STREAMPRUNE: CAUSAL AND QUERY-FREE TOKEN PRUNING FOR ACCELERATING STREAMING VIDEO MLLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal Large Language Models (MLLMs) have made significant progress in visual-language reasoning, including long- and short-term video understanding, but sampling video at high frame rates remains computationally expensive. Perceiving fast-changing or transient events requires dense frame sampling, yet the cost of processing dense visual tokens scales with frame rate. Moreover, consecutive frames in a video stream are often highly similar, causing much of this computation to be spent on redundant content. We propose **StreamPrune**, a training-free, query-free, plug-and-play token pruning module for streaming video LLMs that discards visually redundant tokens using only inter-frame similarity, measured by the Temporal Token Similarity (TTS) module. It is independent of the downstream query and transfers across multiple streaming MLLM backbones.We further introduce the Content-Adaptive Caching (CAC) module, which retains only temporally novel tokens and organizes them in a content-adaptive cache that is compatible with existing streaming retrieval. Across six streaming video benchmarks, StreamPrune reduces visual tokens by **92.6** $\pm$ **1.8%** and increases encoding throughput by **6.18**$\times$, at an average accuracy drop of only **0.8** $\pm$ **2.9** points (both spreads are standard deviations across the six benchmarks), while outperforming existing pruning methods on **StreamingBench** and **StreamBench**. Code will be released publicly after the review process.

1 INTRODUCTION

Video understanding is central to how multimodal large language models are deployed, and most progress on it assumes offline access: the clip is complete before the model sees it, and frames may be sampled freely from anywhere in it before an answer is produced. A large class of applications cannot wait for the video to end. An assistant narrating a live feed, a driving system reasoning about the road ahead, and a robot acting on what it has just observed must all answer questions about video that is still arriving. Streaming video MLLMs are built for this setting: each frame is encoded as it appears into a growing key-value cache, and questions are answered from that cache at any point in the stream (Chen et al., 2024a; Zhang et al., 2025a; Di et al., 2025).

Current streaming models such as ReKV, LongVA, and Flash-VStream (Di et al., 2025; Zhang et al., 2025a;b) assign a fixed number of visual tokens to every frame, and therefore a fixed amount of prefill compute and KV-cache memory, regardless of how much new content the frame contains. Their cost thus grows linearly with frame rate, which is why streaming systems typically operate near 1 FPS (Li et al., 2025). Yet the amount of new information does not grow that way: consecutive frames are highly redundant, and this redundancy increases as the sampling rate rises. Allocating compute by frame rate rather than by temporal novelty is what prevents streaming models from sampling densely and running fast.

We observe substantial redundancy in the video stream, as shown in Figure 1, and hypothesize that this cost need not be paid in full. Across six streaming video benchmarks, we find that accuracy remains almost unchanged even under heavy token pruning. This indicates that the majority of visual tokens encoded from a video stream are redundant rather than informative.

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.

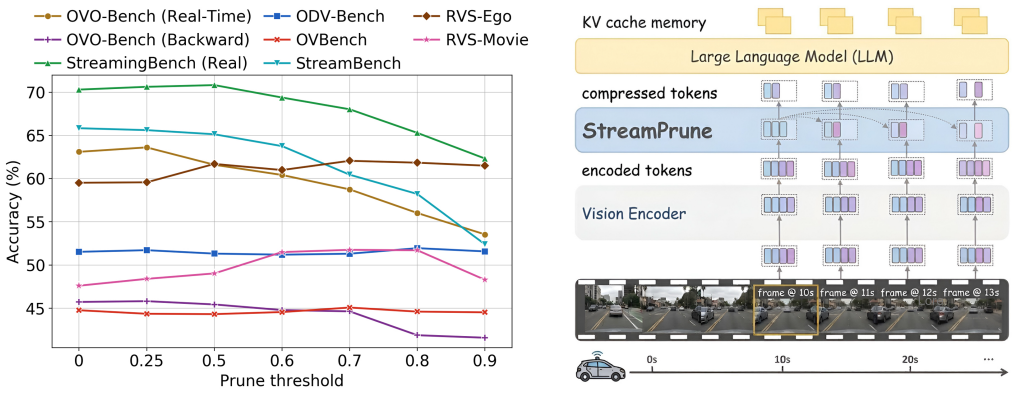

Figure 1: **Left:** Accuracy as the pruning threshold τ varies, shown for each of the six evaluation benchmarks (OVO-Bench and VStream-QA are reported on their two sub-settings each); $\tau = 0$ is the unpruned baseline. Accuracy is largely flat up to $\tau = 0.50$, indicating that most of the visual tokens encoded from a stream are redundant rather than informative. **Right:** Overview of StreamPrune, a causal token-pruning method for streaming video LLMs. The video stream is first encoded, then passed through StreamPrune, which prunes the vision tokens according to their temporal novelty and forwards only the survivors to the LLM decoder, which answers the question asked about the stream.

Existing token pruning methods for video LLMs reduce spatial redundancy but not temporal redundancy. The predominant approach operates within a single frame: ToMe (Bolya et al., 2023) merges similar patches through bipartite matching in the vision transformer, while VisionZip (Yang et al., 2025), LLaVA-PruMerge (Shang et al., 2025), and FastV (Chen et al., 2024b) rank tokens according to `[CLS]` or query attention. Such criteria are compatible with causal processing, but they evaluate each frame independently of those already encoded. A scene that remains unchanged is therefore encoded at full cost at every timestep, and the redundancy that accumulates over a stream is never removed. Methods that do operate across frames relax one of the two constraints that streaming imposes: Chat-UniVi (Jin et al., 2024) and TESTA (Ren et al., 2023) merge and cluster along the temporal axis, but require the complete token set of a segment before any pruning decision is made, while PruneVid (Huang et al., 2025a) and MovieChat+ (Song et al., 2025) select tokens according to their relevance to the downstream question. In a streaming setting, at the time a frame is encoded, neither the frames that follow it nor the query eventually asked about the video are yet known, so pruning decisions must rely solely on the information observed so far.

