionally applying LoRA to the `v_proj` in self-attention layers; and (3) further applying LoRA to the `qkv_proj` in linear-attention layers. As shown in Table 9, progressively extending LoRA to additional modules consistently improves the average performance, with the best results achieved when all three module types are adapted.

What is the best LoRA rank? The results are presented in Table 10. Among the evaluated configurations, rank 8 achieves the best overall performance, with an average score of 69.2. Rank 4 performs slightly worse, suggesting limited adaptation capacity, whereas increasing the rank to 16 causes a substantial performance drop on both benchmarks, indicating overfitting.

What is the best MaLoW module structure? We investigate key design choices for the MaLoW module: (1) whether temporal modeling with a State-Space Model (SSM) and global semantic aggregation are both necessary, and (2) how global semantics should be effectively preserved.

Table 11 evaluates the contributions of the SSM and global semantic aggregation. The *w/ sum* variant achieves stronger performance on EPM, which emphasizes information from earlier observations, whereas the *w/ SSM* variant performs better on StreamingBench, where a large portion of the evaluation focuses on real-time understanding. These results suggest that the SSM is particularly effective at modeling temporally local and causal information, while global semantic aggregation better preserves information over longer temporal horizons. Combining both components balances real-time understanding and long-term memory, yielding the best overall performance.

Table 12 further compares different strategies for preserving global semantics. We consider three variants: *sum*, which directly accumulates all previous compact visual representations q_t; *weighted sum*, which weights each q_t according to the cosine similarity between adjacent temporal blocks; and *gated sum*, which selectively retains q_t based on temporal similarity and is adopted in our final design. The *gated sum* achieves the best average performance. By filtering temporally redundant representations while retaining informative observations, the gating mechanism preserves global semantics more effectively and improves overall performance.

How does the size of the current observation affect performance? Although the main experiments in Section 4.1 treat each C_t as a 2-second video clip, updating the hidden state h_t every 2 seconds, the observation window can be varied independently of update frequency. To isolate the effect of the current observation window while keeping the hidden-state update interval fixed, we ablate the window sizes of 1, 2, and 4 seconds, with 2 seconds corresponding to our default setting. For the 4-second window, h_t is updated using the oldest 2-second segment evicted from the window. For the 1-second window, MaLoW accumulates consecutive observations and updates h_t after every 2 seconds of newly observed video. The result is presented in Table 13. A larger 4-second current observation window improves real-time perception, as reflected by the higher StreamingBench score, but degrades memory performance, with EPM dropping to 59.9. In contrast, a 1-second window reduces real-time perception while yielding similar EPM performance as the 2-second window. To balance real-time perception and long-term memory, we therefore use a 2-second current observation window as the default setting.

Table 13: Ablation study on the current observation size; frames are sampled at 1fps.

Variant	OVO-Bench (EPM)	StreamingBench	Avg.
1 second (1 frame)	66.0 (63.2)	71.0	68.5
2 seconds (2 frames)	66.3 (63.3)	73.1	69.7
4 seconds (4 frames)	65.4 (60.0)	75.1	**70.3**

Table 14: Ablation study on the hidden state (h_t) refresh rate; frames are sampled at 1fps.

Variant	OVO-Bench (EPM)	StreamingBench	Avg.
1 second (1 frame)	65.2 (62.8)	73.2	69.2
2 seconds (2 frames)	66.3 (63.3)	73.1	**69.7**
4 seconds (4 frames)	66.5 (63.6)	72.3	68.4

Table 15: Breakdown of the MaLoW (Qwen3.5-9B) latency. Percentages are relative to the total latency of 152.89 ms.

Component	**Latency (ms)**	**Percentage**
Could be paralleled with the base VLMs prefilling and decoding		
Encode new frames and update state h_t	12.11	7.91%
Generate LoRA tensors A_{t+1} and B_{t+1}	3.27	2.14%
Combine LoRAs and generate $\Delta\theta_{t+1}$	5.69	3.72%
Can not be paralleled with the base VLMs prefilling and decoding		
Apply LoRA $\Delta\theta_t$	1.13	0.74%
Wrapper overhead	0.62	0.41%
Base-model prefill + decoding first tokens	130.07	85.07%
Total	**152.89**	**100.00%**

How does the refresh rate of the hidden state affect the performance? Although the main experiments in Section 4.1 treat each C_t as a 2-second video clip, updating the hidden state h_t every 2 seconds, the update frequency can be varied independently of the observation window. We therefore conduct an ablation with update intervals of 1, 2 (which used in the main experiments), and 4 seconds while keeping the base VLM input fixed to the most recent 2 seconds. For a 1-second interval, the hidden state is updated using the oldest 1-second segment evicted from the current 2-second observation window; for a 4-second interval, MaLoW accumulates newly observed video and updates the hidden state only after every 4 seconds of new observations.

