AUDIOROUTER: AUDIO-GUIDED VISUAL COMPRESSION FOR STREAMING VIDEO LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Streaming video language models require efficient visual compression to preserve task-relevant evidence under causal constraints, where future observations are unavailable. Existing approaches mainly rely on visual signals for token selection, which can miss event-related information under aggressive compression. Meanwhile, audio-visual models typically inject audio representations into the LLM, requiring audio-capable architectures. We propose AudioRouter, an audio-guided visual compression framework based on a simple principle: *audio determines where to look, while vision determines what the LLM sees*. Instead of introducing audio tokens into the LLM, AudioRouter uses synchronized audio to condition learnable queries and generate a routing distribution over native visual tokens. The compressed representation is formulated as $Z_t = P_t V_t$, where audio controls the routing weights while the aggregated values remain native visual values.. This preserves the standard visual-token interface and enables causal streaming inference. Experiments on long-video and streaming benchmarks show that AudioRouter maintains performance close to the full 196-token representation with only 64 tokens per frame, retaining about one-third of the visual tokens while substantially reducing LLM inference cost. https://anonymous.4open.science/r/AudioRouter-1F96.

1 INTRODUCTION

Video language models have shown strong performance in long-video understanding, online question answering, and interactive agents Song et al. (2024); Chen et al. (2024); Di et al. (2025a); Qian et al. (2025). However, directly deploying existing video language models in streaming scenarios introduces a fundamental constraint: visual content arrives continuously, and the model must process past observations before future content becomes available. Unlike offline video understanding, a streaming model at time t can access only the current and previously observed video content. It cannot revisit the current frame after seeing future events to reassess which visual information matters. Meanwhile, each incoming frame produces a large number of visual tokens. Retaining all of them causes the input sequence, LLM prefill computation, and memory usage to grow continuously with video length Qian et al. (2024); Wang et al. (2026). Under a streaming setting with no access to future segments, visual compression is therefore more than a general efficiency problem. It becomes a continual problem of causal visual evidence selection: the model must decide which visual evidence to retain using only information available at the current time Chen et al. (2026).

Existing streaming visual compression methods mainly estimate token importance from the visual stream itself, using signals such as visual saliency, cross-frame redundancy, token similarity, or historical memory. These methods can effectively remove visual redundancy, but streaming inference without future observations creates an inherent limitation. When future events and future questions are unavailable, current visual features alone provide limited information about which parts of a frame are relevant to the event that is unfolding. A single frame may contain many objects, people, and background regions, while the evidence related to the current event may occupy only a small portion of the frame. This makes aggressive compression particularly difficult because visual evidence discarded at the current time cannot be recovered later. This limitation raises a simple question: **can audio help decide which visual tokens to keep without being passed to the LLM?**

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.

Synchronized audio provides a natural cue for this decision. Sounds in a video are often closely tied to the visual events happening at the same time Tian et al. (2018). Speech can indicate a likely speaker, an impact sound can indicate where objects interact, and an engine or horn can indicate vehicle-related regions. We therefore use audio to guide visual compression: audio changes how visual tokens are weighted, while the visual tokens provide the values passed to the LLM.

Based on this idea, we propose AudioRouter, an audio-guided visual compression for streaming video language models. For each incoming video segment, we encode the synchronized visual and audio inputs separately. The audio representation, together with temporal context, modulates a set of learnable queries. These audio-conditioned queries interact with the current visual tokens and define a conditional routing distribution that compresses the dense visual representation into a fixed number of compact visual tokens. Unlike conventional audio-visual fusion, audio affects the selection and aggregation weights but is never used as a value in the compressed representation. Formally, $Z_t = P(A_t, V_t)V_t$, where A_t affects the routing matrix P, while the values aggregated into Z_t always come from the native visual representation V_t. The LLM therefore receives only the resulting visual tokens, with no explicit audio token or audio feature vector added to its input.

This design has two important properties. First, AudioRouter supports segment-wise online inference: the representation for the current segment is computed without accessing any future video or audio segment. Second, audio is used only to control visual routing and is never passed to the LLM as an explicit input token or feature vector. Our method therefore does not require the language model to support audio input or rely on a dedicated omni-modal language model. The LLM receives the same visual-token interface as a standard visual language model (VLM). In principle, the same audio-guided compression module can therefore be used with different visual-only language models.

The benefit of AudioRouter is most evident under a limited visual-token budget. With only 64 tokens per segment, synchronized audio helps preserve task-relevant visual evidence that vision-only compression may discard. As a result, AudioRouter maintains performance close to the full 196-token representation while substantially reducing LLM prefill computation and memory usage, achieving the best overall performance under causal streaming evaluation. These results demonstrate that audio can improve visual compression by guiding where to retain visual evidence without being explicitly passed to the LLM.

Our main contributions are summarized as follows:

- We introduce AudioRouter: an audio-guided visual compression framework for streaming video understanding, where synchronized audio guides visual evidence selection without being passed to the language model as a separate input stream. This design allows a streaming VLM to use audio cues while preserving the standard visual-token interface of a visual-only language model.
- We propose an Audio-Conditioned Query Bottleneck and an Audio-Guided Where-to-Look mechanism. The former enables event-adaptive query generation from audio and temporal context, while the latter performs conditional visual routing to preserve task-relevant evidence under a constrained token budget.
- We systematically evaluate AudioRouter on long-video and causal streaming video benchmarks. With only 64 visual tokens per segment, AudioRouter maintains understanding performance close to the full 196-token visual representation while substantially reducing LLM prefill computation and memory usage. Under the no-future-segment streaming protocol, it further achieves the best overall performance among the compared methods.

2 Related Work

2.1 Streaming Video Understanding

Streaming video understanding requires a model to process continuously arriving frames and answer questions without access to future content. Existing methods mainly follow two directions. End-to-end streaming models, including VideoLLM-Online, Flash-VStream, and Dispider, introduce online training or streaming-specific modules for continuous perception and response generation Chen et al. (2024); Zhang et al. (2025); Qian et al. (2025). Memory-based methods, such as MovieChat, ReKV,

LiveVLM, and StreamMem, maintain compact historical states through token merging, KV-cache reuse, retrieval, or fixed-size memory Song et al. (2024); Di et al. (2025a); Ning et al. (2025); Yang et al. (2025). LongVA and LLaVA-OneVision instead support long videos through larger context windows or dense visual inputs Zhang et al. (2024a); Li et al. (2024).

These methods improve temporal coverage and online response, but most of them either retain dense frame representations or decide what to keep using visual features, user queries, or model attention. Our method compresses each incoming segment before LLM prefill and uses synchronized audio to guide visual evidence selection. This design reduces the visual sequence entering the LLM without requiring a streaming-specific or audio-capable language model.