We introduce **StreamPrune**, a lightweight, plug-and-play module that prunes redundant visual tokens causally, as the stream arrives. A Temporal Token Similarity (TTS) step compares each incoming frame's tokens against those already kept at the same spatial position and discards whatever is not new, while a Content-Adaptive Caching (CAC) step packs only the novel tokens across frame boundaries into fixed-size cache blocks.

We summarize our contributions as follows:

1. We introduce **StreamPrune**, a causal, query-free, training-free, plug-and-play token pruning module built on **Temporal Token Similarity** (TTS), which prunes the video stream without access to future frames or the downstream query.

2. We introduce a **Content-Adaptive Caching** (CAC) module that stores the KV-cache only for temporally novel tokens in streaming video MLLMs.

3. We conduct extensive evaluation and analysis across multiple benchmarks, showing that StreamPrune reduces visual tokens by **92.6** $\pm$ **1.8%** and increases throughput by **6.18**$\times$, with an average accuracy drop of only **0.8** $\pm$ **2.9** points across six benchmarks, outperforming existing methods on StreamBench and StreamingBench.

2 METHODOLOGY

Our preliminary analysis on long-video benchmarks reveals significant redundancy in the encoded tokens (Figure 1), which StreamPrune removes as the stream is encoded. StreamPrune adds two components to a streaming video MLLM, both active during stream encoding, as illustrated in Figure 2. Temporal Token Similarity (TTS; Section 2.1) decides, for each arriving frame, which patch tokens carry content the model has not already seen, and discards the rest. Content-Adaptive Caching (CAC; Section 2.2) stores the variable number of retained tokens so that the cache grows with novel content rather than with elapsed time. Both components are training-free and require neither the user's query nor future frames.

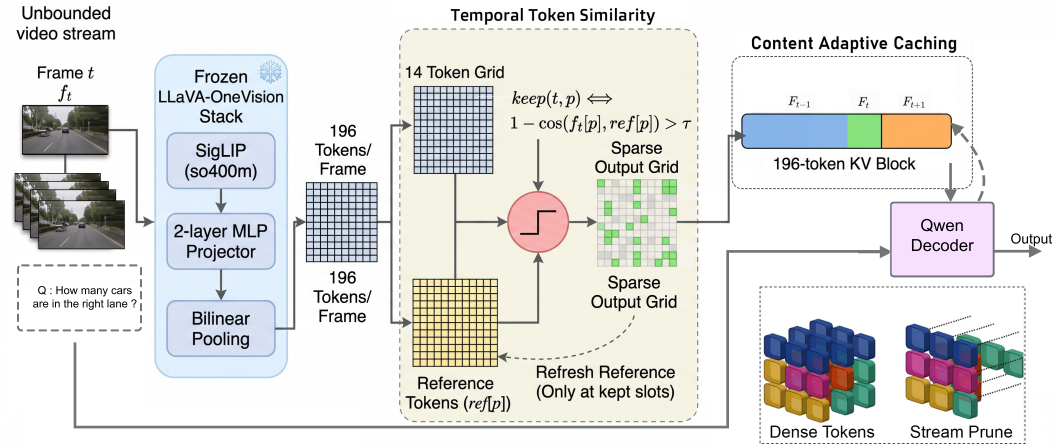

Figure 2: Overview of StreamPrune. The incoming video stream passes through the vision encoder to produce vision tokens. Tokens from the first frame are retained as reference tokens. Tokens from each incoming frame are then compared against the reference tokens using the keep rule described in Section 2.1; the retained tokens are appended to the buffer and fed to a modified LLM (Di et al., 2025). Without pruning, each frame contributes a dense, fixed-size block of visual tokens, so the sequence grows linearly with the number of frames streamed. After pruning, only a sparse subset of tokens is retained. **StreamPrune's token packing** is visualized at the bottom.

2.1 TEMPORAL TOKEN SIMILARITY

Given an input video stream of N frames, we denote the visual backbone by $f(\cdot)$, which maps each frame $\mathbf{x}_t$ at time t to a set of P spatial token embeddings

$$\mathbf{Z}_t = f(\mathbf{x}_t) = \{\mathbf{z}_t^1, \ldots, \mathbf{z}_t^P\},$$

where P is the number of spatial tokens. To identify temporally redundant tokens, we maintain a reference embedding $\mathbf{r}^p$ for each spatial location p, initialized with the tokens of the first frame. For each incoming frame t, the cosine similarity between the current token and its corresponding reference is computed as

$$s_t^p = \frac{\mathbf{z}_t^p \cdot \mathbf{r}^p}{\|\mathbf{z}_t^p\| \|\mathbf{r}^p\|}.$$

A token is retained according to

$$\mathsf{keep}(\mathbf{z}_t^p, \mathbf{r}^p) \iff 1 - s_t^p > \tau.$$

Here $1 - s_t^p$ is the cosine distance between the current token and its reference, and τ is the minimum change required to retain a token. Any token whose distance does not exceed τ is temporally redundant and is discarded. The reference is replaced by the retained token whenever a token is kept, and left unchanged whenever a token is dropped, so $\mathbf{r}^p$ is always the most recent token written to the buffer at position p. The keep rule therefore guarantees, by construction, that every discarded token satisfies $1 - s_t^p \leq \tau$ with respect to the token the model actually holds in memory, however long the run of dropped frames. Comparing instead against the previous frame, as RLT (Choudhury et al.,

2024) does, constrains only the cosine distance between consecutive frames, and that constraint does not compose: cosine distance is not a metric, so a sequence of steps each below τ carries no guarantee that the discarded token still lies within τ of the last token retained at that position. A slowly drifting region can thus be discarded at every step with its distance from the cached content left unconstrained. Anchoring the comparison to the last token kept replaces a per-step condition with the bound the cache actually needs.