The results are presented in Table 14. We observe that less frequent updates, where each update incorporates a larger temporal chunk, slightly improve memory performance at the cost of a small drop in real-time perception. We hypothesize that this behavior is related to the State-Space Model (SSM) used for hidden-state updates. As shown in Table 11, the SSM primarily models short-term temporal dependencies. More frequent updates may therefore bias it toward shorter temporal ranges, improving current perception but slightly weakening longer-term memory, while less frequent updates have the opposite effect. Overall, performance remains stable across different update intervals. We therefore adopt a 2-second interval, which balances memory and real-time perception while matching the current observation window for a simpler design.

D.2 Additional Analysis

Latency Breakdown. We further analyze the latency of MaLoW (Qwen3.5-9B) reported in Figure 4, with the detailed breakdown presented in Table 15. Encoding the new frames and updating the hidden state h_t, generating the LoRA tensors A_{t+1} and B_{t+1}, and combining them to construct $\Delta\theta_{t+1}$ account for 7.9%, 2.1%, and 3.7% of the total latency, respectively. Importantly, these operations compute the adapter for the next temporal block and can therefore be executed in parallel with applying $\Delta\theta_t$ and performing base-VLM prefilling and decoding for the current block. With such pipelining, only the application of $\Delta\theta_t$ and the wrapper overhead remain on the critical inference path, adding 1.75 ms to the 130.07 ms base-model inference latency. This corresponds to approximately 1.3% of the effective end-to-end latency, demonstrating that MaLoW introduces only marginal latency overhead in practical streaming inference.

LoRA Hidden State Evolution over the History. Figure 7 visualizes how the internal LoRA state of MaLoW evolves with changes in video content. We compute a V-JEPA-2 (Assran et al., 2025) surprise score on a YouCook2 (Zhou et al., 2018) video by masking the features of the last temporal block and measuring the distance between its predicted representation and the representation of the subsequent block. In parallel, we track the 512-dimensional LoRA hidden state of MaLoW

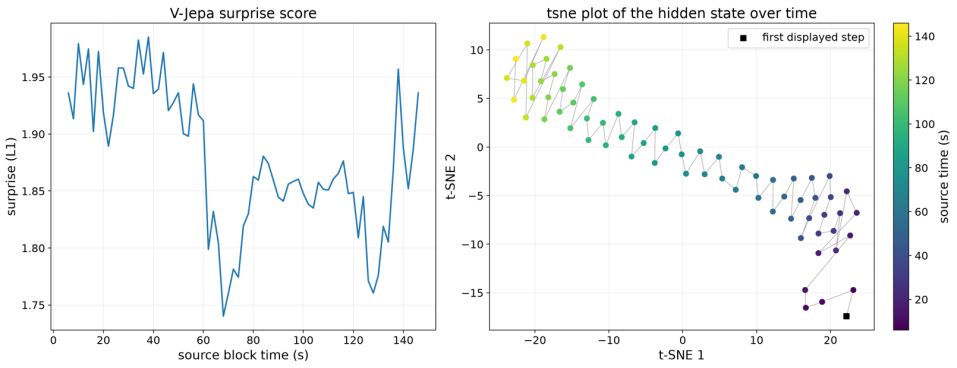


Figure 7: The V-JEPA-2 surprise score versus the t-SNE map of the LoRA hidden state.

