VICOSTREAM: STREAMING VIDEOLLMS CAN RUN BEYOND 100 FPS WITH STAGE-WISE COORDINATED INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Streaming VideoLLMs must continuously process incoming video while maintaining low query latency, making both video-ingestion throughput and query-time responsiveness critical for real-time deployment. Existing methods largely focus on accelerating individual modules, such as visual encoding, token pruning, or KV-cache compression, but provide limited insight into whether the resulting system can sustain real-time streaming performance. We formulate streaming VideoLLM inference as a coordinated pipeline spanning visual preprocessing, visual encoding, token dropping, and LLM prefilling/decoding. Building on this formulation, we propose **ViCoStream** (**Vi**deo **Co**ordinated **Stream**ing), a stage-wise coordinated streaming framework that combines chunk-wise execution, CUDA-stream overlap, visual token control, bounded visual attention, and query-side retrieval to bound per-chunk computation and memory costs. We further provide a systematic study of bottleneck migration, revealing how chunk size, token retention, attention locality, and retrieval scope shape the throughput–accuracy trade-off. Experiments with Qwen2.5-VL-3B/7B-Instruct across multiple streaming benchmarks show that ViCoStream achieves 134 FPS video throughput and < 50ms TTFT on a single NVIDIA A100 GPU while maintaining accuracy close to full-history baselines.

1 INTRODUCTION

Large VideoLLMs have achieved remarkable success across diverse video understanding tasks Chatterjee et al. (2025); Fu et al. (2025); Zhang et al. (2024c); Ma et al. (2026); Zhao et al. (2025); Mu et al. (2023); Xu et al. (2024; 2025); Ma et al. (2024), but they are commonly evaluated with complete, offline videos. In real-world deployments, videos often arrive continuously as streams, and user queries may be issued before the full video has been observed Chen et al. (2024); Qian et al. (2024); Tong et al. (2026); Lin et al. (2026b). A practical streaming VideoLLM must therefore both keep pace with incoming video and answer queries with low latency, making *video-ingestion throughput* and *query-time time-to-first-token* (TTFT) the two primary system objectives.

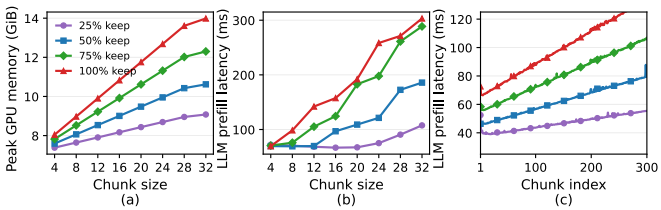


Figure 1: Memory and latency scaling under full-history streaming attention. Token dropping reduces (a) peak GPU memory and (b) LLM prefill latency under simulated 720p streaming input, but (c) prefill latency still grows with the streaming chunk index when the visible visual history accumulates. This suggests that token dropping alone cannot ensure long-duration latency stability without bounding the attended visual context.

Existing streaming VideoLLMs generally adopt one of two paradigms (see Appendix B). The first is *delayed streaming* Dorovatas et al. (2026); Yao et al. (2025), where incoming frames are accumulated but remain largely unprocessed until a user query arrives. At query time, the model performs a single large prefill over the entire observed history. While simple, this design causes query latency and peak GPU memory consumption to grow with stream length Wang et al. (2025b); Zhang et al. (2026a). As a result, delayed streaming struggles to support real-time interaction over long videos.

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.

The second paradigm is *continuous streaming*, which incrementally processes video as frames arrive and continuously updates the model state. Recent work improves individual stages through sparse visual encoding Renggli et al. (2022); Li et al. (2025a); Chen et al. (2024), token pruning and keyframe selection Kim et al. (2024); Wu et al. (2026b), and memory-bank or KV-cache compression Gurukar & Kadav (2025); Di et al. (2025); Yang et al. (2025). However, continuous streaming is fundamentally an end-to-end throughput problem: the system must process video faster than it arrives. Accelerating one stage does not necessarily improve overall throughput, as bottlenecks may simply shift elsewhere. For example, Figure 1 shows that token dropping reduces LLM computation but cannot prevent latency from growing with stream length, eventually limiting sustainable ingestion speed. This motivates our central idea: *treating streaming VideoLLM inference as a coordinated multi-stage pipeline*, where bottlenecks can be identified and bounded to jointly optimize video-ingestion throughput and query-time TTFT under sustained long-video streaming workloads.

To this end, we propose **ViCoStream** (**Vi**deo **Co**ordinated **Stream**ing), a stage-wise coordinated streaming framework that organizes visual preprocessing, visual encoding, token dropping, and LLM inference into a pipelined architecture where different video chunks progress through different thread-based stages concurrently. At the visual front end, incoming frames are accumulated into chunks and processed through overlapped CUDA-stream preprocessing and chunk-wise ViT encoding, improving GPU utilization and exploiting batch parallelism over frame-by-frame execution Li et al. (2025a); Chen et al. (2024). The projector then performs chunk-level temporal token dropping to regulate visual token density, reducing the computation and memory demands of downstream LLM processing Huang et al. (2025); Chen et al. (2025); Wu et al. (2026a). Finally, the LLM employs bounded visual attention together with query-side retrieval, restricting the amount of visible visual history while preserving access to relevant past observations Zhang et al. (2024a; 2026a; 2024b). As a result, the per-chunk prefill cost remains bounded and no longer grows with stream length, enabling arbitrarily long video streams under fixed latency and memory budgets. Collectively, these coordinated controls transform continuous streaming into a slowest-stage scheduling problem and decouple query-time TTFT from streaming FPS across the entire inference pipeline. Experiments show that ViCoStream achieves 134 FPS under 30% token retention while keeping most moderate-compression settings on par with full-history baselines.

Our main contributions are as follows:

1. We formulate continuous VideoLLM inference as a stage-wise streaming pipeline and identify two primary system objectives: sustaining video-ingestion throughput and minimizing query-time TTFT. This perspective exposes how bottlenecks migrate across stages and provides a unified framework for analyzing streaming efficiency.
2. We propose **ViCoStream**, a pipelined architecture that jointly coordinates visual processing and LLM inference. By combining stage-level parallelism with bounded visual-context management, our design maintains fixed per-chunk latency and memory costs independent of total stream length.
3. We provide a systematic efficiency–accuracy study of streaming VideoLLMs, analyzing how chunk granularity, token retention, visual-context size, and retrieval scope affect bottleneck migration and overall system performance. Our approach achieves up to 134 FPS while maintaining competitive accuracy.

2 Related Work

Streaming Video Understanding. Streaming video understanding requires models to process continuously arriving frames and remain responsive to user queries. Many long-video VideoLLMs improve efficiency by sampling frames Li et al. (2024), compressing visual tokens Bolya et al. (2022); Shang et al. (2025); Xing et al. (2024); Shen et al. (2024); Wang et al. (2025a); Wu et al. (2026b;a), or retrieving query-relevant visual context Zhang et al. (2024b); Huang et al. (2025) from a complete video context, but such designs often assume that the full clip or user query is available before compression and are therefore closer to offline processing than strict real-time streaming.

Recent streaming systems process visual inputs incrementally and optimize different stages of the pipeline. For visual preprocessing and visual encoding, TimeChat-Online Yao et al. (2025), VideoScan Li et al. (2025a), VideoLLM-MoD Wu et al. (2024), and CodecSight Zou et al. (2026) reduce incoming visual redundancy, route vision computation, or exploit codec signals before dense model inference. For token dropping, StreamingTOM Chen et al. (2025) reduces redundant temporal

observations during online processing, STC Wang et al. (2025b) maintains a compact streaming token representation to limit LLM input growth, and KFFocus Nie et al. (2025) highlights keyframes so that more computation is allocated to informative visual moments. For LLM prefilling and decoding, StreamMem Yang et al. (2025) summarizes long visual histories into memory states, Flash-VStream Zhang et al. (2024a) uses compact memory to support efficient video streaming, and HERMES Zhang et al. (2026a) reduces the burden of long-context KV caching. Other online assistants focus on when to invoke or respond with the LLM: StreamMind Ding et al. (2025) uses event-gated cognition to avoid unnecessary reasoning, Dispider Qian et al. (2025) separates perception, decision, and reaction for timely interaction, Think-as-You-See Zhang et al. (2026b) explores streaming chain-of-thought reasoning, and LION-FS Li et al. (2025b) combines fast and slow response paths for online video assistance. These component-level designs make online dialogue more practical, but they mainly address local stages of the streaming pipeline in isolation. In contrast, our work analyzes streaming video inference from a multi-stage coordinated streaming perspective, studying how stage-wise controls jointly affect efficiency bottlenecks.

3 Parallel Streaming Pipeline

We formalize streaming VideoLLM inference as a stage-wise parallel pipeline. This formulation specifies how video data enters the system, how it is processed across stages, and how overall streaming throughput is determined under a shared resource model.

Chunk-wise Streaming Input. A video stream is represented as a sequence of frames $V = \{f_1, f_2, \ldots, f_T\}$, which arrive sequentially over time. Since future frames are unavailable in streaming settings, the model must continuously process observed frames while maintaining a state for future query answering under bounded latency constraints.

To enable efficient processing, we partition the stream into non-overlapping chunks of size c. The t-th chunk is defined as $\mathbf{V}_t = \{f_{(t-1)c+1}, \ldots, f_{tc}\}$. Setting $c = 1$ corresponds to frame-by-frame processing. In practice, chunking is beneficial because ViT encoders achieve higher efficiency through batched processing compared to serial per-frame execution.

Pipeline Components. For each incoming chunk, streaming VideoLLM inference consists of four sequential stages. *Visual preprocessing* decodes raw frames and performs resizing, normalization, and sampling. *Visual encoding* maps the preprocessed frames into visual features using a ViT encoder: $\mathbf{X}_t^{(v)} = E_v(\mathbf{V}_t)$. *Token dropping* projects visual features into the language embedding space and removes redundant tokens: $\mathbf{Z}_t = P(\mathbf{X}_t^{(v)})$, where $\mathbf{Z}_t$ denotes the retained visual tokens. *LLM inference* incorporates visual tokens into the language model context, performs prefilling for user queries over the accumulated visual history, and generates autoregressive outputs.

Parallel Streaming Throughput. While a single chunk traverses these stages sequentially, a streaming system can execute different chunks concurrently across stages. We adopt a stage-wise resource model in which each stage processes at most one chunk at a time. Concurrency arises from pipelining: different stages operate on different chunks simultaneously after pipeline warm-up.

Under this model, at time t, different chunks may reside in different stages (e.g., visual preprocessing, ViT encoding, token dropping and LLM inference). Let the per-chunk processing time be denoted by T_{vp}, T_{vit}, T_{drop}, and T_{llm}, respectively. Note that T_{llm} may vary with the amount of visible visual context. In steady state, the pipeline throughput is governed by the slowest stage:

$$T_{\mathrm{pipe}} = \max\{T_{\mathrm{vp}}, T_{\mathrm{vit}}, T_{\mathrm{drop}}, T_{\mathrm{llm}}\}.$$

The corresponding maximum streaming frame rate is $\mathrm{FPS}_{\max} = 1000c/T_{\mathrm{pipe}}$, where time is measured in milliseconds.

Streaming Latency Metrics. Streaming performance has two complementary aspects. The first is continuous video ingestion throughput, captured by $\mathrm{FPS}_{\max}$, which measures how quickly the system can process incoming frames. The second is query-time latency, measured by time-to-first-token (TTFT), which includes LLM prefilling and the generation of the first output token after a complete user query is submitted over the observed video context. TTFT is measured from the moment when the complete user query is submitted to the LLM, using the video state observed up to the query timestamp. Frames arriving afterward may continue through the background pipeline, but are not used as evidence for the current response. In this work, we use $\mathrm{FPS}_{\max}$ to characterize streaming throughput, while evaluating TTFT separately as the user-facing response latency.