2.2 Content-Adaptive Caching

ReKV (Di et al., 2025) writes its offloaded KV cache to CPU memory in fixed-size blocks, with one frame per block. Pruning is incompatible with this layout, since a frame leaves a variable number of surviving tokens, which rarely fill a block exactly. Padding each frame back to a full block preserves the correspondence, but leaves the cache at its original size, so pruning yields no reduction in memory; we retain this variant as an ablation (Section 3.3). Content-Adaptive Caching (CAC) instead decouples frames from blocks: the surviving tokens are packed into a contiguous stream that is cut into a new block every B tokens. Let n_t denote the number of tokens surviving pruning for the frame at time t, out of the fixed per-frame token budget, and let B denote the block size (196 tokens). Each block therefore holds only novel content, spanning many frames of a static scene or part of a single frame of a rapidly changing one, and after T frames the cache holds $\lfloor \sum_t n_t / B \rfloor$ blocks rather than T. A lightweight index records each frame's token count, so any retrieved block remains attributable to the frames it represents. Figure 3 illustrates this packing and the resulting cache layout. Pruning consequently reduces prefill cost and cache memory jointly.

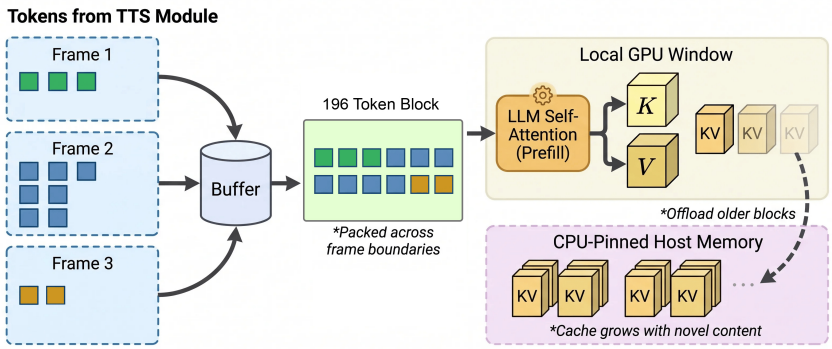

Figure 3: Content-Adaptive Caching (CAC). Tokens surviving TTS pruning carry a variable count per frame, so they are buffered and packed across frame boundaries into fixed-size 196-token blocks. Each completed block is prefilled through self-attention over a local GPU window, which produces the Key and Value tensors that populate the KV-cache; as the window fills, older blocks are offloaded to CPU-pinned host memory, so cache growth tracks novel content rather than elapsed time.

2.3 Inference Throughput

TTS and CAC jointly translate token sparsity into encoding throughput gains. Prefill is the forward pass in which the visual tokens of an arriving frame are processed by the language backbone so that their keys and values enter the cache, before any question is asked. Encoding cost is dominated by this pass, which scales with sequence length, so discarding temporally redundant tokens beforehand reduces this cost accordingly. Because the local window and the retrieval budget are measured in tokens rather than frames, the reduced per-frame footprint allows the same budget to span more of the recent past, while the slower growth of the KV cache defers offloading.

3 Experiments

Benchmarks. We evaluate the proposed method across six streaming video benchmarks: (1) **OVO-Bench**, evaluating real-time and backward tasks for timestamp-aware online video understanding (Niu et al., 2025); (2) **StreamBench**, evaluating real-time streaming video understand-

ing through object search, memory retrieval, and reasoning (Xiong et al., 2025); (3) **ODVBench**, evaluating real-time video understanding in autonomous driving scenarios (Zeng et al., 2026); (4) **OVBench**, evaluating online video understanding across past, present, and future temporal contexts (Huang et al., 2025b); (5) **VStream-QA**, evaluating long-form streaming video understanding on both egocentric (*RVS-Ego*) and movie (*RVS-Movie*) videos (Zhang et al., 2025a); and (6) **StreamingBench**, evaluating diverse real-world videos, including tutorials, movies, sports, and documentaries (Lin et al., 2026).

Implementation Details. StreamPrune is a plug-and-play module that can be integrated with any streaming video MLLM (Zhang et al., 2025a;b). We use ReKV (Di et al., 2025) as the base model for most experiments. ReKV is a training-free, end-to-end online video language model designed for streaming video question answering. Each video is sampled at 1 FPS unless stated otherwise and processed strictly causally: frames are encoded one at a time, and each question sees only the frames up to its own timestamp. Each frame is passed through the vision encoder and pooled to a fixed set of tokens, with the exact count depending on the backbone. We implement our pruner on top of the official ReKV codebase, modifying its KV-cache construction so that only novel visual content is stored. Before prefill, each frame's tokens are compared against the last kept tokens at the same spatial position, and a token is written to the cache only if it satisfies the keep rule of Section 2.1. All throughput results were obtained on a single NVIDIA A100 80GB GPU.

Baselines. We compare against two groups of baselines. The first consists of training-free token compression methods that accelerate video LLM inference by removing visual redundancy: ToMe (Bolya et al., 2023), VisionZip (Yang et al., 2025), VidCom^2 (Liu et al., 2025), and the STC variants (Wang et al., 2026). The STC variants are the closest baselines to ours, being the only ones designed for streaming: *STC-Pruner* discards redundant visual tokens hierarchically before they reach the decoder, and *STC-Cacher & Prune* pairs that pruner with a compressed KV cache, so it reduces both prefill cost and cache memory as StreamPrune does. Each is applied to the same ReKV backbone as StreamPrune, so that the comparison isolates the pruning criterion. The second consists of streaming video MLLMs evaluated end to end: VideoLLM-online (Chen et al., 2024a), Flash-VStream (Zhang et al., 2025a), and StreamChat (Xiong et al., 2025), the memory-based streaming model introduced alongside StreamBench. StreamChat is released in three configurations – *Slow*, *Base*, and *Fast* – that trade accuracy against encoding throughput; we report all three, since together they span the throughput range in which StreamPrune operates.

Table 1: **(a)** Performance and efficiency metrics evaluated at 1 FPS across different values of the pruning threshold (τ), with accuracy deltas relative to the unpruned baseline shown in parentheses. *Avg Acc* is the average accuracy across all six benchmarks; *TR(%)* is the mean token reduction; both are means over the six per-benchmark values, and where a spread is quoted for them in the text it is the standard deviation across those benchmarks; *Thr* is the encoding throughput (frames/s); and *KV (GB/h)* is the KV-cache growth rate. **(b)** Performance across different backbones, measured on StreamingBench. The row adopted as the default operating point is shaded.