2.2 Visual Token Compression for VideoLLMs

Visual-token compression reduces VideoLLM cost by merging, pruning, or reallocating visual tokens. ToMe merges similar tokens during visual encoding, while SparseVLM, DyCoke, and VidCom2 select or compress tokens according to text relevance, temporal redundancy, or frame-level visual information Bolya & Hoffman (2023); Zhang et al. (2024b); Tao et al. (2025); Liu et al. (2025). These methods are effective in offline settings, but some require full-video context or a known user query, which may be unavailable in online streaming inference without access to future segments.

Streaming methods remove this look-ahead assumption. TimeChat-Online removes tokens according to adjacent-frame changes, StreamingTOM combines causal temporal reduction with online memory, and STC uses feature reuse and spatiotemporal token pruning to reduce both visual encoding and LLM prefill costs Yao et al. (2025); Chen et al. (2026); Wang et al. (2026). ReKV, LiveVLM, and StreamMem further reduce the cost of maintaining long visual histories through KV-cache or memory compression Di et al. (2025a); Ning et al. (2025); Yang et al. (2025).

The main limitation is that these compression decisions are largely based on visual redundancy or model-internal statistics. Under a small token budget, visually salient or temporally novel regions may still be unrelated to the event currently occurring. Our method adds synchronized audio as an extra selection cue, allowing the visual bottleneck to retain event-related evidence that vision-only compression may discard. This difference is most relevant under aggressive compression, where an incorrect causal decision cannot be corrected after future frames arrive.

2.3 Audio-Visual Learning and Language Models

Prior audio-visual learning methods use temporal synchronization to learn cross-modal representations, localize sounding objects, or select informative video segments Chen et al. (2022); Gao et al. (2020). More recent audio-visual language model, including Audio-Visual LLM, VideoLLaMA 2, Video-SALMONN 2, and Qwen2.5-Omni, feed audio and visual representations into a multimodal language model for joint reasoning Shu et al. (2025); Cheng et al. (2024); Tang et al. (2025); Xu et al. (2025).

Our goal differs from direct audio-visual reasoning. Instead of feeding audio representations into the language model, we use synchronized audio to guide which visual evidence is retained by the compressor. The language model continues to receive the same visual-token interface as a standard visual-only VLM. This separation allows synchronized audio to guide visual compression without requiring an audio-capable language model.

3 Methodology

3.1 Overall Architecture

We study efficient visual representation for streaming video understanding. Unlike existing audio-visual LLMs that feed both audio and visual representations into the language model, we use synchronized audio to guide visual compression without passing a separate audio token or audio feature stream to the LLM. The key idea is simple: audio determines where to look, while vision determines what the LLM sees.

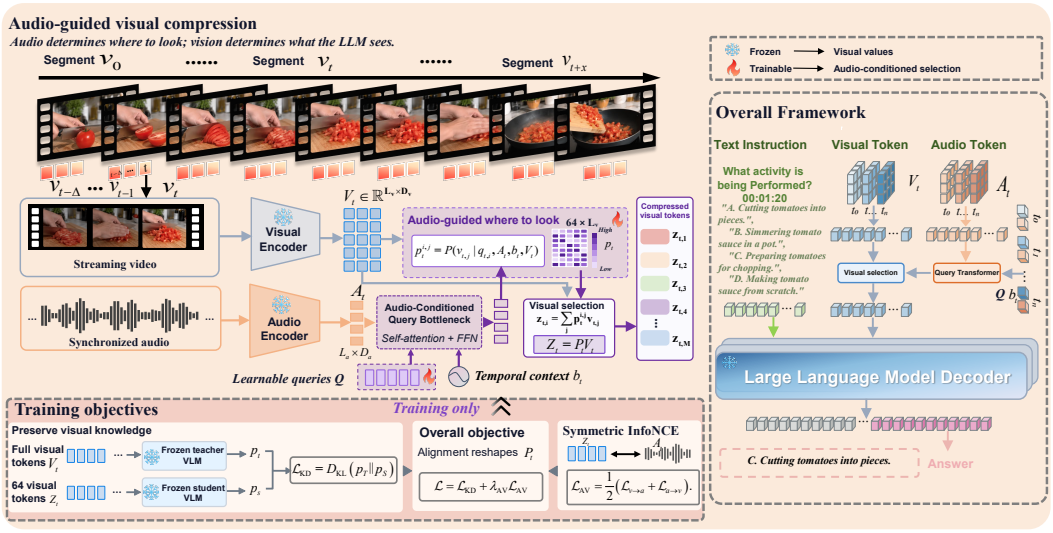


Figure 1: Overview of Architecture. **(a)** For each currently available video segment, synchronized audio and temporal context condition learnable queries, which parameterize a routing distribution P_t over native visual tokens V_t. The resulting representation $Z_t = P_t V_t$ contains M compressed visual tokens: Audio guides the weights; vision provides the values. **(b)** During training, knowledge distillation preserves the predictions of the full-token teacher, and a symmetric audio–visual InfoNCE objective aligns the compressed visual representation with synchronized audio using time-shifted and cross-video negatives. Only the query bottleneck is optimized under $\mathcal{L} = \mathcal{L}_{\mathrm{KD}} + \lambda_{\mathrm{AV}}\mathcal{L}_{\mathrm{AV}}$.

We divide a continuous video stream into segments and use t to index the current segment. A frozen vision backbone and an audio encoder independently produce the native visual tokens V_t and audio representations A_t:

$$V_t = \mathrm{VisualEnc}(v_{t-\Delta:t}), \qquad A_t = \mathrm{AudioEnc}(a_{t-\Delta:t}).$$

where $V_t \in \mathbb{R}^{L_v \times D_v}$, $A_t \in \mathbb{R}^{L_a \times D_a}$. L_v and L_a denote the numbers of visual and audio tokens, respectively. We use $M = 64$ learnable queries to compress the dense visual representation V_t into

$$Z_t \in \mathbb{R}^{M \times D_v}, \qquad M = 64,$$

where M is the visual-token budget retained for each streaming segment and passed to the LLM.

Our framework consists of two main components. The Audio-Conditioned Query Bottleneck uses synchronized audio to adapt the learnable queries to the current event. The Audio-Guided Where-to-Look module then uses these audio-conditioned queries to aggregate event-relevant evidence from the native visual tokens. Only the compressed visual tokens Z_t are passed to the original VLM. Audio remains entirely within the visual compression path.

3.2 Audio-Conditioned Query Bottleneck

Query-based visual resamplers typically use a fixed set of learnable queries to extract compact representations from visual features. In streaming video, however, different segments can contain different events, and fixed queries cannot adapt their visual selection behavior to the event observed at the current time.