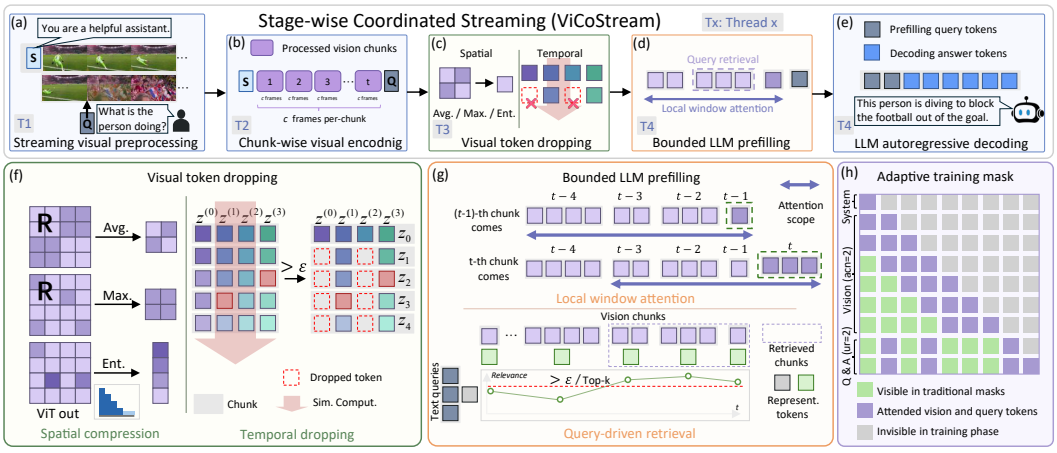


Figure 2: Overview of stage-wise coordinated streaming (ViCoStream). (a)–(e) show the main parallel pipeline from streaming input to autoregressive decoding in different threads. (d) and (e) share a thread due to their dependencies. (f) is spatial and temporal visual token compression, (g) is vision-based locality and query-driven retrieval for cross-attention scope, and (h) shows the adaptive training mask that matches these vision and query attention.

4 Stage-Wise Coordinated Streaming

The pipeline formulation in Section 3 suggests that bounded real-time streaming requires controlling each stage rather than only reducing a single cost term. We therefore organize coordinated streaming around four inference stages: visual preprocessing, visual encoding, token dropping, and LLM prefilling/decoding. In addition, we use an adaptive attention mask to align training-time visibility with the bounded vision-side attention and query-side retrieval used at inference. For each inference stage, we identify the corresponding bottleneck and choose a lightweight control that keeps the continuous visual-ingestion pipeline bounded, as summarized in Figure 2.

Visual Preprocessing. Visual preprocessing samples, resizes, and normalizes incoming frames. Although this stage is not part of the neural model, it can become the throughput ceiling once LLM-side cost is reduced. We therefore treat visual preprocessing as the first stage of the streaming pipeline. When preprocessing involves GPU-side operations such as tensor resizing, normalization, patchification, or host-to-device transfer, we assign them to an independent CUDA stream so that this stage can overlap with later model stages rather than being serialized on the default stream. In steady state, visual preprocessing can prepare chunk $t{+}3$ while visual encoding, token dropping, and LLM prefilling process earlier chunks. This does not remove preprocessing cost, but it prevents preprocessing latency from being unnecessarily serialized with the later stages.

Visual Encoding. Visual encoding determines how quickly new visual tokens are produced. Frame-level encoding invokes the visual encoder frequently and introduces high scheduling overhead, while overly large chunks delay token availability and increase per-chunk latency. We therefore use chunk-wise visual encoding, where consecutive frames are grouped into chunks and encoded together. The latency breakdown in Table 2 shows why this granularity matters: increasing c changes both VP and ViT latency and shifts the maximum streaming FPS, rather than simply scaling all stages uniformly.

Token Dropping. The projector and token-dropping stage determines how many visual tokens are passed to the LLM. Dense visual token sequences are highly redundant in streaming videos Yao et al. (2025). We observe the same pattern under full attention: as shown in Figure 3, dropping 78.1% of visual tokens preserves the overall accuracy relative to the full-token baseline, while accuracy only drops substantially under more aggressive compression. This motivates token dropping as a direct way to reduce LLM prefilling cost before attention is applied.

We consider both spatial and temporal token dropping, as illustrated in Figure 2(f). Spatial compression operates at the frame level. Given projected visual tokens $X_v^t \in \mathbb{R}^{n_f \times d}$ for the t-th frame, where n_f denotes the number of tokens per frame, we obtain a compact representation $\bar{X}_v^t \in \mathbb{R}^{\bar{n}_f \times d}$ with $\bar{n}_f < n_f$. This operation is applied independently to each frame, regardless of whether frames are processed individually or grouped into chunks. We evaluate three spatial strategies. *Average*

Context	Drop	All
Full tokens	0.0%	70.76
Nearest-8K	49.6%	71.68
Nearest-4K	64.5%	**72.96**
Nearest-2K	73.4%	71.76
Nearest-1K	82.2%	71.40

Table 1: Nearest visual context results with Qwen2.5-VL-3B under full attention.

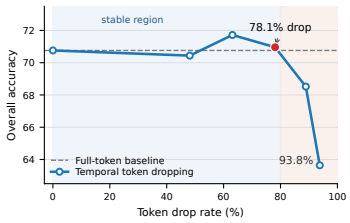


Figure 3: Token drop rate versus accuracy with trained Qwen2.5-VL-3B under full attention.

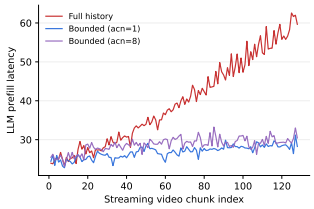


Figure 4: Full-history latency grows, while bounded attention remains stable.

pooling aggregates tokens in a non-overlapping 2×2 region $\mathcal{R}_i$ as $\bar{x}_i = \frac{1}{|\mathcal{R}_i|} \sum_{x_j \in \mathcal{R}_i} x_j$, preserving region-level semantics at reduced spatial resolution. *Max pooling* retains the strongest activation in each region, $\bar{x}_i = \max_{x_j \in \mathcal{R}_i} x_j$, and discards less salient tokens. *Entropy-based selection* measures token importance by feature entropy $\mathcal{I}(x_j) = -\sum_m p_{jm} \log p_{jm}$, where p_{jm} denotes the normalized probability of the m-th feature dimension, and retains top-K tokens with the highest entropy scores.

Temporal compression is applied at the chunk level and we always preserve the first frame of each chunk as an anchor, so every time segment leaves a residual visual record even when most pixels remain unchanged. For each spatial position p, we compute pairwise cosine similarity between the corresponding projected tokens in every frame pair $i, j \in \{1, \ldots, c\}$ with $i < j$:

$$\mathcal{S}_{t,i\to j}(p) = \frac{\langle \mathbf{z}_t^{(i)}(p), \mathbf{z}_t^{(j)}(p) \rangle}{\|\mathbf{z}_t^{(i)}(p)\| \, \|\mathbf{z}_t^{(j)}(p)\|}. \tag{1}$$

Here, $\mathbf{z}_t^{(i)}(p) \in \mathbb{R}^d$ denotes the projected feature of the token at spatial position p in the i-th frame of the t-th chunk. A token at position p is dropped when all pairwise similarities exceed a threshold τ:

$$\mathrm{drop}_t(p) \iff \bigwedge_{i<j} \left[\mathcal{S}_{t,i\to j}(p) > \tau\right]. \tag{2}$$

This rule removes only tokens that are consistently redundant across the whole chunk. The preserved first-frame anchor provides a stable reference for what occurred in the segment and reduces the risk of error accumulation under aggressive temporal pruning.

When c=1, this formulation degenerates into frame-level temporal dropping, where each frame is treated as an independent dropping unit. In this setting, we prune tokens according to the projector-level similarity between the current frame and the previous frame, using the same threshold-based rule but without a multi-frame chunk anchor.

LLM Prefilling and Decoding. The LLM stage consumes retained visual tokens through prefilling and later responds to user queries. Full-history attention makes this stage grow with accumulated visual context, so the key bottleneck is the historical visual cache visible to each incoming chunk.

Our empirical observations support a locality assumption for this stage: under full attention, retaining the nearest 4K visual tokens improves the overall accuracy from 70.76 to 72.96 while dropping 64.5% of tokens, as shown in Table 1. This suggests that each incoming vision chunk often relies on nearby visual context rather than the entire accumulated history. As illustrated in Figure 2(g), we therefore adopt two complementary attention scopes: vision-side local prefilling and query-driven retrieval. For vision-side prefilling, we restrict each incoming chunk to attend to itself and the preceding *acn* chunks. Let $\mathbf{Q}_v^{(t)}$, $\mathbf{K}_v^{(t)}$, and $\mathbf{V}_v^{(t)}$ denote the query, key, and value projections of the current visual chunk. The attended keys and values are $\hat{\mathbf{K}}_v^{(t)} = \{\mathbf{K}_v^{(t-acn)}, \ldots, \mathbf{K}_v^{(t)}\}$ and $\hat{\mathbf{V}}_v^{(t)} = \{\mathbf{V}_v^{(t-acn)}, \ldots, \mathbf{V}_v^{(t)}\}$. The bounded visual attention is then computed as $\mathrm{Attn}_v^{(t)} = \mathrm{softmax}(\mathbf{Q}_v^{(t)}(\hat{\mathbf{K}}_v^{(t)})^\top/\sqrt{d})\hat{\mathbf{V}}_v^{(t)}$. This bounds the visible cache size and prevents LLM prefilling from growing with stream length.

For query-time decoding, the user query can introduce a different access pattern because relevant evidence may appear at non-recent positions in the stream. To bound query-time visual access without forcing full-history attention, we retrieve a limited set of visual chunks according to query–chunk similarity. The query tokens $\mathbf{X}_q$ and each visual chunk $\mathbf{X}_v^{(i)}$ are averaged into representative features $\bar{\mathbf{x}}_q$ and $\bar{\mathbf{x}}_v^{(i)}$, and their cosine similarity is $s_i = \langle \bar{\mathbf{x}}_q, \bar{\mathbf{x}}_v^{(i)} \rangle / \|\bar{\mathbf{x}}_q\| \, \|\bar{\mathbf{x}}_v^{(i)}\|$.

Only the top-k chunks are exposed to query tokens during decoding. Let $\tilde{\mathbf{K}}_v^{(t)}$ and $\tilde{\mathbf{V}}_v^{(t)}$ denote the key and value caches constructed from these retrieved chunks. The query-side attention is computed as $\mathrm{Attn}_q^{(t)} = \mathrm{softmax}(\mathbf{Q}_q(\tilde{\mathbf{K}}_v^{(t)})^\top/\sqrt{d})\tilde{\mathbf{V}}_v^{(t)}$. Together, token dropping, bounded visual prefilling, and query-side retrieval convert the LLM cost from a history-dependent term into a bounded per-chunk and bounded per-query cost.