(a) Pruning Performance

Model – 1 FPS	Avg Acc ↑	TR% ↑	Thr ↑	KV (GB/h)↓
ReKV-7B (Di et al., 2025)	56.1	0.0%	11	37.7
+ StreamPrune @ 0.25	56.2 (+0.1)	41.4%	19	22.7
+ StreamPrune @ 0.50	56.1 (+0.0)	73.8%	37	10.6
+ StreamPrune @ 0.60	55.7 (-0.4)	85.2%	53	6.3
+ StreamPrune @ 0.70	55.3 (-0.8)	92.6%	68	3.6
+ StreamPrune @ 0.80	53.9 (-2.2)	96.4%	77	2.2
+ StreamPrune @ 0.90	52.0 (-4.1)	98.1%	81	1.6

(b) Different Backbone

Backbone	Acc	Thr	TR%
ReKV-7B	69.1	11	0%
+ StreamPrune @ 0.50	70.8	37	73.8%
+ StreamPrune @ 0.70	68.0	68	92.6%
LongVA-7B	62.5	18	0%
+ StreamPrune @ 0.50	61.3	42	71.1%
+ StreamPrune @ 0.70	61.0	69	84.3%
Flash-VStream-7B	48.8	15	0%
+ StreamPrune @ 0.50	47.1	34	73.3%
+ StreamPrune @ 0.70	41.2	59	85.1%

3.1 Main Results

We show that token pruning is achievable in video streams without substantial accuracy loss: applied to our base model, StreamPrune improves streaming throughput by $6.18\times$ while reducing tokens by

$92.6 \pm 1.8\%$. These gains do not come from exploiting benchmark shortcuts. Many video benchmarks are answerable from commonsense or textual cues alone (Buch et al., 2022; Ko et al., 2023), and on such a benchmark pruning would cost nothing regardless of which tokens were removed. We therefore evaluate on benchmarks that require visual grounding. Table 3(a) reports blind (video-free) accuracy alongside the unpruned baseline; the large gap between the two confirms that video content is required. Together with StreamPrune's accuracy comparable to that of the baseline, this indicates that StreamPrune preserves informative visual tokens while discarding primarily redundant ones.

We further show that StreamPrune transfers to other streaming video MLLMs and preserves accuracy after token pruning. StreamPrune is a plug-and-play module: it requires no retraining and no changes to model weights, and acts only on how the KV cache is constructed. Table 1(b) shows that it generalizes across two further streaming MLLMs built on distinct backbones, LongVA-7B (Zhang et al., 2025b) and Flash-VStream (Zhang et al., 2025a). Finally, Table 2 compares StreamPrune against training-free compression baselines applied to the same backbone. On StreamingBench, StreamPrune outperforms the strongest competing configuration, STC-Cacher & Prune, by 5.6 points; on StreamBench it reaches 63.8, matching StreamChat-Base at $2.65\times$ its encoding throughput. Per-task breakdowns for these benchmarks are given in Appendix A.1 and Appendix A.2, and a threshold sweep on a smaller backbone in Appendix A.3.

3.2 Efficiency Analysis

We measure efficiency along two axes: KV-cache memory growth and encoding throughput. Without pruning, every token of every frame is prefilled into the language model, so the KV cache grows approximately linearly with video duration. StreamPrune prefills only the tokens carrying novel content, and offloads KV states progressively to the CPU. Cache growth is then governed by the rate at which new content enters the stream rather than by elapsed time. Fewer tokens per frame also means more frames encoded within a fixed budget, which raises encoding throughput. Both effects are reported in Table 1(a): at $\tau = 0.70$ the cache grows at 3.6 GB/h against 37.7 GB/h unpruned, while throughput rises from 11 to 68 frames/s.

Table 2: Comparison of **StreamPrune (Ours)** against prior baselines on two streaming video benchmarks. **(a)** Accuracy and streaming throughput (**Thr**, frames/s) on **StreamBench**. **(b)** Training-free token compression methods on the **StreamingBench Real-time Task**, all applied to the ReKV-7B backbone; the parenthesized value is the change relative to that backbone. StreamPrune is evaluated at $\tau = 0.60$ in (a) and $\tau = 0.50$ in (b).

(a) StreamBench			(b) StreamingBench	
Method	**Thr**	**Acc.**	**Method**	**Acc**
ReKV-7B (Unpruned)	11	65.8	ReKV-7B	69.1
VideoLLM-online (Chen et al., 2024a)	5	56.4	+ ToMe (Bolya et al., 2023)	59.4
Flash-VStream (Zhang et al., 2025a)	1	52.1	+ VisionZip (Yang et al., 2025)	60.4
StreamChat-Slow (Xiong et al., 2025)	15	64.7	+ VidCom^2 (Liu et al., 2025)	63.6
StreamChat-Base	20	63.8	+ STC-Pruner (Wang et al., 2026)	63.7
StreamChat-Fast	32	61.7	+ STC-Cacher & Prune (Wang et al., 2026)	65.2
StreamPrune (Ours)	**53**	**63.8**	**StreamPrune (Ours)**	**70.8** (+1.7)

3.3 Ablation

Comparison at matched throughput. Table 3(b) places StreamPrune against the two StreamChat operating points at comparable encoding throughput. Against the slow baseline, StreamPrune at $\tau = 0.25$ reaches 65.6 while encoding at 19 frames/s. Against the fast baseline the margin is wider: at $\tau = 0.50$ StreamPrune sustains 37 frames/s at 65.1, exceeding StreamChat-Fast in accuracy by 3.4 points while encoding more frames per second.

What each token is compared against. Section 2.1 anchors every keep-or-drop decision to the last token retained at a spatial position. Table 4(a) varies only that anchor: `prev` compares against the same position in the preceding frame, the rule used by RLT; `frame` keeps or drops whole frames

Table 3: **(a)** Video ablation: accuracy on each benchmark with and without the video. The **ReKV-7B** column reports the unpruned ReKV-7B backbone with the video present, and **Blind** the same model with the video withheld; both are accuracies (%). **(b)** StreamPrune against the StreamChat baselines on StreamBench at comparable encoding throughput. **Thr**: throughput (frames/s); **Acc**: accuracy (%). The shaded row in each block is the StreamPrune operating point matched to that baseline.