Text-conditioned queries usually encode task-level semantic intent, whereas synchronized audio provides local context about the event occurring in the current segment. We therefore use audio to dynamically condition the queries so that their visual selection behavior can adapt over time.

At time t, we first project the audio features into the query hidden space, initialize a set of M learnable queries and condition them on the projected audio features and the current temporal context:

$$\bar{A}_t = A_t W_A \qquad Q_t = \mathcal{B}(Q_0, \bar{A}_t, b_t), \qquad Q_0 \in \mathbb{R}^{M \times d}.$$

where $\mathcal{B}$ denotes a lightweight query generator consisting of cross-attention and feed-forward layers, and b_t represents the temporal context available at time t.

The resulting queries Q_t are not passed to the language model. They are used only to determine how visual tokens are routed in the subsequent compression stage.

3.3 AUDIO-GUIDED WHERE-TO-LOOK

Given the audio-conditioned queries Q_t, our goal is to learn a visual evidence selection distribution under a fixed budget of M output tokens.

We first project the native visual tokens into the routing space and compute the matching score between each query and each visual token:

$$\tilde{V}_t = V_t W_V, \qquad R_t(i,j) = \mathrm{sim}(q_{t,i}, \tilde{v}_{t,j}),$$

where $\mathrm{sim}(\cdot,\cdot)$ denotes cosine similarity. We then normalize the matching scores over the visual-token dimension:

$$P_t = \mathrm{Softmax}\left(\frac{R_t}{\tau}\right), \qquad P_t \in \mathbb{R}^{M \times L_v},$$

where τ is a temperature parameter. Each row of P_t specifies the routing distribution of one audio-conditioned query over the visual tokens in the current segment. **We interpret P_t as the model's Where-to-Look distribution under a limited visual-token budget.**

The compressed representation is obtained by aggregating the native visual tokens according to this routing distribution:

$$Z_t = P_t V_t.$$

Audio therefore controls the routing weights through P_t, while the values aggregated into Z_t are always the native visual tokens V_t. In other words, audio determines where to look, while vision provides the values passed forward by the compressor.

Finally, the compressed visual tokens Z_t are directly fed into the language model. No explicit audio token or audio feature vector is added to the LLM input, so the original visual-token interface is unchanged.

3.4 TRAINING THE AUDIO-CONDITIONED BOTTLENECK

During training, we freeze the visual encoder, the audio encoder, and the original VLM, and optimize only the audio-conditioned compression module. The training objective contains two terms: visual knowledge preservation and audio-visual alignment.

Visual Knowledge Preservation. Aggressive visual compression can remove information that the original VLM requires for downstream reasoning. We therefore use the model operating on the full visual representation as a frozen teacher and encourage the compressed model to preserve its output distribution:

$$\mathcal{L}_{\mathrm{KD}} = D_{\mathrm{KL}}\left(p_T \,\|\, p_S\right),$$

where p_T and p_S denote the output distributions of the teacher and the compressed student, respectively. This objective encourages the compressed visual representation to retain task-relevant information available in the full visual representation.

Audio-Visual Alignment. Knowledge preservation alone does not explicitly require the conditioned queries to use synchronized audio. We therefore add an audio-visual alignment objective that encourages the compressed visual representation to remain associated with the audio event occurring in the same segment.

For the current segment, (Z_t, A_t) forms a positive pair. We use temporally mismatched samples and a feature queue as negative examples and define a bidirectional contrastive objective:

$$\mathcal{L}_{\mathrm{AV}} = \frac{1}{2}\left(\mathcal{L}_{v\to a} + \mathcal{L}_{a\to v}\right).$$

The final training objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{KD}} + \lambda_{\mathrm{AV}} \mathcal{L}_{\mathrm{AV}},$$

where λ_{AV} controls the contribution of the audio-visual alignment term.

4 EXPERIMENTS

We evaluate AudioRouter from three perspectives: long-video understanding, streaming video understanding under a no-future-segment protocol, and inference efficiency. We first compare the proposed method with dense and streaming video language models, then study how audio changes visual routing, and finally analyze the effects of the training objective, visual-token budget, and system-level inference cost.

4.1 EXPERIMENTAL SETUP

Datasets. We evaluate AudioRouter on both long-video understanding and real-time streaming video understanding.

For long-video understanding, we use two widely adopted benchmarks: VideoMME and MLVU. VideoMME covers videos at different temporal scales and reports results on the Short, Medium, and Long subsets. MLVU evaluates multi-task long-video understanding, including event understanding, temporal relations, and reasoning over long contexts. Unless otherwise specified, we follow the evaluation protocols used by existing streaming video understanding methods and evaluate on the official test splits.

We further evaluate streaming performance under the no-future-segment protocol of Streaming-Bench Real-Time Visual Understanding (RTVU), where the model answers questions using only observations available up to the current time. AudioRouter is trained on external video data from InternVid, WebVid, YouCook2, Ego4D/EgoQA, VideoChatGPT, and Kinetics, while VideoMME, MLVU, and RTVU are reserved for evaluation. For each benchmark, we train a separate AudioRouter bottleneck on the external corpus and directly evaluate the resulting checkpoint on the corresponding benchmark. Detailed training data and filtering procedures are provided in Appendix **??**.

Model and Implementation Details. We build AudioRouter on LLaVA-OneVision-Qwen2-7B-OV. During training, we freeze the visual encoder, audio encoder, and language model, and optimize the proposed Audio-Conditioned Query Bottleneck.

The original visual encoder produces 196 visual tokens per sampled frame. Unless otherwise specified, we use 64 learnable queries and compress the visual representation to 64 visual tokens.

Audio features condition only the query and routing process, $A_t \rightarrow Q_t \rightarrow P_t$, and the compressed representation is constructed as $Z_t = P_t V_t$. The language model receives Z_t through the original visual projection, with no separate audio-token or audio-feature input.

For VideoMME, MLVU, and RTVU, we follow the sampling strategy used by existing streaming video understanding methods. We sample videos shorter than 30 minutes at 0.5 FPS and longer videos at 0.2 FPS. All experiments use greedy decoding with the same prompt and inference configuration across compared settings.

Evaluation Metrics. We report accuracy as the primary metric for video understanding. To measure inference efficiency, we additionally report the number of visual tokens passed to the language model, peak GPU memory usage, LLM prefill latency, Time-to-First-Token (TTFT), and end-to-end inference latency when available.