To align training with these two attention scopes, we add a visual-scope mask $M \in \{0, -\infty\}^{T\times T}$ on top of the causal mask, as illustrated in Figure 2(h). Let $\mathcal{I}_{\mathrm{vis}}$ and $\mathcal{I}_{\mathrm{qry}}$ index vision and query tokens, and $c(i)$ denote the chunk index of vision token i. Local prefilling keeps the current chunk and the preceding *acn* chunks visible, $\mathcal{V}_{\mathrm{vis}}(c(i)) = \{c(i) - acn, \ldots, c(i)\}$, producing a staircase-shaped local band in the vision–vision region. For a query token, visibility is determined by query-driven retrieval: only chunks whose similarity score s_j exceeds a threshold τ_q remain visible, giving $\mathcal{V}_{\mathrm{qry}} = \{j \mid s_j \geq \tau_q\}$. Thus the query–vision region can be sparse and non-contiguous, allowing query tokens to access semantically relevant chunks regardless of temporal distance. Formally, the visual-scope mask is defined as

$$M_{i,j} = \begin{cases} 0, & j \notin \mathcal{I}_{\mathrm{vis}}, \\ 0, & i, j \in \mathcal{I}_{\mathrm{vis}},\ c(j) \in \mathcal{V}_{\mathrm{vis}}(c(i)), \\ 0, & i \in \mathcal{I}_{\mathrm{qry}},\ j \in \mathcal{I}_{\mathrm{vis}},\ c(j) \in \mathcal{V}_{\mathrm{qry}}, \\ -\infty, & \text{otherwise.} \end{cases} \tag{3}$$

where 0 leaves an attention logit unchanged and $-\infty$ masks it before softmax. Because M is added to the causal mask, entries that do not involve visual keys are unaffected. This unified mask enforces local visual prefilling while preserving query-specific access to retrieved visual evidence.

5 Experiments

5.1 Experimental Setup

Datasets. We train our models on two video-language instruction datasets: LLaVA-Video-100K Zhang et al. (2024c) and TimeChat-Online-139K Yao et al. (2025). Unless otherwise specified, models are trained on their union. More dataset preprocessing details are provided in Appendix A.1.

We evaluate models on six streaming-oriented benchmarks: StreamingBench Lin et al. (2026a), OVO-Bench Li et al. (2025c), StreamBench Xiong et al. (2025), VStream-Ego Zhang et al. (2024a), VStream-Movie Zhang et al. (2024a), and ETBench Qian et al. (2025). These benchmarks cover real-time visual understanding, online video question answering, long-duration stream comprehension, and event-time reasoning. For each benchmark, we follow its official evaluation protocol and report the corresponding accuracy or task-level score; ETBench is summarized separately because it mixes heterogeneous metrics across sub-tasks. For StreamBench and VStream, we use GPT-3.5-Turbo as the judge for the score metric following the setup of Flash-VStream Zhang et al. (2024a). In compact tables, we use SBench to denote StreamingBench and RVS to denote VStream.

Implementation Details. Experiments are mainly conducted on Qwen2.5-VL-3B-Instruct and Qwen2.5-VL-7B-Instruct. Unless otherwise specified, our ablation studies use 3B-level models trained on a 50% random subset of each dataset. During fine-tuning, the ViT encoder is frozen while the multimodal aligner and language model are updated. Models are trained on up to four NVIDIA H100 GPUs, while all evaluation and latency measurements are performed on a single NVIDIA A100-SXM4-80GB GPU. The ViCoStream implementation details are summarized in Appendix A.3.

Table 2: Per-module latency (ms) breakdown and maximum streaming FPS (*acn*=4).

c	VP	ViT	Drop	Prefill	FPS
Token Retention 30% ($r\approx70\%$)					
1	5	30	6	38	26
4	37	40	12	44	91
8	73	64	24	44	110
16	113	119	44	42	**134**
32	248	235	87	69	129
Token Retention 50% ($r\approx50\%$)					
1	5	30	6	38	26
4	47	41	12	45	86
8	82	64	23	45	98
16	160	119	45	64	100
32	317	236	87	124	**101**

5.2 Efficiency and Accuracy Analysis

We first evaluate the end-to-end behavior of ViCoStream after jointly applying the stage-wise controls introduced above. This Section focuses on the resulting stage-wise latency, bottleneck migration, long-duration stability, and accuracy retention under real-time streaming constraints. We defer component-level analyses of visual preprocessing, visual encoding, token dropping, intra-chunk attention scope, attention locality, and query retrieval to the Section 5.3.

Table 3: Accuracy under ViCoStream. **Baseline** uses full cross-attention over all accumulated visual tokens without compression. † denotes ViCoStream with temporal token compression and bounded cross-attention. Token drop rates in parentheses are measured on StreamingBench, and subscripts indicate gains, <3% drops, or >3% drops relative to baseline. Values are percentages.

Model	acn	ur	s_{drop}	StreamingBench	OVO-Bench	StreamBench	RVS-E	RVS-M
Traditional Streaming Prefill (Baseline)								
Qwen2.5-VL-3B	∞	∞	–	70.32	47.13	48.37	34.59	33.91
Qwen2.5-VL-7B	∞	∞	–	74.60	49.35	54.90	34.86	34.60
ViCoStream†								
Qwen2.5-VL-3B†	4	64	$0.25_{\downarrow 57.6\%}$	$70.44_{\uparrow 0.1}$	$45.30_{\downarrow 1.8}$	$49.72_{\uparrow 1.4}$	$35.22_{\uparrow 0.6}$	$31.97_{\downarrow 1.9}$
	8	64	$0.25_{\downarrow 57.3\%}$	$70.52_{\uparrow 0.2}$	$46.35_{\downarrow 0.8}$	$49.06_{\uparrow 0.7}$	$34.12_{\downarrow 0.5}$	$31.55_{\downarrow 2.4}$
	4	64	$0.10_{\downarrow 72.9\%}$	$68.32_{\downarrow 2.0}$	$42.19_{\downarrow 4.9}$	$46.35_{\downarrow 2.0}$	$34.40_{\downarrow 0.2}$	$29.04_{\downarrow 4.9}$
	8	64	$0.10_{\downarrow 73.0\%}$	$67.39_{\downarrow 2.9}$	$43.28_{\downarrow 3.9}$	$47.52_{\downarrow 0.9}$	$35.43_{\uparrow 0.8}$	$32.25_{\downarrow 1.7}$
	4	32	$0.25_{\downarrow 57.6\%}$	$70.68_{\uparrow 0.4}$	$45.16_{\downarrow 2.0}$	$49.94_{\uparrow 1.6}$	$32.29_{\downarrow 2.3}$	$31.98_{\downarrow 1.9}$
	8	32	$0.25_{\downarrow 57.3\%}$	$70.48_{\uparrow 0.2}$	$46.17_{\downarrow 1.0}$	$49.14_{\uparrow 0.8}$	$32.18_{\downarrow 2.4}$	$30.93_{\downarrow 3.0}$
	4	32	$0.10_{\downarrow 72.9\%}$	$68.28_{\downarrow 2.0}$	$42.48_{\downarrow 4.7}$	$46.09_{\downarrow 2.3}$	$30.99_{\downarrow 3.6}$	$28.39_{\downarrow 5.5}$
	8	32	$0.10_{\downarrow 72.9\%}$	$67.68_{\downarrow 2.6}$	$43.85_{\downarrow 3.3}$	$47.66_{\downarrow 0.7}$	$32.70_{\downarrow 1.9}$	$31.34_{\downarrow 2.6}$
Qwen2.5-VL-7B†	4	64	$0.25_{\downarrow 48.0\%}$	$71.64_{\downarrow 3.0}$	$49.41_{\uparrow 0.1}$	$54.07_{\downarrow 0.8}$	$36.68_{\uparrow 1.8}$	$31.67_{\downarrow 2.9}$
	8	64	$0.25_{\downarrow 48.4\%}$	$71.80_{\downarrow 2.8}$	$50.77_{\uparrow 1.4}$	$53.68_{\downarrow 1.2}$	$34.31_{\downarrow 0.6}$	$33.23_{\downarrow 1.4}$
	4	64	$0.10_{\downarrow 70.6\%}$	$69.71_{\downarrow 4.9}$	$47.56_{\downarrow 1.8}$	$51.01_{\downarrow 3.9}$	$32.70_{\downarrow 2.2}$	$28.14_{\downarrow 6.5}$
	8	64	$0.10_{\downarrow 70.5\%}$	$70.42_{\downarrow 4.2}$	$47.23_{\downarrow 2.1}$	$50.97_{\downarrow 3.9}$	$35.63_{\uparrow 0.8}$	$31.07_{\downarrow 3.5}$
	4	32	$0.25_{\downarrow 48.0\%}$	$71.40_{\downarrow 3.2}$	$49.71_{\uparrow 0.4}$	$53.51_{\downarrow 1.4}$	$36.71_{\uparrow 1.9}$	$31.70_{\downarrow 2.9}$
	8	32	$0.25_{\downarrow 48.4\%}$	$71.48_{\downarrow 3.1}$	$50.35_{\uparrow 1.0}$	$53.51_{\downarrow 1.4}$	$34.40_{\downarrow 0.5}$	$33.02_{\downarrow 1.6}$
	4	32	$0.10_{\downarrow 70.6\%}$	$69.56_{\downarrow 5.0}$	$46.77_{\downarrow 2.6}$	$50.53_{\downarrow 4.4}$	$32.12_{\downarrow 2.7}$	$28.30_{\downarrow 6.3}$
	8	32	$0.10_{\downarrow 70.5\%}$	$70.44_{\downarrow 4.2}$	$47.32_{\downarrow 2.0}$	$50.42_{\downarrow 4.5}$	$34.88_{\uparrow 0.0}$	$31.76_{\downarrow 2.8}$

Table 4: ETBench task-level results. ECA, GVQ, RAR, and RVQ are accuracy-based tasks; EPM is temporal grounding with mIoU/F1; TAL-PT, TAL-TH, and TVG are F1-based temporal localization tasks; TEM reports mean recall. † denotes ViCoStream with temporal token compression and local cross-attention. Values are percentages.

Model	acn	ur	s	ECA ↑	EPM (mIoU↑ / F1↑)	GVQ ↑	RAR ↑	RVQ ↑	TAL-PT ↑	TAL-TH ↑	TEM ↑	TVG ↑
Qwen2.5-VL-3B	∞	∞	–	37.80	1.29 / 1.60	53.45	34.80	51.60	18.63	13.09	16.80	27.30
Qwen2.5-VL-7B	∞	∞	–	45.00	1.66 / 2.10	56.55	36.60	54.60	20.17	12.61	26.70	33.35
Qwen2.5-VL-3B†	4	32	0.25	$37.20_{\downarrow 0.6}$	$1.54_{\uparrow 0.2}$ / $1.75_{\uparrow 0.1}$	$52.07_{\downarrow 1.4}$	$35.80_{\uparrow 1.0}$	$48.20_{\downarrow 3.4}$	$21.74_{\uparrow 3.1}$	$14.66_{\uparrow 1.6}$	$20.40_{\uparrow 3.6}$	$32.10_{\uparrow 4.8}$
	4	32	0.1	$37.20_{\downarrow 0.6}$	$1.62_{\uparrow 0.3}$ / $1.95_{\uparrow 0.3}$	$54.48_{\uparrow 1.0}$	$35.20_{\uparrow 0.4}$	$45.60_{\downarrow 6.0}$	$21.63_{\uparrow 3.0}$	$14.42_{\uparrow 1.3}$	$16.35_{\downarrow 0.4}$	$30.35_{\uparrow 3.1}$
Qwen2.5-VL-7B†	4	32	0.25	$37.60_{\downarrow 7.4}$	$2.04_{\uparrow 0.4}$ / $2.50_{\uparrow 0.4}$	$54.48_{\downarrow 2.1}$	$35.60_{\downarrow 1.0}$	$51.20_{\downarrow 3.4}$	$17.95_{\downarrow 2.2}$	$12.51_{\downarrow 0.1}$	$21.80_{\downarrow 4.9}$	$28.30_{\downarrow 5.1}$
	4	32	0.1	$37.00_{\downarrow 8.0}$	$1.25_{\downarrow 0.4}$ / $1.50_{\downarrow 0.6}$	$53.10_{\downarrow 3.4}$	$33.60_{\downarrow 3.0}$	$46.60_{\downarrow 8.0}$	$12.22_{\downarrow 8.0}$	$12.66_{\uparrow 0.1}$	$21.40_{\downarrow 5.3}$	$26.40_{\downarrow 7.0}$

Stage-Wise Latency and Maximum FPS. Table 2 reports stage-wise latency on a single NVIDIA A100-SXM4-80GB GPU with simulated 720p input resized to 448×448. At c=1, frequent ViT and model invocations limit throughput to 26 FPS. Chunk-wise processing amortizes this overhead, reaching 134 FPS at 30% token retention and 101 FPS at 50% retention. Different stages can process different chunks concurrently, so throughput is governed by the slowest continuous stage rather than their serial latency sum, i.e., $\mathrm{FPS}_{\max} = 1000c/T_{\mathrm{pipe}}$. Larger chunks are not always better because they also increase VP, ViT, and token-dropping latency. Bounded attention keeps LLM prefilling independent of stream length; once this LLM-side cost is reduced, the bottleneck migrates toward visual encoding or preprocessing as the next throughput ceiling.