(a) Video Ablation

Dataset	ReKV-7B	Blind
OVO-Bench (Real-Time)	63.0	32.0
OVO-Bench (Backward)	45.7	38.5
StreamingBench	70.3	50.6
ODVBench	51.5	46.2
OVBench	44.8	36.7
StreamBench	65.8	32.8
RVS-Ego	59.5	36.3
RVS-Movie	47.6	38.1

(b) Accuracy on StreamBench

Method	τ	Thr	Acc
Slow Baseline			
StreamChat-Slow	–	15	64.7
StreamPrune	0.25	19	65.6
Fast Baseline			
StreamChat-Fast	–	32	61.7
StreamPrune	0.25	19	65.6
	0.50	37	65.1
	0.60	53	63.8

by majority vote; `ref` is ours. At $\tau = 0.25$ the anchor barely matters: all three rules fall within half a point of one another. `frame` collapses once pruning becomes substantial, losing 22.6 points by $\tau = 0.60$: the events these benchmarks probe are spatially localized, so the changed tokens are a minority, never carry the vote, and are discarded with the static background around them. `prev` stays marginally ahead through $\tau = 0.50$, where runs of dropped tokens are short and the two anchors rarely diverge, but `ref` overtakes at $\tau = 0.60$ and leads at every higher threshold by 1.0 to 5.3 points, holding 52.4 at $\tau = 0.90$ against 48.1 and 39.6. The ordering follows from what each rule guarantees: `prev` constrains only the distance between consecutive frames, and those per-step constraints do not compose into a bound on the distance between a discarded token and the reference token held in the buffer, so a slowly drifting region can stay below the threshold at every step while diverging from what the model stores. Anchoring to the last kept token guarantees that bound in embedding cosine distance directly, which is why the gap widens with τ, as longer runs of dropped tokens are exactly where the two rules come apart.

How the surviving tokens are cached. Table 4(b) isolates the cache layout. `TTS` prunes but pads each frame's survivors back to a full block, preserving the one-frame-per-block layout of ReKV; `TTS+CAC` packs those survivors across frame boundaries (Section 2.2). Both variants encode exactly the same tokens, so the difference is attributable to the layout alone: because the retrieval budget is fixed in blocks rather than in tokens, a padded block spends part of its capacity on filler and returns less distinct evidence. Neither module is sufficient on its own. Pruning alone reaches 63.6 at $\tau = 0.50$, 1.1 points below StreamChat-Slow and 0.2 below StreamChat-Base. Packing the same survivors adds 1.5 points, lifting StreamPrune to 65.1 past both baselines at more than twice the throughput of StreamChat-Slow. TTS supplies the sparsity and CAC converts it into usable retrieval capacity.

Table 4: **Ablations on StreamBench. (a)** Accuracy across comparison strategies (`frame`, `prev`, and our proposed `ref`) under varying pruning thresholds τ. **(b)** Contribution of each module at $\tau = 0.50$, against the two StreamChat baselines; under `TTS` alone the surviving tokens are padded back to a full per-frame block. Our configuration is shaded in each block.

(a) Comparison Strategies

Method	τ	frame	prev	ref (Ours)
StreamPrune	0.25	65.7	66.1	65.6
	0.50	59.5	65.7	65.1
	0.60	43.1	62.4	63.8
	0.70	41.6	59.5	60.5
	0.80	39.7	52.9	58.2
	0.90	39.6	48.1	52.4

(b) Module Ablation

Method	τ	Acc
StreamChat-Slow	–	64.7
StreamChat-Base	–	63.8
StreamPrune (TTS only)	0.50	63.6
StreamPrune (TTS+CAC)	0.50	65.1

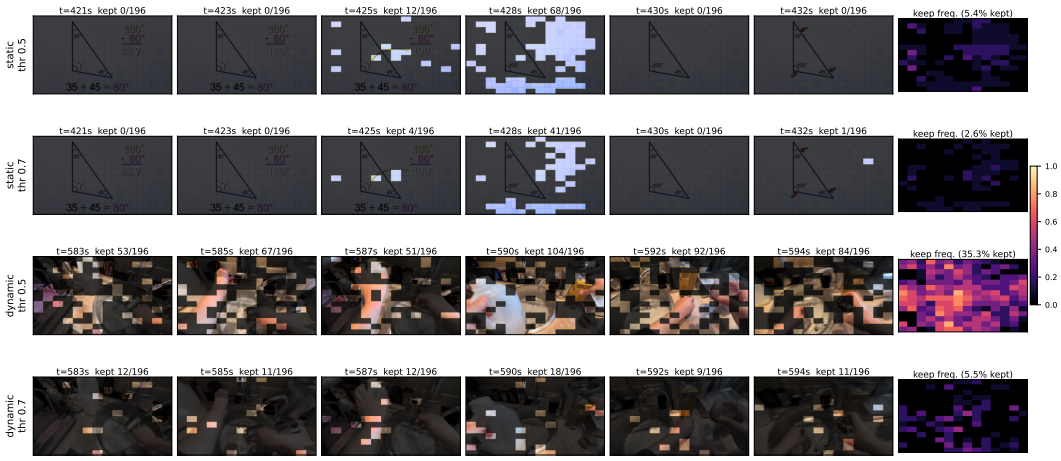

Figure 4: **Where TTS keeps and drops visual tokens, in a static scene and a dynamic one.** Each panel shows a video frame with a visual-token grid laid over it. *Dimmed* cells are tokens TTS dropped because they add nothing new: their content is already in the cache from an earlier frame. *Full-color* cells, outlined in yellow, are tokens passed to the LLM. The rightmost column summarizes the whole clip as a heatmap of how often each position was kept; bright means frequently kept.