4.2 MAIN RESULTS

Table 1 compares AudioRouter with representative dense, streaming, and efficient video understanding methods. With only 64 retained visual tokens, AudioRouter obtains an average accuracy of 60.19% on VideoMME, 72.40% on RTVU, and 68.4% on MLVU. Among the compared streaming and efficient methods, AudioRouter achieves the highest RTVU average and the highest MLVU score, while remaining competitive on VideoMME. In particular, it reaches 53.01% on the Long subset of VideoMME, where aggressive compression is especially challenging.

Controlled Comparison under a Unified Backbone. Table 2 compares representative streaming and compression methods using the same LLaVA-OV-7B backbone. AudioRouter achieves the

Table 1: Comparison with representative long-video and streaming video understanding methods. The Frames column reports the sampling rate or frames. The last column reports the averaged score over VideoMME, RTVU, and MLVU. Higher benchmark scores are better.

Method	Frames	VideoMME				RTVU											MLVU	Overall
		Short	Medium	Long	AVG	OP	CR	CS	ATP	EU	TR	PR	SU	ACP	CT	AVG		
Dense Video Language Models																		
LLaVA-OV-7B [9]	0.5 FPS	70.1	56.4	48.8	58.4	**80.38**	74.22	76.03	80.72	72.67	71.65	67.59	65.45	65.72	45.08	71.12	64.7	64.74
GPT-4o	–	–	–	–	–	77.11	80.47	83.91	76.47	70.19	**83.80**	66.67	62.19	69.12	49.22	**73.28**	64.6	–
Streaming Video Understanding Methods																		
MovieChat-7B [16]	2048	–	–	33.4	38.2	–	–	–	–	–	–	–	–	–	–	–	25.8	–
VideoLLM-Ol-8B 2	2 FPS	–	–	–	56.3	39.07	40.06	34.49	31.05	45.96	32.40	31.48	34.16	42.49	27.89	35.99	52.6	48.27
Flash-VStream [24]	1 FPS	**72.0**	61.1	50.3	**61.2**	25.89	43.57	24.91	23.87	27.33	13.08	18.52	25.20	23.87	48.70	23.23	66.31	50.24
Dispider-7B [14]	1 FPS	49.7	53.7	49.7	50.2	74.92	75.53	74.10	73.08	74.44	59.92	**76.14**	62.91	62.16	45.80	67.63	61.7	62.18
LongVA-7B [25]	128	–	50.4	46.2	52.6	70.0	63.3	61.2	70.9	62.7	59.5	61.1	53.7	54.7	34.7	60.0	56.3	56.30
ReKV [7]	0.5 FPS	70.4	55.4	47.3	57.7	74.39	78.91	78.55	77.12	68.32	67.91	67.59	62.60	64.31	44.56	70.59	68.2	65.70
LiveVLM-7B [12]	0.5 FPS	66.7	56.4	48.8	57.3	79.84	79.69	**84.86**	80.72	67.08	70.09	74.07	66.26	67.99	42.49	71.31	68.1	65.57
StreamMem [22]	0.5 FPS	71.5	56.6	50.1	59.4	–	–	–	–	–	–	–	–	–	–	–	66.9	–
StreamingTOM [4]	0.5FPS	71.3	**57.8**	50.6	59.9	68.18	**88.89**	80.26	75.00	69.05	65.12	42.86	**72.55**	72.73	42.11	67.40	67.9	65.07
STC [20]	0.5 FPS	67.3	53.9	48.3	56.5	71.3	77.3	74.4	72.9	60.9	66.9	69.4	52.4	58.9	44.0	65.2	67.0	62.90
AudioRouter	**0.5FPS**	68.57	57.14	**53.01**	60.19	74.24	**88.89**	81.58	**85.42**	**76.19**	68.60	66.67	64.71	**77.27**	**52.63**	72.40	**68.4**	**67.00**

Table 2: Controlled comparison on VideoMME using the same LLaVA-OV-7B backbone. *Frame Setting* denotes the temporal input configuration, and *Pre-LLM Tokens/Frame* reports the number of visual tokens from each sampled frame that enter the language model. Methods based only on post-LLM memory compression therefore retain the original 196 visual tokens per frame.

Method	Frame Setting	Pre-LLM Tokens/Frame	Short	Medium	Long	AVG
LLaVA-OV-7B [9]	0.5 FPS	196	70.1	56.4	48.8	58.4
+ ReKV [7]	0.5 FPS	196	70.40	55.40	47.30	57.70
+ StreamMem [22]	0.5 FPS	196	**71.5**	56.6	50.1	59.4
+ StreamingTOM [4]	0.5 FPS	50	71.3	**57.8**	50.6	59.9
+ Vista [10]	0.5 FPS	50	71.5	57.6	48.5	59.2
+ STC [20]	0.5 FPS	58	67.30	53.90	48.30	56.50
+ AudioRouter	0.5 FPS	**64**	68.57	57.14	**53.01**	**60.19**
(b) Effect of temporally available audio context						
Full Vision	0.5 FPS	196	**70.10**	56.40	48.80	58.40
Vision-only ($A_\emptyset$)	0.5 FPS	64	54.29	51.02	43.72	49.81
Audio-Conditioned (A_{t-1})	0.5 FPS	64	62.78	54.44	48.96	55.81
Audio-Conditioned (A_t, Full)	0.5 FPS	64	68.57	**57.14**	**53.01**	**60.19**

highest overall VideoMME accuracy of 60.19% while retaining only 64 visual tokens per sampled frame. Compared with visual-only compression methods operating at similar token budgets, AudioRouter obtains a clear advantage on the Long subset, reaching 53.01%, the best result among the compared methods. This improvement supports our main design: synchronized audio provides useful event context for deciding where to retain visual evidence, which becomes particularly important when long videos must be represented under a limited visual-token budget.

Effect of Temporally Available Audio Context. The lower part of Table 2 examines the effect of audio conditioning under the same query bottleneck and 64-token budget. The vision-only setting ($A_\emptyset$) achieves an overall accuracy of 49.81%. Incorporating the previously observed audio segment (A_{t-1}) improves the accuracy to 55.81%, indicating that past audio provides useful temporal context for visual routing. Using audio available from the current segment (A_t) further raises the accuracy to 60.19%, outperforming A_{t-1} by 4.38 percentage points. The progression from $A_\emptyset$ to A_{t-1} and A_t shows that visual routing benefits from audio context that is more closely aligned with the ongoing event.

4.3 Analysis of Audio-Guided Visual Routing

The central idea of AudioRouter is to use audio as a control signal for visual perception. We therefore examine whether audio conditioning changes how the bottleneck allocates routing weight over the visual tokens. Figure 2 shows representative examples. Given the same visual input, the vision-

only bottleneck distributes its routing weights relatively broadly, whereas the audio-conditioned bottleneck places more weight on regions associated with the current event.