Long-Duration Latency Stability. We next examine whether LLM prefilling remains bounded as visual history accumulates. As shown in Figure 4, full-history prefilling makes each incoming chunk attend to an increasingly long visual cache, so LLM prefill latency rises steadily with the streaming chunk index. By contrast, bounded attention with acn=1 or acn=8 keeps the visible visual context fixed and yields nearly flat per-chunk latency. This confirms that attention locality converts LLM prefilling from a history-dependent cost into a bounded per-chunk cost, which is necessary for long-duration streaming input under a fixed latency budget.

Query-Side Response Latency. Figure 5 (Left) further shows that TTFT is governed by the product of retrieval scope and token retention. When only 25% or 50% of tokens are kept, increasing the query-visible chunks has a modest latency impact. However, at 75% or 100% retention, TTFT rises sharply once the query attends to 16 or more chunks. Thus, query-side retrieval should be tuned jointly with token dropping: exposing more chunks is only affordable when each chunk has already been sufficiently compressed. Therefore, the decoding stage will not become a new bottleneck under sufficiently aggressive token retention. Figure 5 (Middle) further compares full-history query access

with bounded query selection across 25% token retention. With *ur*=32 or *ur*=64, query-side prefill latency is reduced to below 50 ms relative to full-history access, indicating that retrieval keeps the query-prefill cost nearly constant even as the accumulated stream grows.

In our measured configurations, even a conservative serial sum of visual preprocessing, visual encoding, token dropping, and vision-side prefilling for the last observed chunk is at most 764 ms in Table 2, below typical spoken-query duration. Thus, the reported TTFT isolates query-side prefilling and first-token generation, while chunk-processing cost is captured by the stage-wise latency analysis.

Accuracy Preservation Under ViCoStream After analyzing stage-wise latency, we evaluate the full ViCoStream configuration by jointly applying token dropping, bounded visual attention, and query-side retrieval. The goal is not to claim a standalone accuracy improvement, but to measure how much accuracy is retained once the pipeline is constrained for real-time streaming. For each base model, we compare against traditional streaming prefill with full accumulated visual context and no compression ($r = 0\%$, $acn = \infty$, $ur = \infty$), where *ur* denotes the number of visual chunks attended by the user query. Additional comparisons against representative streaming models under matched backbone and experimental settings are reported in Table 15.

Table 3 shows that moderate token dropping provides a favorable trade-off: with s=0.25, most results remain within 3 points of the full-history baseline for both 3B and 7B models, and several settings improve on StreamingBench, StreamBench, or RVS-E. This suggests that full visual history is often unnecessary once recent visual attention and query-side retrieval are preserved. More aggressive dropping (s=0.1) exposes the failure modes: StreamingBench and StreamBench remain relatively robust, whereas OVO-Bench and RVS-M degrade more often, indicating higher sensitivity to fine temporal evidence or movie-level context.

Table 5: Component analyses of ViCoStream: (a) full vs. bounded attention, (b) VP latency, (c) frame- vs. chunk-level dropping on StreamingBench.

(a) Full vs. Bounded Attention

s	Full Attn.	SBench	OVO-Bench		
			RT	BT	FAR
Qwen2.5-VL-3B					
0.25	✓	69.96	54.53	39.33	34.96
	✗	70.44	55.74	37.36	32.35
0.10	✓	67.40	52.13	40.72	34.80
	✗	68.32	50.45	34.78	33.07
Qwen2.5-VL-7B					
0.25	✓	73.28	59.71	44.14	39.14
	✗	71.64	60.11	42.44	35.01
0.10	✓	71.36	55.51	43.54	37.65
	✗	69.71	57.70	42.15	32.69

(b) VP Latency

c	Trad. VP	CUDA VP	Speedup
1	52.96	4.48	11.8×
4	227.59	29.38	7.7×
8	461.40	59.49	7.8×
16	989.48	122.22	8.1×
32	1961.17	244.25	8.0×

(c) Frame- vs. Chunk-Level Dropping

Input Setting	c	acn	s	All
Baseline	–	∞	–	70.32
Frame-level	1	16	0.25	68.80
	1	16	0.10	66.48
Chunk-level	4	4	0.25	70.68
	4	4	0.10	68.28
	4	8	0.25	70.48
	4	8	0.10	67.68

Table 4 further reveals task-dependent effects under *ur*=32. For the 3B model, several temporal-localization metrics, including TAL-PT, TAL-TH, and TVG, improve despite bounded context, suggesting that local attention can remove redundant history while retaining event evidence. The 7B model is less stable, with larger drops on ECA, TEM, and TVG, indicating greater sensitivity to compressed visual history on some event-time tasks.

The comparisons above establish the accuracy competitiveness of stage-wise coordinated streaming, but they do not factor out token dropping as a controlled variable: the baseline uses full visual context without compression, while the coordinated variants combine dropping with bounded attention and query-side retrieval. To isolate the effect of attention scope from the effect of token reduction, we compare full-attention and bounded-attention intra-chunk prefill under the same chunk-level token-dropping configuration. As shown in Table 5(a), bounded attention can slightly help StreamingBench and real-time perception for the 3B model, but it can weaken memory-heavy OVO-Bench splits such as backward tracing under stronger dropping. Overall, neither attention scope is uniformly better across models and tasks. The accuracy trade-off is therefore dominated by the token-dropping and retrieval controls, while the intra-chunk attention scope mainly changes how temporal evidence is exposed to the LLM during coordinated parallel execution.

5.3 Ablation Studies

The preceding analysis examines each pipeline stage individually to understand per-component bottlenecks and the resulting accuracy–latency trade-offs. The front-end ablations analyze the VP and

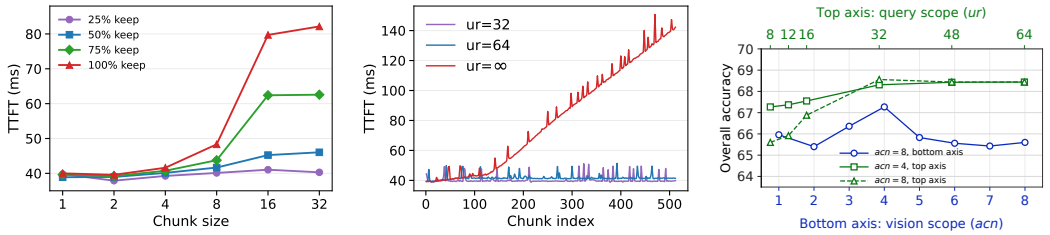


Figure 5: Latency and accuracy under bounded cross-attention. (Left) Query response latency (TTFT) under different token retention and retrieval budgets. (Middle) Query-side prefill TTFT under full-history access and bounded query-selection budgets. (Right) Accuracy under different vision-side and query-side attention scopes.

ViT stages that limit throughput once LLM-side cost is reduced, while the token dropping, attention locality, and retrieval ablations characterize the controls that govern the efficiency–accuracy balance.

Visual Preprocessing. Visual preprocessing mainly determines whether incoming frames can be prepared fast enough for the downstream model stages. Table 5(b) shows that CUDA-stream VP substantially reduces preprocessing latency compared with traditional CPU-based VP, with roughly $8\times$ speedup for larger chunks. Locally, this removes the CPU preprocessing bottleneck for each chunk, but globally the optimized VP cost still grows with chunk size, so preprocessing remains a real throughput stage once LLM-side cost is bounded.

Visual Encoding. Visual encoding is governed by the ViT stage, where chunk-wise processing is the main alternative to frame-level encoding. As reflected in Table 2, increasing the chunk size amortizes frequent small ViT invocations, but it also raises the per-chunk ViT latency. This explains why throughput improves from frame-level processing to moderate chunk sizes, while larger chunks can shift the limiting stage from LLM prefilling to visual encoding or preprocessing.

Token Dropping Ablations. Before comparing token-compression strategies, we first examine the granularity of temporal token dropping. As defined in Section 4, setting c=1 degenerates the chunk-level rule into frame-level temporal dropping, where each frame is compressed independently based on projector-level similarity to the previous frame. Table 5(c) shows that frame-level dropping remains competitive at moderate thresholds, but becomes less robust under aggressive dropping. This supports our use of chunk-level token dropping, where chunk anchors provide a more stable unit for preserving temporal evidence. With chunk-level granularity fixed, we further study temporal and spatial token reduction, with detailed settings and per-task results reported in Appendix A.5. Temporal dropping gives a smooth trade-off, with backward tracing and real-time perception more sensitive than forward active response. Spatial compression is more strategy-dependent: pooling consistently hurts accuracy, whereas entropy-based selection preserves salient tokens. The two axes are complementary, so combining them gives the strongest compression at the largest cost.

LLM Attention Locality. Token dropping reduces tokens per chunk but does not prevent visible history from growing with stream length, so bounded attention is needed to control LLM prefilling over long-duration inputs. As shown in Figure 5 (Right), *acn* is not monotonic: increasing the vision-side window to *acn*=4 improves accuracy, but further increasing it to 8 brings no additional gains and can slightly reduce performance. The best region is a moderate local window around *acn*=4, suggesting that recent chunks provide useful evidence while older chunks add redundancy or noise. On the query side, increasing *ur* helps initially but largely saturates around *ur*=32 (about 128 frames when c=4). Thus, query-time access is treated as a compact retrieval budget that preserves evidence while bounding response latency.

6 Conclusion

We study why real-time streaming remains difficult for VideoLLMs even with compression or memory modules, and formulate streaming inference as a stage-wise coordinated pipeline over visual preprocessing, visual encoding, token dropping, and LLM prefilling/decoding, where continuous ingestion is limited by the slowest stage and response latency is reflected by TTFT. This view reveals that reducing LLM-side cost alone is insufficient: once visual tokens are compressed or attention is bounded, the bottleneck can migrate to visual encoding, preprocessing, or query-time retrieval. Our results show that ViCoStream enables high-frame-rate streaming while preserving accuracy close to full-history baselines under bounded latency and memory budgets.