4 Qualitative Analysis

Figure 4 shows which tokens TTS retains across two clips and two redundancy thresholds. Row 1 (static scene, moderate threshold): a lecture video with a fixed slide. While the slide stays unchanged, TTS drops almost every token, since re-encoding the same background adds no information. When the lecturer erases or writes on the slide, the kept tokens appear exactly on the edited text, and the untouched diagram and background stay pruned. Row 2 (static scene, higher threshold): the same clip under a stricter redundancy test. The pattern is the same, but even sparser: TTS keeps only the most strongly changed part of each edit. Row 3 (dynamic scene, moderate threshold): an egocentric cooking video, with the camera and hands both moving. Most of the frame changes from second to second, so TTS keeps many more tokens, concentrated on the hands, the objects being handled, and the edges where new content enters the view. Its heatmap is dense, whereas the static clip's is nearly empty. Row 4 (dynamic scene, higher threshold): under the stricter test, TTS keeps only a scattered subset of the moving region and treats small changes as redundant as well. This is the high-compression end of the accuracy–efficiency trade-off. Taken together, the rows show that TTS does not spend a fixed number of tokens per frame. It spends them where the video changes: very few on a static scene, many on a dynamic one. The threshold sets how large a change must be before it is worth a new token.

5 Related Work

Token Reduction/Pruning in MLLMs: Visual inputs generate far more tokens than text, so reducing them is essential for tractable MLLM inference. Spatial methods – ToMe (Bolya et al., 2023), VisionZip (Yang et al., 2025), LLaVA-PruMerge (Shang et al., 2025), and FastV (Chen et al., 2024b) – reduce the token count within a single frame, by merging similar patches in the vision transformer or by ranking tokens under `[CLS]` or query attention. They score each frame independently of those already encoded and therefore leave temporal redundancy unexploited: an unchanging scene incurs the full per-frame cost at every timestep. Methods that do reduce tokens across frames forgo one of the two properties streaming requires. Chat-UniVi (Jin et al., 2024) and TESTA (Ren et al., 2023) cluster along the temporal axis but need the complete token set of a segment, while MovieChat+ (Song et al., 2025) and PruneVid (Huang et al., 2025a) require the user's query. Neither the completed segment nor the query is available when each frame must be pruned on arrival.

Streaming MLLMs: Streaming MLLMs process continuously arriving video frames causally without access to global future context. Chen et al. (2024a) pioneered VideoLLM-online, which introduced streaming dialogue through continuous pre-computation, while Flash-VStream (Zhang et al., 2025a) and Dispider (Qian et al., 2025) decoupled perception from response generation using multiprocessing frameworks. To manage long-horizon streams, memory-augmented models introduce selective Key-Value (KV) cache compression and retrieval. ReKV (Di et al., 2025) offloads uncompressed KVs to CPU memory, while StreamForest (Zeng et al., 2026) compresses KVs at segment or event-tree boundaries. StreamMem (Yang et al., 2026) employs query-agnostic proxy templates, but most other retrieval-based approaches still require the user's query to make pruning or retrieval decisions. Closest to our setting, STC (Wang et al., 2026) compresses streaming video LLMs hierarchically, combining token pruning with a compressed KV cache; StreamPrune differs in what the keep decision is anchored to, comparing each token against the last token retained at its own spatial position so that the cached content itself bounds what may be discarded.

6 Conclusion

We presented **StreamPrune**, a training-free, plug-and-play token pruning module for streaming video MLLMs that removes visual redundancy causally, using only inter-frame novelty and without access to future frames or the downstream query. By retaining temporally novel tokens in a content-adaptive cache, it allocates computation according to the dynamics of the stream, pruning heavily in static scenes while preserving detail in dynamic ones under a single configuration. Across six streaming video benchmarks it reduces visual tokens by $92.6 \pm 1.8\%$ and improves throughput by $6.18\times$ at an average accuracy cost of 0.8 ± 2.9 points (mean $\pm$ standard deviation across the six benchmarks), and generalizes across multiple backbones.

7 Limitations

StreamPrune measures novelty in the vision encoder's embedding space, at the granularity of a fixed spatial grid, which bounds what it can exploit. A change that displaces the whole grid, such as a camera pan or a global illumination shift, renders every token novel, so little is pruned even when the underlying scene is unchanged; conversely, a change confined to a small object may fall below τ at every position it occupies. The threshold τ is also a single global constant, fixed before the stream begins rather than adapted to it, so the accuracy–throughput operating point must be chosen in advance: Table 1(a) shows the cost of choosing it too aggressively, with average accuracy falling 4.1 points at $\tau = 0.90$. Finally, pruning is irreversible and query-free by construction. Content discarded during encoding cannot be recovered when a question arrives, so a query about detail the pruner judged redundant cannot be answered from the cache, however the query is phrased.

AI USE STATEMENT

In this work, we used generative AI tools for three purposes. First, we used them to aid and polish writing: improving the clarity, grammar, and phrasing of text written by the authors. Second, we used them for retrieval and discovery to search for and surface potentially relevant prior work. Third, we used them to produce initial drafts of individual sections of the paper, which the authors then revised. Every reference surfaced by these tools was read and verified by the authors before being cited, and every claim we attribute to prior work was checked against the original paper.

ETHICS STATEMENT

This work introduces a training-free inference-time module and collects no new data. All six evaluation benchmarks are publicly released datasets, used under their original licenses and for their intended purpose of evaluating video understanding; no human subjects were involved and no personally identifying information was collected or annotated. Because StreamPrune neither retrains nor alters model weights, it inherits the biases, hallucinations, and failure modes of the backbones it accelerates and does not introduce a new source of them, but it also does not mitigate them. Two properties do warrant care in deployment. First, pruning is irreversible: content judged temporally redundant during encoding is unavailable to every later query, so in safety-critical streaming settings such as driving or monitoring, the threshold should be validated on the target distribution rather than transferred from our benchmarks. Second, by lowering the cost of dense streaming perception, the method makes continuous video analysis cheaper in general, including for surveillance applications where the same efficiency gains may raise privacy concerns; we note this dual-use potential explicitly. On the other side, the reduction in prefill compute and cache memory we report lowers the energy required per hour of video processed.