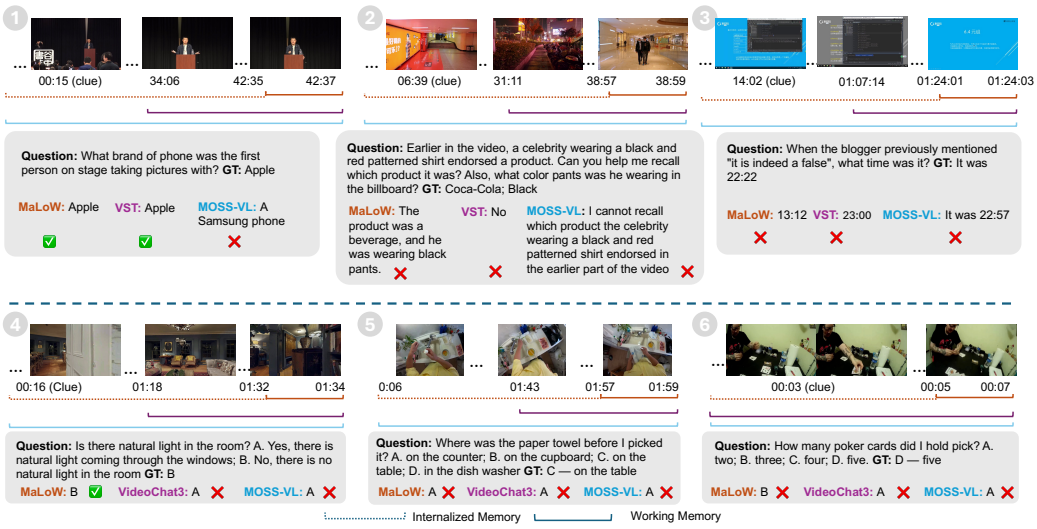


Figure 8: The additional qualitative results.

(Qwen3.5-9B), defined as $\mathrm{Norm}(o_t) + \alpha\mathrm{Norm}(s_t)$. We average the hidden states used to generate layer-specific LoRA adapters and project the resulting representation into two dimensions using t-SNE.

The LoRA state changes more rapidly when the V-JEPA-2 surprise score is high (before 60s and after 130s), while evolving more smoothly during the lower-surprise interval between 60s and 130s. This correspondence suggests that the LoRA state adapts dynamically to changes in the underlying video events.

Additional Qualitative Results. We provide additional qualitative results in Figure 8. Samples 1, 2, and 4 further demonstrate that MaLoW (Qwen3.5-9B) can retain long-term historical context through its internalization mechanism. In some cases, however, the retained memory can be vague, as humans do. For example, in Sample 2, MaLoW appears to retain a coarse memory of Coca-Cola and answers "beverage" rather than the specific object.

Sample 5 illustrates a failure case in which both MaLoW and the baselines produce incorrect answers, partly due to an inaccurate ground-truth annotation. Sample 3 requires not only remembering the relevant video timestamp, but also an agentic procedure to revisit the corresponding frames and localize the clock displayed in the bottom-right corner. Since MaLoW is not trained with agentic tool-use data, extending it to such agentic settings is left for future work. Finally, Sample 6 requires strong visual perception to recognize that four cards are initially present on the table, followed by temporal reasoning to determine that one additional card is added, yielding five cards in total. We expect such cases to benefit from future base VLMs with stronger fine-grained visual perception.

Table 16: Full results in streaming video understanding. *VideoChat3 uses 2 fps on OVO-Bench and 4 fps on StreamingBench. Gray numbers denote results reported in the original paper that differ from our re-evaluation using the official implementation and reported evaluation hyperparameters; we suspect an inconsistency in the released hyperparameters; please refer to *Baselines Re-evaluation* in sec C.1 for more information. MaLoW (MOSS-VL-Real-11B) achieves state-of-the-art performance on OVO-Bench, while MaLoW (Qwen3.5-9B) performs best on the episodic memory (EPM) task.