The qualitative comparison suggests that synchronized audio changes visual selection in a structured manner rather than simply adding another input representation. Additional qualitative comparisons with the frozen-teacher reference distribution are provided in Appendix A.5.

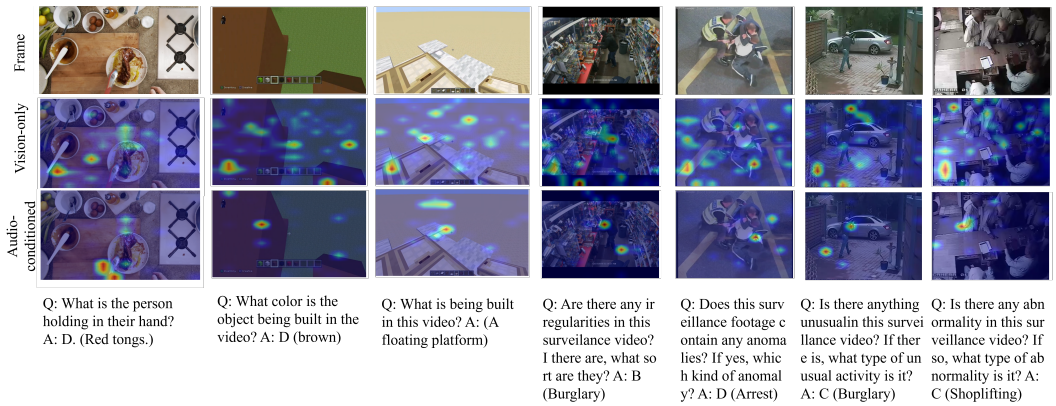


Figure 2: Visualization of audio-guided visual routing.

We further compare the routing distribution of the compressed representation against the reference distribution produced by the full visual model. As shown in Table 3, audio conditioning reduces the routing cross-entropy from 10.4686 to 8.7012 and the JS divergence from 0.5820 to 0.4619. Both metrics indicate that synchronized audio moves the learned visual routing distribution closer to the full-model reference, consistent with our qualitative observations.

4.4 Ablation Studies

Training Objective. We investigate the effects of the knowledge-distillation objective $\mathcal{L}_{\mathrm{KD}}$ and the audio-visual alignment objective $\mathcal{L}_{\mathrm{AV}}$ on the compressed representation. Table 4 reports the corresponding VideoMME performance under a 64-token visual budget. The KD-only and AV-alignment-only variants achieve accuracies of 57.04% and 58.32%, respectively, whereas the full training objective reaches 60.19%. These results indicate that audio-visual alignment provides an effective supervision signal, while the full objective yields the best overall performance under the same token budget.

Table 3: Routing-distribution comparison with and without audio conditioning.

Method	CE ↓	JS ↓
Vision only	10.4686	0.5820
Audio conditioned	**8.7012**	**0.4619**

Table 4: Training-objective ablation under a 64-token visual budget.

Training Setting	Accuracy (%)
KD only	57.04
AV alignment only	58.32
Full objective	**60.19**

4.5 Efficiency Analysis

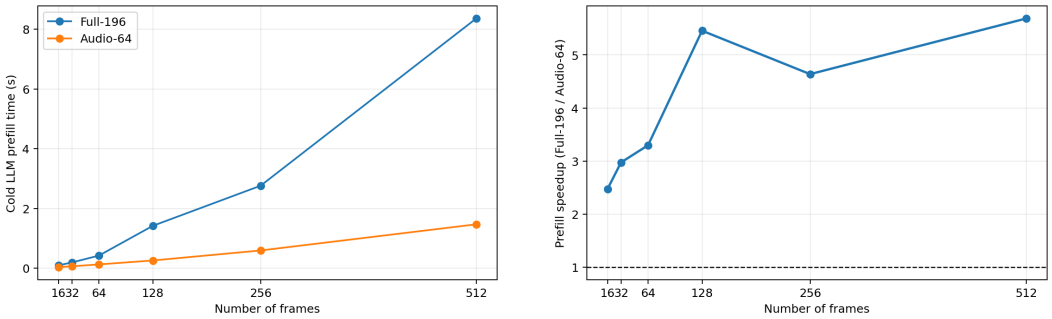
We measure the system-level benefit of visual compression using the complete inference pipeline rather than theoretical FLOPs alone. The measurement includes visual encoding, audio encoding, bottleneck computation, and LLM inference.

Table 5 reports the results. Compared with the full visual representation, the 64-token Audio-Conditioned Bottleneck reduces peak GPU memory from $58.22\,\mathrm{GB}$ to $29.70\,\mathrm{GB}$ and reduces LLM prefill latency from $10.93\,\mathrm{s}$ to $1.85\,\mathrm{s}$. Although audio encoding and bottleneck computation introduce additional work before the LLM, their cost is smaller than the reduction in LLM-side computation obtained by compressing the visual sequence.

Table 5: Measured inference efficiency of the dense visual representation and AudioRouter. The 64-token bottleneck substantially reduces peak GPU memory and LLM prefill latency while preserving the original visual-only LLM interface.

Setting	Tokens	Peak GPU Mem. (GB)	Prefill (s)	TTFT (s)
Full Vision	196	58.22	10.93	23.06
AudioRouter	64	29.70	1.85	20.88

We further study how LLM prefill latency scales with video length. Figure 3 compares the dense visual representation and the proposed 64-token bottleneck as the number of processed frames increases. The efficiency gap grows with the length of the video stream. At 512 frames, Full Vision requires 8.36 s of prefill time, whereas AudioRouter requires only 1.47 s, corresponding to a 5.69× prefill speedup.



(a) LLM prefill latency.

(b) Prefill speedup over Full Vision.

Figure 3: Scaling of LLM prefill cost with increasing video length. (a) Prefill latency for the dense 196-token visual representation and the proposed 64-token Audio-Conditioned Bottleneck as the number of processed frames increases. (b) The corresponding prefill speedup over the dense representation. The benefit of visual compression grows with video length and reaches 5.69× at 512 frames.

These results show that audio-guided visual compression can retain strong video understanding performance while substantially reducing the LLM-side cost of long and streaming video inference.

5 Conclusion

In this work, we introduced AudioRouter, an audio-guided visual compression framework for streaming video language models. Unlike existing audio-visual approaches that incorporate audio as an additional modality for language reasoning, AudioRouter treats synchronized audio as a perception-level routing signal: *audio determines where to look, while vision determines what the LLM sees*. By conditioning visual token routing on audio cues while preserving native visual values, AudioRouter enables effective visual compression without modifying the underlying visual-only VLM interface. Extensive experiments on long-video and streaming benchmarks demonstrate that AudioRouter retains strong understanding performance with only one-third of the original visual tokens, while substantially reducing LLM inference cost and memory usage. These results highlight the potential of leveraging complementary modalities not for direct multimodal fusion, but for guiding more efficient information selection in large models.