REPRODUCIBILITY STATEMENT

The method is specified in full in Sections 2.1 and 2.2: the keep rule, the reference-update rule, and the block-packing scheme are stated explicitly, and the pruning threshold τ is the only hyperparameter, with a complete sweep reported in Table 1. StreamPrune is training-free and introduces no learned parameters, so reproducing it requires no training run. The base model, frame sampling rate, per-frame token budget, block size, and the hardware on which all throughput figures were measured are given in the Implementation Details paragraph of Section 3; all six evaluation benchmarks are public and are cited in the Benchmarks paragraph of the same section.

REFERENCES

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *International Conference on Learning Representations (ICLR)*, 2023.

Shyamal Buch, Cristóbal Eyzaguirre, Adrien Gaidon, Jiajun Wu, Li Fei-Fei, and Juan Carlos Niebles. Revisiting the "video" in video-language understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2917–2927, 2022.

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.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pp. 19–35. Springer, 2024b.

Rohan Choudhury, Guanglei Zhu, Sihan Liu, Koichiro Niinuma, Kris M Kitani, and László A Jeni. Don't look twice: Faster video transformers with run-length tokenization. *Advances in Neural Information Processing Systems*, 37:28127–28149, 2024.

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.

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, 2025a.

Zhenpeng Huang, Xinhao Li, Jiaqi Li, Jing Wang, Xiangyu Zeng, Cheng Liang, Tao Wu, Xi Chen, Liang Li, and Limin Wang. Online video understanding: Ovbench and videochat-online. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3328–3338. IEEE, 2025b.

Peng Jin, Ryuichi Takanobu, Wancai Zhang, Xiaochun Cao, and Li Yuan. Chat-univi: Unified visual representation empowers large language models with image and video understanding. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13700–13710. IEEE, 2024.

Dohwan Ko, Ji Lee, Woo-Young Kang, Byungseok Roh, and Hyunwoo J. Kim. Large language models are temporal and causal reasoners for video question answering. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4300–4316, 2023.

Yixuan Li, Changli Tang, Jimin Zhuang, Yudong Yang, Guangzhi Sun, Wei Li, Zejun Ma, and Chao Zhang. Improving LLM video understanding with 16 frames per second. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 35942–35956. PMLR, 2025.

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.

Xuyang Liu, Yiyu Wang, Junpeng Ma, and Linfeng Zhang. Video compression commander: Plug-and-play inference acceleration for video large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1910–1924, 2025.

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.

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.

Shuhuai Ren, Sishuo Chen, Shicheng Li, Xu Sun, and Lu Hou. Testa: Temporal-spatial token aggregation for long-form video-language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 932–947, 2023.

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.

Enxin Song, Wenhao Chai, Tian Ye, Jenq-Neng Hwang, Xi Li, and Gaoang Wang. Moviechat+: Question-aware sparse memory for long video question answering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18523–18533, 2026.

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. In *International Conference on Learning Representations (ICLR)*, 2025.

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. Visionzip: Longer is better but not necessary in vision language models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19792–19802. IEEE, 2025.

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. In *CVPR 2026 Workshop on Video Large Language Models (VidLLMs)*, 2026.

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.

Haoji Zhang, Yiqin Wang, Yansong Tang, Yong Liu, Jiashi Feng, and Xiaojie Jin. Flash-vstream: Efficient real-time understanding for long video streams. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 21059–21069. IEEE, 2025a.

Peiyuan Zhang, Kaichen Zhang, Bo Li, Guangtao Zeng, Jingkang Yang, Yuanhan Zhang, Ziyue Wang, Haoran Tan, Chunyuan Li, and Ziwei Liu. Long context transfer from language to vision. *Transactions on Machine Learning Research (TMLR)*, 2025b.

A APPENDIX

A.1 PER-TASK RESULTS ON STREAMBENCH AND STREAMINGBENCH

Table 2 reports a single averaged score per method. Table 5 and Table 6 give the per-task accuracies behind those averages, together with the encoding throughput on StreamBench. Both tables use the ReKV-7B backbone at 1 FPS; StreamPrune is evaluated at $\tau = 0.60$ on StreamBench and $\tau = 0.50$ on StreamingBench. Column headings are expanded in each caption.

Table 5: Per-task performance on **StreamBench**. **THR** is the encoding throughput of the model (frames/s) and **Avg** is the benchmark score. The first row is the unpruned ReKV-7B backbone; StreamPrune is evaluated at $\tau = 0.60$, the operating point reported in Table 2(a). Acronyms: **OS**: Object Search, **LM**: Long-term Memory Search, **SM**: Short-term Memory Search, **CI**: Conversational Interaction, **KG**: Knowledge-based Question Answering, **SF**: Simple Factual.

Method	THR	OS	LM	SM	CI	KG	SF	Avg
ReKV-7B (Unpruned)	11	50.9	62.1	59.6	75.4	73.1	74.8	65.8
VideoLLM-online	5	41.4	48.8	52.9	62.7	69.2	64.1	56.4
Flash-VStream	1	37.1	44.5	48.6	58.1	66.4	59.2	52.1
StreamChat-Slow	15	51.7	53.9	57.8	68.5	88.1	69.3	64.7
StreamChat-Base	20	50.5	52.9	56.1	67.6	87.9	68.3	63.8
StreamChat-Fast	32	48.1	49.5	53.5	65.2	86.7	67.6	61.7
StreamPrune (Ours)	**53**	**48.5**	**61.4**	**55.7**	**71.0**	**78.9**	**68.2**	**63.8**

Table 6: Per-task accuracy of token compression methods on the **StreamingBench Real-time Task**. All methods are applied on top of the ReKV-7B baseline, and StreamPrune is evaluated at $\tau = 0.50$, the operating point reported in Table 2(b). Values beneath the StreamPrune row indicate the change relative to the unpruned ReKV-7B baseline. Acronyms: **CS**: Clip Summarization, **OP**: Object Perception, **ATP**: Attribute Perception, **PR**: Prospective Reasoning, **ACP**: Action Perception, **SU**: Spatial Understanding, **EU**: Event Understanding, **CT**: Counting, **TR**: Text-Rich Understanding, **CR**: Causal Reasoning.