Model	#Frames	Streaming Bench (R)	*OVO-Bench*											Avg.
			Real-Time Visual Perception							*Backward Tracing*				
			OCR	ACR	ATR	STU	FPD	OJR	Avg.	**EPM**	ASI	HLD	Avg.	
Offline Video LLMs														
LLaVA-Video-7B (Zhang et al., 2025b)	64	–	69.1	58.7	68.8	49.4	74.3	59.8	63.5	56.2	57.4	7.5	40.4	52.0
LongVU-7B (Shen et al., 2025)	1 fps	–	55.7	49.5	59.5	48.3	68.3	63.0	57.4	43.1	66.2	9.1	39.5	48.5
Qwen3.5-9B (Qwen Team, 2026)	1 fps	81.9	83.9	62.4	70.7	46.1	76.2	70.6	68.3	61.6	62.8	59.1	61.2	64.8
Streaming Video LLMs with full context														
VideoLLM-online-8B (Chen et al., 2024a)	2 fps	35.99	8.1	23.9	12.1	14.0	45.5	21.2	20.8	22.2	18.8	12.2	17.7	19.3
Streamo-7B (Xia et al., 2026)	1 fps	–	79.2	57.8	75.0	49.4	64.4	70.1	66.0	54.6	52.0	31.7	46.1	56.1
VideoChat3-4B (Li et al., 2026)	2-4 fps*	79.0 (83.0)	92.6	78.0	77.6	59.0	74.3	71.7	75.5	58.8	58.6	46.8	54.7	65.1
MOSS-VL-Real-11B (Wang et al., 2026)	1 fps	80.6 (82.9)	86.6	70.6	84.8	64.6	71.2	73.9	75.2	60.6	64.5	92.4	**72.5**	73.8
Streaming Video LLMs with Local Context Window														
Mage-VL-4B (Yang et al., 2026)	4 (1 fps)	–	94.6	83.5	79.3	74.2	70.3	77.2	79.8	50.2	61.5	32.8	48.2	64.0
Simplestream-8B (Shen et al., 2026)	4 (1 fps)	78.3	89.3	77.1	83.6	68.5	76.2	81.0	**79.3**	49.5	56.1	54.8	53.5	66.4
AURA-8B (Lu et al., 2026)	30 (1 fps)	**83.2**	89.9	79.8	80.2	70.2	77.2	81.5	79.8	54.9	67.6	58.6	60.4	70.1
Streaming Video LLMs with Local Context Window and Memory Mechanism														
Flash-VStream-7B (Zhang et al., 2024)	1 fps	23.23	24.2	29.4	28.5	33.7	25.7	28.8	28.4	39.1	37.2	5.9	27.4	27.9
Dispider-7B (Qian et al., 2025)	1 fps	67.63	57.7	49.5	62.1	44.9	61.4	51.6	54.6	48.5	55.4	4.3	36.1	45.4
TimeChat-Online-7B (Yao et al., 2025)	1 fps	75.28	75.2	46.8	70.7	47.8	69.3	61.4	61.9	55.9	59.5	9.7	41.7	51.8
StreamForest-7B (Zeng et al., 2026)	1 fps	77.26	68.5	53.2	71.6	47.8	65.4	60.9	61.2	58.9	64.9	32.3	52.0	56.6
JoyAI-VL-Interaction-9B (Yao et al., 2026)	1 fps	–	72.5	61.5	75.0	59.0	78.2	64.1	68.4	62.0	70.9	12.9	48.6	58.5
HERMES-7B (Zhang et al., 2026a)	1 fps	79.44	85.2	64.2	71.6	53.4	74.3	65.2	69.0	48.5	62.2	37.6	49.4	59.2
ThinkStream-3B (Liu et al., 2026)	2 fps	75.0	85.2	64.2	69.8	49.4	69.3	64.1	67.0	53.9	59.5	43.6	52.3	59.7
VST-7B (Guan et al., 2026)	2 fps	79.5	80.5	55.1	72.4	55.1	76.2	64.1	67.2	56.9	64.9	48.4	56.7	62.0
Ours														
MaLoW (VideoChat3-4B) - 197.6M trainable (2.2%)	1 fps	81.2	89.9	70.6	80.2	60.7	76.2	75.3	75.5	60.2	58.1	54.8	57.7	66.6
MaLoW (Qwen3.5-9B) - 308.9M trainable (3.0%)	1 fps	82.3	87.9	66.1	81.0	60.1	84.2	77.7	76.2	**63.6**	66.2	69.4	66.4	71.3
MaLoW (MOSS-VL-Real-11B) - 321.1M trainable (1.4%)	1 fps	81.0	91.9	74.3	85.3	62.9	73.3	76.6	77.4	62.0	62.8	90.3	71.7	**74.5**

D.3 Full Main Tables

In this section, we provide the full Table 1 in the main paper here by extending the baselines.

We categorize the baselines into three groups according to how they manage historical visual context during streaming inference. **1. Streaming Video LLMs with Full Context.** This category includes VideoLLM-Online (Chen et al., 2024a), Streamo (Xia et al., 2026), VideoChat3 (Li et al., 2026), and MOSS-VL-Realtime (Wang et al., 2026). These methods retain the historical context during inference, resulting in an ever-growing context as the video stream progresses and, consequently, substantial memory and computational overhead for long-duration streams. In particular, VideoChat3 adopts a local context window for OVO-Bench while using the full historical context for StreamingBench. **2. Streaming Video LLMs with a Local Context Window.** This category includes Mage-VL (Yang et al., 2026), SimpleStream (Shen et al., 2026), and AURA (Lu et al., 2026). These methods retain only the most recent observations within a fixed local window, ranging from approximately 4 to 30 seconds. Historical observations that fall outside the local window are directly evicted. **3. Streaming Video LLMs with a Local Context Window and Memory Mechanism.** This category includes Flash-VStream (Zhang et al., 2024), Dispider (Qian et al., 2025), TimeChat-Online (Yao et al., 2025), StreamForest (Zeng et al., 2026), JoyAI-VL (Yao et al., 2026), HERMES (Zhang et al., 2026a), ThinkStream (Liu et al., 2026), and VST (Guan et al., 2026). These methods retain the most recent observations in a local context window while compressing historical information outside the window into a more compact memory representation, such as textual summaries or condensed visual features.