AI use statement

In this work, we used generative AI tools solely for language editing of the manuscript, including improving grammar, clarity, readability, and academic phrasing. Generative AI tools were not used to generate, modify, or analyze experimental data, quantitative results, scientific figures, or images. All research methodology, experiments, analyses, and scientific conclusions were developed

and conducted by the authors. All AI-assisted textual edits were carefully reviewed by the authors to ensure that they did not alter the scientific meaning or claims of the manuscript. We take full responsibility for the final content of this work.

Reproducibility statement

We have made substantial efforts to ensure the reproducibility of this work. The complete implementation of AudioRouter is anonymously available at `https://anonymous.4open.science/r/AudioRouter-1F96`. The repository includes the source code, released model weights, and a quick-start tutorial to facilitate straightforward reproduction and evaluation of the proposed method. The main paper describes the model architecture, methodology, and experimental setup, while additional implementation details, hyperparameter settings, and experimental configurations are provided in the supplementary materials where applicable. Together, the released code, model weights, and step-by-step instructions are intended to support independent reproduction and verification of the reported results.

References

Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. In *2023 IEEE/CVF conference on computer vision and pattern recognition workshops (CVPRW)*, pp. 4599–4603. IEEE, 2023.

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, 2024.

Sanyuan Chen, Yu Wu, Chengyi Wang, Shujie Liu, Daniel Tompkins, Zhuo Chen, and Furu Wei. Beats: Audio pre-training with acoustic tokenizers. *arXiv preprint arXiv:2212.09058*, 2022.

Xueyi Chen, Keda Tao, Kele Shao, and Huan Wang. Streamingtom: Streaming token compression for efficient video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24675–24685, 2026.

Zesen Cheng, Sicong Leng, Hang Zhang, Yifei Xin, Xin Li, Guanzheng Chen, Yongxin Zhu, Wenqi Zhang, Ziyang Luo, Deli Zhao, et al. Videollama 2: Advancing spatial-temporal modeling and audio understanding in video-llms. *arXiv preprint arXiv:2406.07476*, 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, 2025a.

Shangzhe Di, Zhelun Yu, Guanghao Zhang, Haoyuan Li, Hao Cheng, Bolin Li, Wanggui He, Fangxun Shu, Hao Jiang, et al. Streaming video question-answering with in-context video kv-cache retrieval. In *ICLR*, 2025b.

Ruohan Gao, Tae-Hyun Oh, Kristen Grauman, and Lorenzo Torresani. Listen to look: Action recognition by previewing audio. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10454–10464. IEEE, 2020.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024.

Zhenyu Li, Zuchao Li, Ping Wang, Lefei Zhang, and Haojun Ai. Vista-LLM: Decoupled query-guided visual token pruning for efficient long-video large language models. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13171–13187, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.601. URL `https://aclanthology.org/2026.acl-long.601/`.

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.

Zhenyu Ning, Guangda Liu, Qihao Jin, Chengwei Li, Wenchao Ding, Minyi Guo, and Jieru Zhao. Livevlm: Efficient online video understanding via streaming-oriented kv cache and retrieval. *arXiv preprint arXiv:2505.15269*, 2025.

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

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

Fangxun Shu, Lei Zhang, Hao Jiang, and Cihang Xie. Audio-visual llm for video understanding. In *2025 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 4305–4314. IEEE, 2025.

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

Changli Tang, Yixuan Li, Yudong Yang, Jimin Zhuang, Guangzhi Sun, Wei Li, Zejun Ma, and Chao Zhang. video-salmonn 2: Caption-enhanced audio-visual large language models. *arXiv preprint arXiv:2506.15220*, 2025.

Keda Tao, Can Qin, Haoxuan You, Yang Sui, and Huan Wang. Dycoke: Dynamic compression of tokens for fast video large language models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18992–19001. IEEE, 2025.

Yapeng Tian, Jing Shi, Bochen Li, Zhiyao Duan, and Chenliang Xu. Audio-visual event localization in unconstrained videos. In *European conference on computer vision*, pp. 252–268. Springer, 2018.

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 (CVPR)*, pp. 18523–18533, June 2026.

Jin Xu, Zhifang Guo, Hangrui Hu, Yunfei Chu, Xiong Wang, Jinzheng He, Yuxuan Wang, Xian Shi, Ting He, Xinfa Zhu, et al. Qwen3-omni technical report. *arXiv preprint arXiv:2509.17765*, 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. *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, 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, 2025.

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. *arXiv preprint arXiv:2406.16852*, 2024a.

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.

A Additional Implementation Details

This appendix provides implementation details that are omitted from the main paper for space. We first clarify the temporal and token-level semantics of AudioRouter, and then describe the backbone, audio-conditioned routing module, optimization setup, and evaluation protocol.

- A Additional Implementation Details
  - A.1 Backbone and Streaming Unit
  - A.2 Causal Audio Frontend
  - A.3 Audio-Conditioned Query Bottleneck
  - A.4 Audio-Guided Visual Routing
  - A.5 Visual-Token Budget Analysis
  - A.6 Optimization Details
- B Streaming Evaluation Protocol
- C Evaluation Details
- D Reference Routing Distribution
- E Why Audio Changes Routing but Not the Visual Value Set
- F Efficiency Measurement

A.1 Backbone and Streaming Unit

We build AudioRouter on `lmms-lab/llava-onevision-qwen2-7b-ov`. The visual encoder is SigLIP SO400M Patch14-384. Each sampled frame is first represented by a 27×27 grid of 1,152-dimensional visual features. The original LLaVA visual projector maps these features to the 3,584-dimensional language-model space, followed by the standard spatial pooling used by LLaVA-OneVision, resulting in 196 visual tokens for each sampled frame.

Although we use "segment" in the main paper to describe a generic streaming step, the released implementation operates at the granularity of sampled frames. We therefore use f below to denote a sampled frame. For F observed frames, the dense representation contains $196F$ visual tokens, whereas AudioRouter passes only $64F$ routed visual tokens to the language model. In particular, the default "64-token" configuration means *64 routed tokens per sampled frame*. The 196-token setting reported as Full Vision bypasses the routing module and uses the original dense representation.

The language model, visual encoder, visual projector, and audio encoder remain frozen. Only the audio-conditioned visual bottleneck is optimized. The default 64-slot adapter contains approximately 2.0M trainable parameters.

A.2 Causal Audio Frontend

We use the fine-tuned BEATs encoder `BEATs_iter3_plus_AS2M_finetuned_on_AS2M_cpt1.pt` and retain its temporal encoder states rather than its AudioSet classifier output. Audio is decoded as mono PCM at 16 kHz.