	StreamingBench Real-time Task										
Method	**CS**	**OP**	**ATP**	**PR**	**ACP**	**SU**	**EU**	**CT**	**TR**	**CR**	**Avg.**
ReKV-7B	79.2	77.5	75.6	66.0	62.2	60.3	72.3	43.6	69.7	79.5	**69.1**
+ ToMe (Bolya et al., 2023)	67.5	64.6	66.3	63.0	58.4	53.3	65.2	19.7	57.3	76.6	**59.4**
+ VisionZip (Yang et al., 2025)	69.5	66.4	69.3	50.7	52.9	57.2	64.1	33.3	54.8	73.4	**60.4**
+ VidCom^2 (Liu et al., 2025)	76.0	68.1	71.6	62.0	58.6	52.0	64.0	42.5	60.4	76.6	**63.6**
+ STC-Pruner (Wang et al., 2026)	75.4	66.8	71.2	63.9	57.8	51.2	64.0	45.1	63.2	76.6	**63.7**
+ STC-Cacher & Prune (Wang et al., 2026)	74.4	71.3	72.9	69.4	58.9	52.4	60.9	44.0	66.9	77.3	**65.2**
StreamPrune (Ours)	74.8 (-4.4)	77.9 (+0.4)	79.3 (+3.7)	68.5 (+2.5)	64.2 (+2.0)	59.8 (-0.5)	70.7 (-1.6)	50.0 (+6.4)	70.7 (+1.0)	82.0 (+2.5)	**70.8** (+1.7)

A.2 PER-TASK RESULTS ON OVO-BENCH

Table 7 and Table 8 report the two OVO-Bench splits, real-time visual perception and backward tracing, by sub-task. Both use the same training-free compression baselines as Table 2(b), each applied to the ReKV-7B backbone at 1 FPS. The **Avg.** column is the benchmark score for that split.

A.3 RESULTS ON A SMALLER BACKBONE

The main paper evaluates 7B backbones throughout. Table 9 repeats the threshold sweep of Table 1(a) on a 0.5B LLaVA-OneVision backbone, reporting accuracy on each of the six benchmarks at every value of τ. The first row is the unpruned backbone.

Table 7: Per-task performance on **OVO-Bench (Real-Time Visual Perception)**. All methods are applied on top of the ReKV-7B backbone, whose unpruned scores are given in the first row. **Avg.** is the benchmark score for this split. Sub-tasks: **OCR** (Optical Character Recognition), **ACR** (Action Recognition), **ATR** (Attribute Recognition), **STU** (Spatial Understanding), **FPD** (Future Prediction), and **OJR** (Object Recognition).

Method	OCR	ACR	ATR	STU	FPD	OJR	Avg.
ReKV-7B	73.8	56.0	74.1	51.7	70.3	60.3	**63.0**
+ ToMe (Bolya et al., 2023)	61.1	49.5	52.6	42.7	62.4	50.0	**53.1**
+ VisionZip (Yang et al., 2025)	49.0	49.5	64.7	44.4	64.4	50.5	**53.8**
+ VidCom^2 (Liu et al., 2025)	65.8	59.6	69.0	47.2	64.4	56.5	**60.4**
+ STC-Pruner (Wang et al., 2026)	64.4	59.6	68.1	48.9	65.4	56.5	**60.5**
+ STC-Cacher & Prune (Wang et al., 2026)	68.5	57.8	73.3	47.2	68.3	59.8	**62.5**
StreamPrune (Ours)	67.1	56.0	74.1	42.1	68.3	62.0	**61.6**

Table 8: Per-task performance on **OVO-Bench (Backward Tracing)**. All methods are applied on top of the ReKV-7B backbone, whose unpruned scores are given in the first row. **Avg.** is the benchmark score for this split. Sub-tasks: **EPM** (Episodic Memory), **ASI** (Action Sequence Identification), and **HLD** (Hallucination Detection).

Method	EPM	ASI	HLD	Avg.
ReKV-7B	54.2	57.4	28.5	**45.7**
+ ToMe (Bolya et al., 2023)	49.2	50.0	29.6	**42.9**
+ VisionZip (Yang et al., 2025)	47.1	54.1	30.7	**44.0**
+ VidCom^2 (Liu et al., 2025)	50.5	53.4	32.8	**45.6**
+ STC-Pruner (Wang et al., 2026)	51.2	52.0	33.3	**45.5**
+ STC-Cacher & Prune (Wang et al., 2026)	50.5	55.4	30.1	**45.3**
StreamPrune (Ours)	58.2	58.8	18.3	**45.4**

Table 9: Accuracy across all six benchmarks for StreamPrune applied to the 0.5B LLaVA-OneVision backbone, at each value of the pruning threshold (τ). A dash marks a configuration we did not evaluate. Acronyms: **OVO-RT**: OVO-Bench (Real-Time), **OVO-BW**: OVO-Bench (Backward Tracing), **SB**: StreamingBench, **ODVB**: ODVBench, **OVB**: OVBench, **StB**: StreamBench, **RVS-E**: RVS-Ego, **RVS-M**: RVS-Movie.

Model	τ	OVO-RT	OVO-BW	SB	ODVB	OVB	StB	RVS-E	RVS-M
LLaVA-OV-0.5B	–	47.4	33.9	56.2	36.6	38.2	53.4	52.3	42.5
StreamPrune	0.25	50.2	33.0	58.0	36.1	38.5	53.8	54.1	42.1
	0.50	50.5	32.3	58.1	36.0	38.6	53.6	54.4	42.6
	0.60	48.7	31.7	56.2	35.9	38.8	50.0	54.3	41.6
	0.70	46.3	28.9	55.0	35.2	39.5	46.2	55.4	41.4
	0.80	41.4	29.8	51.0	34.6	38.8	41.1	52.6	40.1
	0.90	36.8	29.7	44.8	34.2	37.5	37.1	–	–