AI USE STATEMENT

Generative AI tools were used only for language polishing, structural editing, and LaTeX formatting. They were not used for research conception, method or experiment design, code implementation, data processing, or result interpretation. The authors reviewed all AI-assisted edits and take full responsibility for the final content of this paper.

ETHICS STATEMENT

This work uses only publicly released datasets, benchmarks, and model checkpoints; we collect no new data and involve no human participants. All artifacts are used under their original access conditions and following their official evaluation protocols, and we neither redistribute nor re-release any dataset or benchmark video. Streaming video understanding can be applied to continuous monitoring of video feeds, which raises privacy and surveillance concerns; our contribution is systems-level methods and analysis for improving streaming inference efficiency rather than a new perception capability, and we neither target nor endorse such applications. Our method modifies inference scheduling and the management of visual context without introducing capabilities beyond those of the underlying VideoLLM; any deployment therefore remains subject to the original model's license, intended-use policy, and associated limitations.

REPRODUCIBILITY STATEMENT

The streaming formulation and stage-wise controls are described in Sections 3 and 4. The experimental setup and latency-measurement protocol are reported in Section 5, with implementation, training, data-processing, and complete evaluation details provided in the appendix. Together, these sections specify the settings needed to reproduce the reported results.

REFERENCES

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022.

Dibyadip Chatterjee, Edoardo Remelli, Yale Song, Bugra Tekin, Abhay Mittal, Bharat Bhatnagar, Necati Cihan CamgÃḳz, Shreyas Hampali, Eric Sauser, Shugao Ma, et al. Memory-efficient streaming videollms for real-time procedural video understanding. *arXiv preprint arXiv:2504.13915*, 2025.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18407–18418, 2024.

Xueyi Chen, Keda Tao, Kele Shao, and Huan Wang. Streamingtom: Streaming token compression for efficient video understanding. *arXiv preprint arXiv:2510.18269*, 2025.

Shangzhe Di, Zhelun Yu, Guanghao Zhang, Haoyuan Li, Hao Cheng, Bolin Li, Wanggui He, Fangxun Shu, and Hao Jiang. Streaming video question-answering with in-context video kv-cache retrieval. In *International Conference on Learning Representations*, volume 2025, pp. 42115–42127, 2025.

Xin Ding, Hao Wu, Yifan Yang, Shiqi Jiang, Qianxi Zhang, Donglin Bai, Zhibo Chen, and Ting Cao. Streammind: Unlocking full frame rate streaming video dialogue through event-gated cognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 13448–13459, 2025.

Evangelos Dorovatas, Soroush Seifi, Gunshi Gupta, and Rahaf Aljundi. Recurrent attention-based token selection for efficient streaming video-llms. *Advances in Neural Information Processing Systems*, 38:144088–144114, 2026.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation

benchmark of multi-modal llms in video analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 24108–24118, 2025.

Saket Gurukar and Asim Kadav. Long-vmnet: Accelerating long-form video understanding via fixed memory. *arXiv preprint arXiv:2503.13707*, 2025.

Xiaohu Huang, Hao Zhou, and Kai Han. Prunevid: Visual token pruning for efficient video large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19959–19973, 2025.

Minchul Kim, Shangqian Gao, Yen-Chang Hsu, Yilin Shen, and Hongxia Jin. Token fusion: Bridging the gap between token pruning and token merging. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1383–1392, 2024.

Ruanjun Li, Yuedong Tan, Yuanming Shi, and Jiawei Shao. Videoscan: Enabling efficient streaming video understanding via frame-level semantic carriers. *arXiv preprint arXiv:2503.09387*, 2025a.

Wei Li, Bing Hu, Rui Shao, Leyang Shen, and Liqiang Nie. Lion-fs: Fast & slow video-language thinker as online video assistant. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3240–3251, 2025b.

Xinhao Li, Yi Wang, Jiashuo Yu, Xiangyu Zeng, Yuhan Zhu, Haian Huang, Jianfei Gao, Kunchang Li, Yinan He, Chenting Wang, et al. Videochat-flash: Hierarchical compression for long-context video modeling. *arXiv preprint arXiv:2501.00574*, 2024.

Yifei Li, Junbo Niu, 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?, 2025. *URL https://arxiv. org/abs/2501.05510*, 2025c.

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, 2026a.

Junyan Lin, Junlong Tong, Hao Wu, Jialiang Zhang, Jinming Liu, Xin Jin, and Xiaoyu Shen. Speak while watching: Unleashing true real-time video understanding capability of multimodal large language models. *arXiv preprint arXiv:2601.06843*, 2026b.

Yingzi Ma, Yulong Cao, Jiachen Sun, Marco Pavone, and Chaowei Xiao. Dolphins: Multimodal language model for driving. In *European Conference on Computer Vision*, pp. 403–420. Springer, 2024.

Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision–language–action models for embodied ai. *IEEE Transactions on Neural Networks and Learning Systems*, 2026.

Yao Mu, Qinglong Zhang, Mengkang Hu, Wenhai Wang, Mingyu Ding, Jun Jin, Bin Wang, Jifeng Dai, Yu Qiao, and Ping Luo. Embodiedgpt: Vision-language pre-training via embodied chain of thought. *Advances in Neural Information Processing Systems*, 36:25081–25094, 2023.

Ming Nie, Chunwei Wang, Hang Xu, and Li Zhang. Kffocus: Highlighting keyframes for enhanced video understanding. *arXiv preprint arXiv:2508.08989*, 2025.

Rui Qian, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Shuangrui Ding, Dahua Lin, and Jiaqi Wang. Streaming long video understanding with large language models. *Advances in Neural Information Processing Systems*, 37:119336–119360, 2024.

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 *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 24045–24055, 2025.

Cedric Renggli, André Susano Pinto, Neil Houlsby, Basil Mustafa, Joan Puigcerver, and Carlos Riquelme. Learning to merge tokens in vision transformers. *arXiv preprint arXiv:2202.12015*, 2022.

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

Xiaoqian Shen, Yunyang Xiong, Changsheng Zhao, Lemeng Wu, Jun Chen, Chenchen Zhu, Zechun Liu, Fanyi Xiao, Balakrishnan Varadarajan, Florian Bordes, et al. Longvu: Spatiotemporal adaptive compression for long video-language understanding. *arXiv preprint arXiv:2410.17434*, 2024.

Junlong Tong, Zilong Wang, YuJie Ren, Peiran Yin, Hao Wu, Wei Zhang, and Xiaoyu Shen. From static inference to dynamic interaction: A survey of streaming large language models. *arXiv preprint arXiv:2603.04592*, 2026.

Xiao Wang, Qingyi Si, Shiyu Zhu, Jianlong Wu, Li Cao, and Liqiang Nie. Adaretake: Adaptive redundancy reduction to perceive longer for video-language understanding. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 5417–5432, 2025a.

Yiyu Wang, Xuyang Liu, Xiyan Gui, Xinying Lin, Boxue Yang, Chenfei Liao, Tailai Chen, and Linfeng Zhang. Accelerating streaming video large language models via hierarchical token compression. *arXiv preprint arXiv:2512.00891*, 2025b.

Hao Wu, Yingqi Fan, Jinyang Dai, Junlong Tong, Yunpu Ma, and Xiaoyu Shen. Hidrop: Hierarchical vision token reduction in mllms via late injection, concave pyramid pruning, and early exit. *arXiv preprint arXiv:2602.23699*, 2026a.

Hao Wu, Junlong Tong, Xudong Wang, Yang Tan, Changyu Zeng, Anastasia Antsiferova, and Xiaoyu Shen. From data to model: A survey of the compression lifecycle in mllms. 2026b.

Shiwei Wu, Joya Chen, Kevin Qinghong Lin, Qimeng Wang, Yan Gao, Qianli Xu, Tong Xu, Yao Hu, Enhong Chen, and Mike Zheng Shou. Videollm-mod: Efficient video-language streaming with mixture-of-depths vision computation. *Advances in Neural Information Processing Systems*, 37: 109922–109947, 2024.

Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, et al. Pyramiddrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. *arXiv preprint arXiv:2410.17247*, 2024.

Haomiao Xiong, Zongxin Yang, Jiazuo Yu, Yunzhi Zhuge, Lu Zhang, Jiawen Zhu, and Huchuan Lu. Streaming video understanding and multi-round interaction with memory-enhanced knowledge. *arXiv preprint arXiv:2501.13468*, 2025.

Yi Xu, Yuxin Hu, Zaiwei Zhang, Gregory P Meyer, Siva Karthik Mustikovela, Siddhartha Srinivasa, Eric M Wolff, and Xin Huang. Vlm-ad: End-to-end autonomous driving through vision-language model supervision. In *Conference on Robot Learning*, pp. 3778–3803. PMLR, 2025.

Zhenhua Xu, Yujia Zhang, Enze Xie, Zhen Zhao, Yong Guo, Kwan-Yee K Wong, Zhenguo Li, and Hengshuang Zhao. Drivegpt4: Interpretable end-to-end autonomous driving via large language model. *IEEE Robotics and Automation Letters*, 9(10):8186–8193, 2024.

Yanlai Yang, Zhuokai Zhao, Satya Narayan Shukla, Aashu Singh, Shlok Kumar Mishra, Lizhu Zhang, and Mengye Ren. Streammem: Query-agnostic kv cache memory for streaming video understanding. *arXiv preprint arXiv:2508.15717*, 2025.

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.

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*, 2024a.

Haowei Zhang, Shudong Yang, Jinlan Fu, See-Kiong Ng, and Xipeng Qiu. Hermes: Kv cache as hierarchical memory for efficient streaming video understanding. *arXiv preprint arXiv:2601.14724*, 2026a.

Jialiang Zhang, Junlong Tong, Junyan Lin, Hao Wu, Yirong Sun, Yunpu Ma, and Xiaoyu Shen. Think-as-you-see: Streaming chain-of-thought reasoning for large vision-language models. *arXiv preprint arXiv:2603.02872*, 2026b.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. Sparsevlm: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*, 2024b.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. Llava-video: Video instruction tuning with synthetic data. *arXiv preprint arXiv:2410.02713*, 2024c.

Baining Zhao, Jianjie Fang, Zichao Dai, Ziyou Wang, Jirong Zha, Weichen Zhang, Chen Gao, Yue Wang, Jinqiang Cui, Xinlei Chen, et al. Urbanvideo-bench: Benchmarking vision-language models on embodied intelligence with video data in urban spaces. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32400–32423, 2025.

Yulin Zou, Yan Chen, Wenyan Chen, JooYoung Park, Shivaraman Nitin, Luo Tao, Francisco Romero, and Dmitrii Ustiugov. Codecsight: Leveraging video codec signals for efficient streaming vlm inference. *arXiv e-prints*, pp. arXiv–2604, 2026.

A Experimental Details

A.1 Dataset Preprocessing

TimeChat-Online-139K is directly adopted from TimeChat-Online Yao et al. (2025). For LLaVA-Video-100K Zhang et al. (2024c), we use the video list provided by TimeChat-Online. During preprocessing, videos are sampled at 1 FPS. For video sequences longer than 64 frames, we apply adaptive-stride uniform sampling during training, so that the retained frames cover a sufficiently broad temporal range.