For a visual frame sampled at timestamp t_f, its audio context contains only the immediately preceding audio interval,

$$a_f = a_{[\max(0, t_f - 2\,\mathrm{s}),\, t_f]}. \tag{1}$$

Intervals near the beginning of a video are left-padded with zeros. Therefore, the routing decision for frame f never observes an audio sample later than the corresponding visual timestamp.

We preserve the temporal BEATs sequence rather than collapsing it into a single global feature. The resulting audio representation has hidden dimension 768 and is projected to the 256-dimensional routing space before conditioning the visual queries.

A.3 AUDIO-CONDITIONED QUERY BOTTLENECK

The bottleneck contains $M = 64$ learnable query slots,

$$Q^0 \in \mathbb{R}^{64\times 256}, \tag{2}$$

initialized from $\mathcal{N}(0, 0.02^2)$. An eight-head cross-attention layer lets these queries attend to the temporal BEATs sequence.

In addition to audio, we provide lightweight temporal context describing the absolute frame timestamp and the elapsed time since the previous sampled frame. Both values are encoded with sinusoidal features using periods

$$\{2, 10, 60, 300, 1800, 7200\} \text{ seconds},$$

and projected through a small MLP. The learned query, audio context, and temporal context are then fused by a shared slot-wise MLP with a residual connection. The same query bank and conditioning module are shared across all frames.

This design keeps temporal adaptation outside the frozen VLM: conditioned queries are used only to construct the visual routing weights and are never passed to the language model.

A.4 AUDIO-GUIDED VISUAL ROUTING

Let

$$V_f \in \mathbb{R}^{196\times 3584}$$

denote the native visual tokens of frame f. A bias-free linear projection maps them into the 256-dimensional routing space,

$$K_f = V_f W_V, \qquad K_f \in \mathbb{R}^{196\times 256}.$$

Given the conditioned queries $Q_f \in \mathbb{R}^{64\times 256}$, the routing distribution is

$$P_f = \mathrm{Softmax}\left(\frac{\mathrm{norm}(Q_f)\,\mathrm{norm}(K_f)^\top}{\tau}\right), \qquad P_f \in \mathbb{R}^{64\times 196}, \tag{3}$$

where the softmax is applied over the 196 visual positions.

The compressed visual representation is then

$$Z_f = P_f V_f. \tag{4}$$

Similarity computation, temperature scaling, and softmax are performed in FP32 for numerical stability. The native-value path contains no learned value projection, output projection, post-routing transformer, or additional normalization.

Importantly, no separate audio-token sequence or projected audio-feature sequence is concatenated to the LLM input. Audio affects the representation through the routing coefficients P_f, while the values aggregated in Z_f are always native visual values.

A.5 QUALITATIVE COMPARISON WITH THE FROZEN TEACHER

In the main paper, we visualize the difference between the vision-only and audio-conditioned routing distributions to illustrate how synchronized audio changes visual evidence selection. Here, we provide additional qualitative comparisons against the reference importance distribution derived from the frozen full-visual teacher.

Following the reference construction described in Appendix **??**, we obtain the teacher distribution by differentiating the correct-answer score with respect to the native visual tokens and computing the normalized Gradient$\times$Activation importance for each visual position. We emphasize that this reference is used only for analysis and is not directly supervised during training.

Figure 4 shows representative examples from different video domains. Compared with the vision-only bottleneck, whose routing weights are often more diffuse or concentrated on visually salient but less event-specific regions, the audio-conditioned router tends to shift its routing mass toward