A.2 Evaluation Benchmarks

We evaluate coordinated streaming on five benchmark sources that cover complementary aspects of streaming video understanding. **StreamingBench** Lin et al. (2026a) evaluates streaming video understanding through time-dependent questions asked along the video timeline; we use its Real-time Visual Understanding subset, which focuses on recognizing objects, actions, scene changes, and other visual evidence available at the current streaming moment. **OVO-Bench** Li et al. (2025c) evaluates online video understanding with timestamped queries under backward tracing, real-time perception, and forward active-response settings, allowing us to test whether bounded context affects different temporal reasoning modes. **StreamBench** Xiong et al. (2025) is designed for streaming video understanding and multi-round interaction, and is used to assess long-duration online comprehension with memory-intensive queries. **VStream-QA** Zhang et al. (2024a) provides timestamped question-answer pairs over ego-centric and movie videos; we report results on VStream-Ego and VStream-Movie to separate online egocentric understanding from long-form narrative reasoning. **ETBench** Qian et al. (2025) contains event-level and time-sensitive video-language tasks. We use the six ETBench subsets most suitable for streaming video understanding: Charades-STA, Ego4D-NLQ, QA-Ego4D, STAR, Perception Test, and THUMOS14. These subsets involve timestamped event localization, action recognition, or question answering grounded in specific temporal evidence, so they naturally test whether a model can preserve event-level information while processing videos incrementally.

Baseline protocol. Our primary comparisons use the same Qwen2.5-VL backbone with traditional full-history streaming prefill as the reference setting. This within-backbone protocol isolates how the coordinated streaming controls affect latency, memory, and accuracy under a fixed model. Because existing streaming VideoLLMs differ in backbone, instruction data, preprocessing, and memory design, cross-system accuracy comparisons conflate those factors with the pipeline itself; the stage-wise latency, ingestion FPS, TTFT, and peak-memory measurements central to our analysis are also not uniformly reported in prior work.

A.3 Coordinated Streaming Implementation

Training setup. For the coordinated streaming variants, we fine-tune Qwen2.5-VL-Instruct backbones with full-parameter supervised fine-tuning on the union of TimeChat-Online-139K and LLaVA-Video-100K. The ViT encoder is frozen, while the multimodal aligner and language model are updated. Training uses bfloat16 precision with DeepSpeed ZeRO-2 on up to four GPUs. We set the per-device batch size to 1, gradient accumulation steps to 16, learning rate to 1×10^{-5}, warmup ratio to 0.05, maximum sequence length to 8192, and train for one epoch. Visual inputs are capped with a pixel budget of 90K pixels for both images and videos, corresponding to an approximately 300×300 resolution budget before model-specific resizing.

A.4 Additional Robustness and Deployment Analyses

Hardware sensitivity. Absolute latency depends on the accelerator and backend. We therefore repeat a matched profiling run on an NVIDIA A100-SXM4-80GB, an NVIDIA RTX 5880 Ada Generation, and an NVIDIA GeForce RTX 4090 D using the same backbone, input, preprocessing path, retention ratio, and attention window. Table 6 shows that the identity of the slowest stage changes across devices: preprocessing is relatively more expensive on the A100, whereas ViT encoding becomes dominant for 32-frame chunks on the RTX GPUs. Nevertheless, every tested configuration sustains

more than 100 FPS. This cross-device run uses matched conditions but an independently collected profiling sweep, so its A100 rows differ slightly from the operating points in Table 2.

Table 6: Cross-GPU stage-wise profiling with Qwen2.5-VL-3B, simulated 720p input, CUDA visual preprocessing, 50% token retention, and acn=4.

GPU	c	VP	ViT	Drop	Prefill	FPS
NVIDIA A100-SXM4-80GB	8	68.37	62.42	21.25	40.57	117
NVIDIA RTX 5880 Ada Generation	8	38.34	59.00	13.96	41.84	136
NVIDIA GeForce RTX 4090 D	8	32.24	59.20	11.01	41.39	135
NVIDIA A100-SXM4-80GB	32	265.69	231.97	86.95	124.36	120
NVIDIA RTX 5880 Ada Generation	32	162.76	315.58	49.99	157.67	101
NVIDIA GeForce RTX 4090 D	32	124.50	281.88	41.65	159.56	114

Compatibility with alternative dropping strategies. To test whether coordinated streaming depends on its default cosine-similarity rule, we replace only the token-dropping module with strategies reimplemented from Flash-VStream Zhang et al. (2024a) and StreamingTOM Chen et al. (2025), while keeping the backbone and coordinated attention setting fixed. Table 7 shows that all three strategies operate within the same pipeline. The default rule attains the highest StreamingBench accuracy, while StreamingTOM gives the strongest OVO-Bench overall result and backward-tracing score. These are controlled plug-in comparisons rather than comparisons with the complete official systems, and they demonstrate that the stage-wise framework is not tied to a single dropping heuristic.

Table 7: Plug-in comparison of token-dropping strategies under the same Qwen2.5-VL-3B coordinated setting (c=4, acn=4, and ur=32).

Dropping Strategy	StreamingBench	OVO-Bench			
		RT	BT	FAR	All
Flash-VStream Zhang et al. (2024a)	65.00	**55.31**	37.95	30.74	44.83
StreamingTOM Chen et al. (2025)	65.24	53.82	**41.31**	**32.69**	**45.41**
ViCoStream	**70.68**	**55.31**	37.47	32.54	45.16

Motion-adaptive threshold. Fixed similarity thresholds may retain too little evidence when motion is high. As an optional training-free extension, we compute the mean cosine similarity between adjacent frames in each chunk, define the motion score $m = 1 - \mathrm{mean}(\mathrm{sim})$, and set

$$s_{\mathrm{adapt}} = \min(s_{\max}, \max(s_0, s_0 + \beta m)) \,. \tag{4}$$

Because a higher threshold relaxes token removal in our formulation, high-motion chunks retain more visual tokens. We evaluate this variant with s_0=0.25, $s_{\max}$=0.5, acn=4, and ur=16. Table 8 shows that all tested adaptive settings improve StreamingBench and OVO-All over the fixed-threshold control, with β=0.1 producing the highest OVO-All score. This variant is a robustness extension and is not used in the default ViCoStream results.

Table 8: Training-free motion-adaptive token dropping.

Rule	β	StreamingBench	OVO-Bench			
			RT	BT	FAR	All
Fixed	0.0	69.48	54.75	39.64	32.33	45.37
Adaptive	0.1	69.84	**56.57**	40.49	32.01	**46.41**
Adaptive	0.2	**69.92**	56.07	40.40	**32.41**	46.24
Adaptive	0.3	**69.92**	56.12	**40.70**	32.07	46.25

Practical configuration guidance. A latency-oriented deployment can start from (c, acn, s, ur)=$(4, 4, 0.25, 32)$. Relative to frame-level processing, c=4 amortizes small ViT and model invocations while keeping per-chunk waiting time low. When maximum ingestion throughput is the priority, c can be increased to 8 or 16, at the cost of higher per-chunk visual-processing latency. The moderate threshold s=0.25 is preferable for general use; s=0.1 should be reserved for stricter resource budgets because memory-heavy tasks are more sensitive to aggressive dropping. A

retrieval budget of ur=32 is a practical default, while ur=64 provides broader temporal evidence for backward tracing, movie-level understanding, and event reasoning at a higher query-side cost.

A.5 Token-Dropping Compression Ablations

In Table 9, we report the full temporal-compression (Left), spatial-compression (Middle), and combined-compression (Right) ablations summarized in the main text. The left and middle panels use (c, acn, ur)=$(4, 4, 32)$, while the right panel uses $(4, 4, 64)$; all experiments use Qwen2.5-VL-3B-Instruct.

Table 9: (Left) Temporal token compression under incremental streaming. (Middle) Spatial token compression strategies under the same setting. (Right) Component-wise ablation combining spatial and temporal compression.

Threshold (s)	SBench	OVO-Bench		
		RT	BT	FAR
s=1.00$_{\downarrow 0\%}$	70.32	56.72	41.04	34.04
s=0.50$_{\downarrow 19\%}$	68.48	57.52	39.06	32.88
s=0.25$_{\downarrow 60\%}$	67.64	55.25	37.33	33.42
s=0.20$_{\downarrow 67\%}$	67.68	54.90	37.51	33.01
s=0.15$_{\downarrow 71\%}$	66.76	53.49	37.20	32.28
s=0.10$_{\downarrow 74\%}$	66.56	52.07	38.08	32.22

Spatial Strat.	SBench	OVO-Bench		
		RT	BT	FAR
Baseline	70.32	56.72	41.04	34.04
Avg.$_{\downarrow 71.3\%}$	66.60	53.43	36.41	31.64
Max.$_{\downarrow 71.3\%}$	65.99	52.13	37.15	32.51
Ent.$_{\downarrow 50.2\%}$	67.63	53.62	40.62	32.66
Ent.$_{\downarrow 74.7\%}$	66.51	53.22	35.69	32.85

Compress.	SBench	OVO-Bench		
		RT	BT	FAR
None	70.32	56.72	41.04	34.04
$S_{Ent.}$	67.63	53.62	40.62	32.66
$S_{Avg.}$	66.60	53.43	36.41	31.64
$T_{0.25}$	70.44	55.74	37.36	32.35
$T_{0.10}$	68.32	50.45	34.78	33.07
$S_{Ent.}$+$T_{0.25}$	65.95	54.44	36.07	31.99

The left panel shows a gradual accuracy–compression trade-off: as the temporal drop rate increases from 19% to 74%, StreamingBench accuracy decreases from 68.48 to 66.56. The effect on OVO-Bench is task dependent; forward active response remains relatively stable, whereas real-time perception becomes more sensitive under aggressive dropping. In the middle panel, entropy-based spatial selection with a 50.2% drop rate preserves backward-tracing performance particularly well, while increasing its drop rate to 74.7% removes this advantage. The right panel further shows that temporal compression alone at s=0.25 preserves StreamingBench accuracy (70.44 vs. 70.32), whereas combining it with entropy-based spatial selection lowers StreamingBench accuracy to 65.95. Together, these results indicate that the cost of token reduction depends on both the compression mechanism and the temporal demands of the evaluated task.

A.6 Frame-level Token Dropping

Table 10 reports the expanded per-task results underlying the summary in Table 5. In the frame-level setting, c=1 and every full frame is treated as an independent token-dropping unit using a per-frame temporal-similarity threshold. For comparison, the chunk-level settings use c=4 and reuse the chunked prefilling and bounded attention described in Section 4. The corresponding chunk-level rows are taken from Table 12.

The results expose the largest granularity-dependent differences on the Counting task: frame-level dropping becomes substantially less robust under the aggressive threshold s=0.1, whereas chunk anchors preserve a more stable temporal unit for token reduction.

Table 10: Expanded StreamingBench per-task accuracy of frame- and chunk-level token dropping on Qwen2.5-VL-3B-Instruct. All settings use the same backbone, training data, and evaluation protocol.

acn	s	OP	CR	CS	ATP	EU	TR	PR	SU	ACP	CT	All
Traditional Streaming Prefill												
Baseline		74.53	75.78	74.13	76.92	74.84	74.45	70.37	63.41	66.19	46.81	70.32
Frame-Level ViCoStream (c=1)												
16	0.25	73.98	70.31	74.76	76.60	76.73	70.09	64.81	64.63	65.06	40.43	68.80
16	0.1	71.00	71.88	73.82	71.15	72.33	69.47	64.81	64.23	66.19	28.19	66.48
Chunk-Level ViCoStream (c=4)												
4	0.25	73.17	69.53	75.39	77.88	70.44	74.45	72.22	63.41	69.60	51.06	70.68
4	0.1	69.92	70.31	74.45	76.28	72.96	69.78	70.37	62.20	68.47	39.89	68.28
8	0.25	73.44	71.88	76.66	77.24	74.84	74.45	69.44	63.01	67.05	48.40	70.48
8	0.1	68.83	69.53	76.03	76.28	71.70	67.60	68.52	63.41	65.91	40.96	67.68

A.7 Detailed Benchmark Results

StreamBench and VStream. Table 11 reports accuracy and score on StreamBench, RVS-Ego, and RVS-Movie. StreamBench remains robust under coordinated streaming: moderate dropping improves the 3B model over the full-history baseline and only mildly reduces the 7B model. RVS-Ego also benefits from bounded processing in several settings, indicating that egocentric streaming questions often rely on recent or locally retrievable evidence. In contrast, RVS-Movie is consistently more fragile, with most coordinated settings below the full-history baseline, suggesting that movie-level understanding depends more heavily on long-range narrative context. These results complement the main table by showing that the same efficiency controls are most reliable for online visual understanding and more costly for long-form story-level reasoning.

Table 11: Expanded results on StreamBench, RVS-Ego (RVS-E), and RVS-Movie (RVS-M) using Qwen2.5-VL backbones in Table 3. Accuracy and Score metrics are judged by GPT-3.5-Turbo.

Model	acn	ur	s	StreamBench		RVS-E		RVS-M	
				Accuracy	Score	Accuracy	Score	Accuracy	Score
Traditional Streaming Prefill (Baseline)									
Qwen2.5-VL-3B	∞	∞	–	48.37	3.07	34.59	3.21	33.91	2.95
Qwen2.5-VL-7B	∞	∞	–	54.90	3.22	34.86	3.21	34.60	2.98
ViCoStream†									
Qwen2.5-VL-3B†	4	64	0.25	49.72	3.12	35.22	3.35	31.97	2.94
	8	64	0.25	49.06	3.10	34.12	3.24	31.55	2.90
	4	64	0.1	46.35	2.97	34.40	3.33	29.04	2.87
	8	64	0.1	47.52	3.00	35.43	3.29	32.25	2.91
	4	32	0.25	49.94	3.12	32.29	3.24	31.98	2.92
	8	32	0.25	49.14	3.11	32.18	3.13	30.93	2.89
	4	32	0.1	46.09	2.96	30.99	3.22	28.39	2.83
	8	32	0.1	47.66	3.00	32.70	3.28	31.34	2.89
Qwen2.5-VL-7B†	4	64	0.25	54.07	3.23	36.68	3.18	31.67	2.90
	8	64	0.25	53.68	3.22	34.31	3.14	33.23	2.97
	4	64	0.1	51.01	3.15	32.70	3.09	28.14	2.77
	8	64	0.1	50.97	3.13	35.63	3.17	31.07	2.89
	4	32	0.25	53.51	3.22	36.71	3.17	31.70	2.92
	8	32	0.25	53.51	3.22	34.40	3.16	33.02	2.95
	4	32	0.1	50.53	3.13	32.12	3.10	28.30	2.79
	8	32	0.1	50.42	3.13	34.88	3.15	31.76	2.90

StreamingBench. Table 12 provides the per-task breakdown on StreamingBench; OP, CR, CS, ATP, EU, TR, PR, SU, ACP, and CT denote Object Perception, Causal Reasoning, Clips Summarization, Attribute Perception, Event Understanding, Text-Rich Understanding, Prospective Reasoning, Spatial Understanding, Action Perception, and Counting, respectively. Coordinated streaming is especially stable under moderate temporal dropping (s=0.25): Qwen2.5-VL-3B slightly improves over the full-history baseline, while several sub-tasks remain comparable or improve. Under stronger dropping (s=0.1), counting and text-rich understanding become more sensitive, indicating that fine-grained visual and textual cues still benefit from denser retained tokens.

OVO-Bench. Table 13 breaks OVO-Bench into real-time visual perception, backward tracing, and forward active response. Real-time perception is relatively robust to bounded attention and token dropping: for the 7B model, several coordinated settings match or exceed the baseline average for this category. Backward tracing is more sensitive, especially for the 3B model under aggressive dropping, reflecting the need to recover evidence from earlier visual history. Forward active response changes less dramatically, partly because clue-revealing response remains stable across many settings. Overall, OVO-Bench confirms that the cost of coordinated streaming is task-dependent: online perception tolerates bounded context better than memory-heavy backward reasoning.

Comparison with existing streaming systems. Table 15 places the coordinated setting in context against representative streaming systems evaluated under the same 3B-scale backbone and evaluation pipeline. Each baseline is reported in its full-history operating mode. Since none of these systems exposes a directly equivalent coordinated inference setting, the table compares different points of the accuracy–efficiency trade-off rather than matched streaming configurations. ViCoStream stays close

Table 12: Expanded StreamingBench results on the real-time visual understanding subset.

Model	acn	ur	s	OP	CR	CS	ATP	EU	TR	PR	SU	ACP	CT	All
Traditional Streaming Prefill (Baseline)														
Qwen2.5-VL-3B	∞	∞	–	74.53	75.78	74.13	76.92	74.84	74.45	70.37	63.41	66.19	46.81	70.32
Qwen2.5-VL-7B	∞	∞	–	80.49	83.59	78.86	81.41	77.36	78.19	72.22	67.48	69.32	50.53	74.60
ViCoStream†														
Qwen2.5-VL-3B†	4	64	0.25	73.98	70.31	75.71	77.56	71.07	73.83	71.30	63.41	69.60	46.81	70.44
	8	64	0.25	73.98	71.88	75.39	76.92	75.47	75.39	69.44	63.01	67.05	48.40	70.52
	4	64	0.1	70.19	71.88	75.39	75.64	72.96	69.16	69.44	62.60	67.90	40.43	68.32
	8	64	0.1	70.46	70.31	74.45	75.64	71.07	66.36	66.67	62.60	66.19	41.18	67.39
	4	32	0.25	73.17	69.53	75.39	77.88	70.44	74.45	72.22	63.41	69.60	51.06	70.68
	8	32	0.25	73.44	71.88	76.66	77.24	74.84	74.45	69.44	63.01	67.05	48.40	70.48
	4	32	0.1	69.92	70.31	74.45	76.28	72.96	69.78	70.37	62.20	68.47	39.89	68.28
	8	32	0.1	68.83	69.53	76.03	76.28	71.70	67.60	68.52	63.41	65.91	40.96	67.68
Qwen2.5-VL-7B†	4	64	0.25	76.42	78.12	78.86	77.24	72.33	74.77	75.93	67.48	68.47	39.36	71.64
	8	64	0.25	75.88	78.91	79.50	77.24	77.36	74.77	76.85	66.26	68.75	37.23	71.80
	4	64	0.1	74.80	78.12	76.66	77.88	72.33	71.34	77.78	66.26	66.76	28.88	69.71
	8	64	0.1	75.27	77.34	77.29	77.17	73.42	70.72	74.07	65.85	67.24	38.42	70.42
	4	32	0.25	76.15	78.12	79.18	76.60	73.58	73.52	75.00	68.29	67.90	38.83	71.40
	8	32	0.25	76.42	78.91	80.13	75.96	75.47	73.83	76.85	66.67	67.61	37.77	71.48
	4	32	0.1	75.27	77.34	77.29	78.46	72.33	69.78	75.00	66.26	66.95	27.07	69.56
	8	32	0.1	74.72	77.34	77.29	77.17	74.36	71.34	74.04	66.12	68.19	36.81	70.44

Table 13: Expanded OVO-Bench results following the standard task order of Real-Time Visual Perception, Backward Tracing, and Forward Active Response.

Model	acn	ur	s	Real-Time Visual Perception							Backward Tracing				Forward Active Response				All
				OCR	ACR	ATR	STU	FPD	OJR	Avg.	EPM	ASI	HLD	Avg.	REC	SSR	CRR	Avg.	
Traditional Streaming Prefill (Baseline)																			
Qwen2.5-VL-3B	∞	∞	–	67.11	44.04	64.66	40.45	65.35	58.70	56.72	51.85	54.05	17.20	41.04	27.37	34.75	40.00	34.04	47.13
Qwen2.5-VL-7B	∞	∞	–	67.11	47.71	70.43	43.82	71.29	60.33	60.11	56.23	60.14	10.22	42.19	26.33	38.16	40.43	34.97	49.35
ViCoStream†																			
Qwen2.5-VL-3B†	4	64	0.25	63.76	45.87	62.93	40.45	64.36	57.07	55.74	47.81	51.35	12.90	37.36	23.59	33.47	40.00	32.35	45.30
	8	64	0.25	65.77	49.54	62.93	38.20	63.37	57.07	56.15	51.18	54.73	17.20	41.04	21.64	34.58	40.00	32.07	46.35
	4	64	0.1	52.35	40.37	55.17	40.45	64.36	50.00	50.45	48.15	51.35	4.84	34.78	25.30	33.92	40.00	33.07	42.19
	8	64	0.1	50.34	44.04	58.62	39.89	62.38	51.09	51.06	47.14	53.38	18.28	39.60	19.17	35.08	40.00	31.41	43.28
	4	32	0.25	61.74	45.87	62.93	39.33	64.36	57.61	55.31	48.15	51.35	12.90	37.47	24.42	33.21	40.00	32.54	45.16
	8	32	0.25	65.10	46.79	63.79	39.33	65.35	57.61	56.33	48.15	54.73	16.67	39.85	21.87	34.68	40.00	32.18	46.17
	4	32	0.1	51.68	41.28	57.76	39.89	65.35	50.54	51.08	47.81	52.03	5.38	35.07	23.42	34.64	40.00	32.69	42.48
	8	32	0.1	50.34	44.95	58.62	39.33	63.37	53.80	51.73	49.49	53.38	18.82	40.56	18.78	35.28	40.00	31.35	43.85
Qwen2.5-VL-7B†	4	64	0.25	70.47	48.62	67.24	50.00	68.32	55.98	60.11	51.85	61.49	13.98	42.44	27.67	35.68	41.67	35.01	49.41
	8	64	0.25	76.51	48.62	62.93	48.31	69.31	60.87	61.09	56.57	62.84	19.89	46.43	28.36	35.07	40.00	34.48	50.77
	4	64	0.1	62.42	49.54	62.07	46.07	72.28	53.80	57.70	50.34	60.81	15.30	42.15	24.62	33.46	40.00	32.69	47.56
	8	64	0.1	65.10	47.71	62.93	44.38	70.30	52.17	57.10	53.20	59.46	11.83	41.50	24.90	34.82	40.00	33.24	47.23
	4	32	0.25	71.81	44.95	67.24	49.44	70.30	55.43	59.86	53.87	62.16	15.59	43.88	27.27	36.38	42.08	35.24	49.71
	8	32	0.25	75.84	46.79	64.66	46.07	69.31	58.70	60.23	55.56	62.16	21.51	46.41	28.19	35.40	40.00	34.53	50.35
	4	32	0.1	61.07	47.71	62.07	45.51	70.30	51.63	56.38	49.83	59.46	15.05	41.45	25.39	33.28	40.00	32.89	46.77
	8	32	0.1	63.09	45.87	62.83	43.60	74.26	53.25	57.15	52.94	60.81	12.37	42.04	24.25	34.63	40.00	32.96	47.32

Table 14: Expanded ETBench task-level results. ECA, GVQ, RAR, and RVQ are accuracy-based tasks; EPM is temporal grounding with mIoU/F1; TAL-PT, TAL-TH, and TVG are F1-based temporal localization tasks; TEM reports mean recall.

Model	acn	ur	s	ECA ↑	EPM (mIoU↑ / F1↑)	GVQ ↑	RAR ↑	RVQ ↑	TAL-PT ↑	TAL-TH ↑	TEM ↑	TVG ↑
Traditional Streaming Prefill (Baseline)												
Qwen2.5-VL-3B	∞	∞	–	37.80	1.29 / 1.60	53.45	34.80	51.60	18.63	13.09	16.80	27.30
Qwen2.5-VL-7B	∞	∞	–	45.00	1.66 / 2.10	56.55	36.60	54.60	20.17	12.61	26.70	33.35
ViCoStream†												
Qwen2.5-VL-3B†	4	64	0.25	37.20	1.58 / 2.05	52.41	35.80	48.20	21.74	14.57	20.80	32.30
	4	64	0.1	37.20	1.62 / 1.90	54.83	36.00	46.20	20.84	14.41	16.45	30.00
	4	32	0.25	37.20	1.54 / 1.75	52.07	35.80	48.20	21.74	14.66	20.40	32.10
	4	32	0.1	37.20	1.62 / 1.95	54.48	35.20	45.60	21.63	14.42	16.35	30.35
Qwen2.5-VL-7B†	4	64	0.25	37.60	2.02 / 2.55	53.79	36.60	51.20	17.93	13.54	21.90	28.05
	4	64	0.1	37.00	1.21 / 1.30	53.79	33.40	46.80	12.37	13.70	21.60	26.15
	4	32	0.25	37.60	2.04 / 2.50	54.48	35.60	51.20	17.95	12.51	21.80	28.30
	4	32	0.1	37.00	1.25 / 1.50	53.10	33.60	46.60	12.22	12.66	21.40	26.40

on the aggregate metrics despite attending to a bounded visual context: it trails StreamingTOM by 1.56 points on StreamingBench and by 0.13 points on the OVO-Bench average. The per-category columns show where that budget is spent: the gap is smallest for forward active response and largest for backward tracing, which depends most on retained history. Table 7 complements this comparison from the opposite direction, by measuring the same dropping modules inside the coordinated framework.

Table 15: Comparison with representative streaming VideoLLM systems on Qwen2.5-VL-3B.

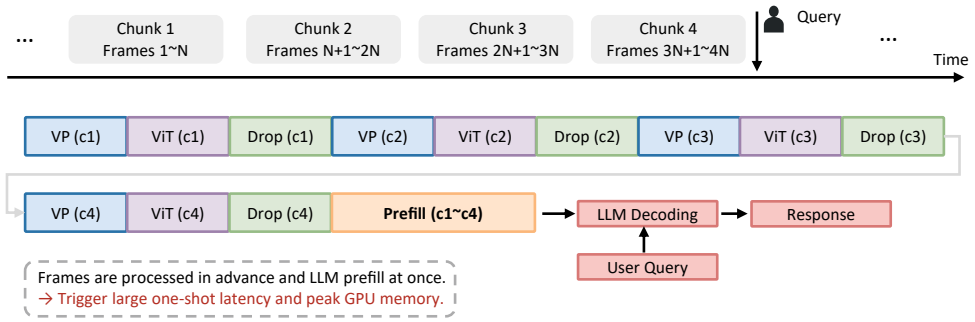
Method	Setting	StreamingBench	OVO-Bench			
			RT	BT	FAR	All
TimeChat-Online Yao et al. (2025)	Full history	71.64	58.78	40.08	35.12	44.66
Flash-VStream Zhang et al. (2024a)	Full history	70.92	59.10	42.75	34.72	45.53
StreamingTOM Chen et al. (2025)	Full history	72.00	58.88	42.19	35.22	45.43
ViCoStream	Coordinated	70.44	55.74	37.36	32.35	45.30

B Streaming Paradigms

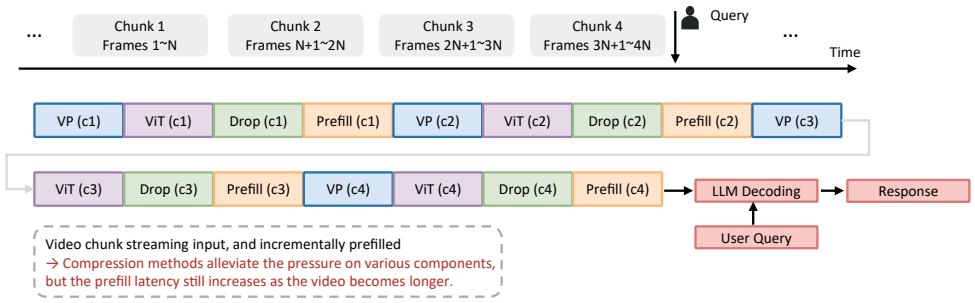
Figure 6(a) illustrates **delayed streaming**, which corresponds to the fake-streaming setting mentioned in the Introduction. The complete video is pre-cached before inference, even if frames are originally delivered as a stream, and the cached context may be compressed before the user query arrives. The LLM performs a single one-shot prefill only at query time, analogous to a human watching the entire video first and only beginning to reason when asked a question. This paradigm has a light early-stage burden, since it mainly runs visual encoding when no LLM-side prefill is performed. However, if the cached context has not been sufficiently compressed, the query-time prefill concentrates a long visual history into one forward pass, producing large latency and GPU peak-memory spikes. It is therefore unsuitable for strict real-time streaming, where queries may arrive at arbitrary moments and responses must remain bounded.

Figure 6(b) illustrates **continuous streaming**, where video chunks are received incrementally and the LLM performs prefilling after each new chunk arrives. This is analogous to a human reasoning after each newly observed video segment before continuing to watch. Such systems can apply local acceleration techniques within individual components, such as token compression or memory reduction. However, since visual preprocessing, visual encoding, token dropping, and LLM prefilling are still executed serially for each chunk, the per-chunk latency may grow with the accumulated visual history. For long streams, the system may fail to finish all stages within one chunk interval, making long streaming video inference difficult.

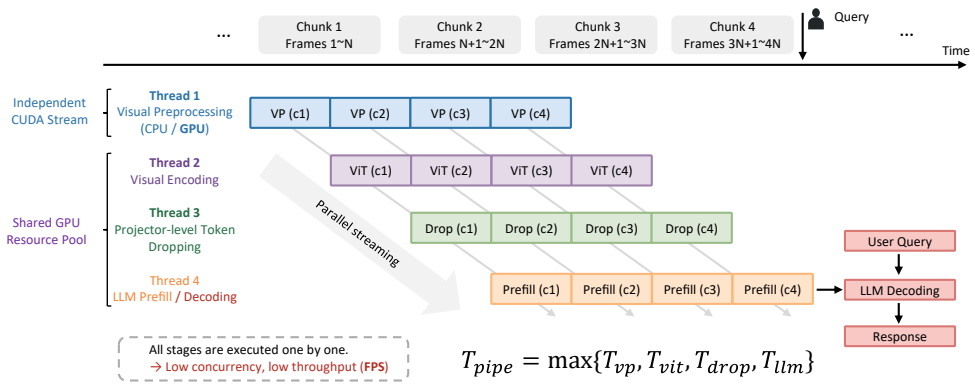
Figure 6(c) illustrates our **stage-wise coordinated streaming**. It also receives video incrementally, but decomposes continuous inference into stage-wise pipeline slots: after warm-up, visual preprocessing, visual encoding, token dropping, and LLM prefilling operate on different chunks concurrently. The maximum supported streaming rate is therefore governed by $T_{\mathrm{pipe}} = \max\{T_{\mathrm{vp}}, T_{\mathrm{vit}}, T_{\mathrm{drop}}, T_{\mathrm{llm}}\}$ rather than the sum of all stage latencies. This formulation exposes the true streaming bottleneck and reveals how it migrates after individual stages are optimized. In our experiments, stage-wise streaming supports over 130 FPS while keeping per-chunk latency and GPU peak memory bounded over long streams.



(a) Delayed streaming. The full video is available before inference and can be cached or compressed in advance.



(b) Continuous streaming. Video chunks arrive incrementally, but stages are executed sequentially for each chunk.



(c) Stage-wise coordinated streaming (Ours). Different chunks occupy different stages in a pipeline, enabling stage-wise overlap.

Figure 6: Comparison of streaming inference paradigms for VideoLLMs.

C TOKEN-DROPPING VISUALIZATION

Figure 7 shows how the intra-chunk temporal token-dropping rule acts on a 64-second StreamingBench video sampled at 1 FPS. With chunk-wise processing and threshold s=0.25, tokens that remain redundant within each chunk are effectively removed, as indicated by the white patches. Meanwhile, the first frame of each chunk is kept as an anchor, preserving rich representative information about what occurs in that segment. This design maintains a high drop rate while reducing the risk of discarding all evidence for a temporally stable event.

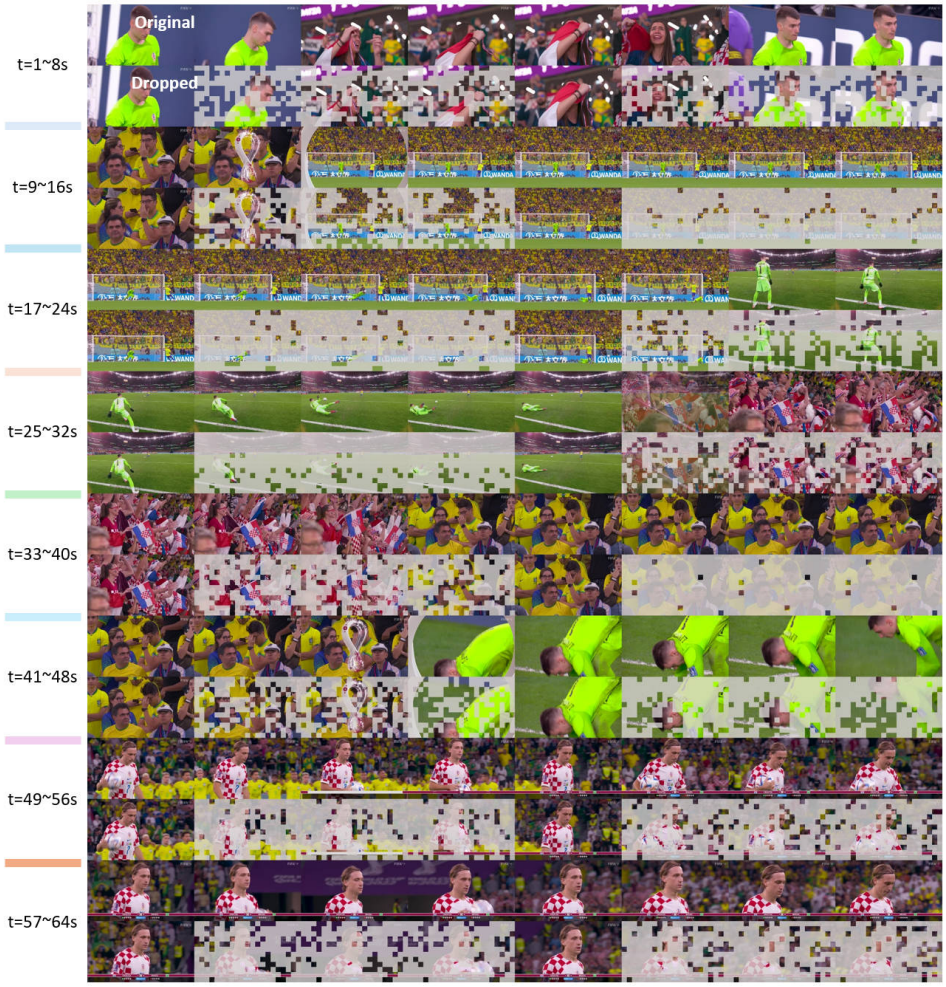


Figure 7: Visualization of intra-chunk temporal token dropping on StreamingBench. White patches indicate dropped visual tokens, while the remaining regions are retained for LLM prefilling.