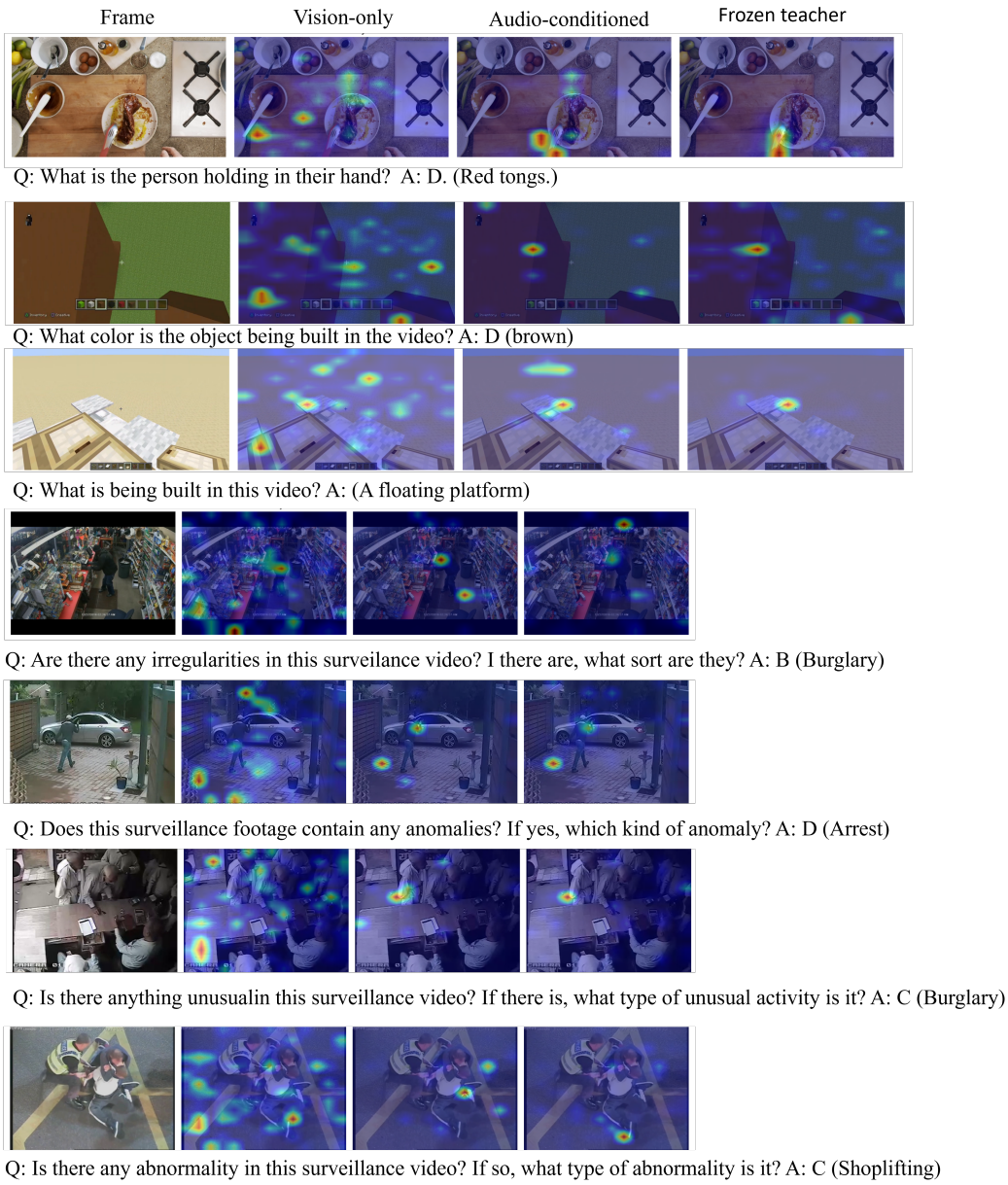


Figure 4: **Qualitative comparison with the frozen full-visual teacher.** Each row shows, from left to right, the original video frame, the routing distribution produced by the vision-only bottleneck, the routing distribution produced by the audio-conditioned bottleneck, and the reference importance distribution derived from the frozen full-visual teacher. Compared with vision-only routing, audio conditioning generally shifts routing mass toward regions emphasized by the teacher reference. The teacher visualization is obtained from the normalized Gradient$\times$Activation importance of the correct-answer score and is used only for analysis; it is not directly used as routing supervision during training.

regions emphasized by the frozen full-visual teacher. This behavior is visible across diverse scenes, including cooking activities, game environments, human interactions, and surveillance videos. The audio-conditioned distribution does not exactly reproduce the teacher reference, nor is it explicitly trained to do so; rather, synchronized audio provides an additional event-level cue that reshapes visual routing toward evidence that is more consistent with the full-visual model.

Table 6: Sensitivity to the number of retained visual tokens on VideoMME. The 64-token setting preserves strong accuracy while using about one third of the original visual-token budget. The 196-token setting still applies audio-guided routing, with 196 input visual tokens routed to 196 output tokens.

Tokens	Short	Medium	Long	Overall
32	65.71	55.78	52.46	58.52
64	68.57	57.14	53.01	60.19
96	67.62	57.14	55.74	60.74
128	59.05	53.06	48.09	53.70
196	71.90	60.54	48.09	60.74

These qualitative observations are consistent with the quantitative results reported in Table **??**. Audio conditioning reduces the cross-entropy to the teacher reference from 10.4686 to 8.7012 and the Jensen–Shannon divergence from 0.5820 to 0.4619. Together, the qualitative and quantitative results indicate that synchronized audio not only changes the routing distribution, but on average moves the compressed visual representation toward the evidence-selection pattern of the frozen full-visual teacher.

A.6 Visual-Token Budget Analysis.

We further investigate how the number of retained visual tokens affects the compression-performance trade-off. As shown in Table 6, we evaluate AudioRouter with different token budgets ranging from 32 to 196 tokens per frame on VideoMME. The results demonstrate that increasing the number of tokens does not always lead to consistent performance improvements, suggesting that the quality of visual token routing is more important than the number of retained tokens alone.

With only 32 tokens, AudioRouter already achieves an overall accuracy of 58.52%, indicating that the audio-conditioned routing mechanism can preserve a substantial amount of task-relevant visual evidence under aggressive compression. Increasing the budget to 64 tokens further improves the overall accuracy to 60.19%, while retaining only 64/196 = 32.65% of the original visual tokens. This setting provides a favorable balance between recognition performance and computational efficiency.

When increasing the token budget to 96 tokens, the overall accuracy improves slightly to 60.74%, while requiring 50% more visual tokens compared with the 64-token configuration. This marginal gain indicates diminishing returns when allocating additional tokens. Interestingly, the 128-token setting performs worse than the 64-token setting, which suggests that a larger token budget alone does not guarantee better representations. Instead, effective compression relies on learning informative routing distributions that allocate the limited visual capacity to event-relevant regions.

The 196-token AudioRouter configuration retains the original token count but still applies audio-guided routing. It therefore differs from the Full Vision baseline, which bypasses the routing module entirely.

A.7 Optimization Details

We freeze the complete base VLM and audio encoder throughout training and optimize only the proposed bottleneck. For the default VideoMME 64-token model, we first train for two epochs with a learning rate of 2×10^{-5} and then resume for one additional epoch with a learning rate of 1×10^{-5} while retaining the AdamW optimizer state.

We use AdamW with weight decay 0.01, without learning-rate warmup or a learning-rate scheduler. The per-step question batch size is one, gradient accumulation is one, and the global gradient norm is

clipped to 1.0. The frozen VLM and extracted visual features are kept in FP16, whereas the trainable adapter and optimizer states use FP32. Decoder gradient checkpointing is enabled. All experiments use random seed 1234.

Training was conducted on two NVIDIA H800 80 GB GPUs. The frozen VLM is placed automatically across available devices; the bottleneck training does not rely on DDP, FSDP, or DeepSpeed.

The training objectives follow Sec. 3 of the main paper. All loss terms operate only on the trainable compression path, while the full visual model is used as a frozen reference.

B Streaming Evaluation Protocol

We follow the timestamp-ordered causal evaluation protocol commonly used in streaming video understanding and adopted by StreamingTOM. For each video, questions are sorted by their associated timestamps and the visible video prefix advances monotonically with time.

Consider a question arriving at timestamp T_q. Let

$$\mathcal{F}(T_q) = \{f_i \mid t_i \leq T_q\}$$

denote the sampled frames that have already arrived. AudioRouter is allowed to use only frames in $\mathcal{F}(T_q)$ and their corresponding causal audio windows. Since the audio window associated with frame f_i ends exactly at t_i, all audio used by the router also satisfies

$$t_{\text{audio}} \leq t_i \leq T_q.$$

No future visual frame or future audio sample contributes to the representation used to answer the question.

Operationally, the video stream is processed in chronological order. When a new question advances the observation time, only newly available frames are added to the current visual history. Previously observed content is not recomputed using information from future frames. If multiple questions fall within an already observed prefix, the same causal history is reused without exposing additional video content.

For implementation efficiency, frozen frontend features can be cached for the sampled timestamps required by a video. This caching does not change the information available to an individual question: every question indexes only its own prefix $\mathcal{F}(T_q)$. Thus, the evaluation is causal with respect to model input even when deterministic frozen features are reused across questions.

For the vision-only control A, BEATs is bypassed and the static learned query bank performs routing. For the previous-audio control A_{f-1}, we shift the encoded audio context by one sampled-frame index and use a zero audio state for the first frame. The current-frame visual input remains unchanged in all three settings.

C Evaluation Details

Following existing streaming VLM evaluations, videos shorter than 30 minutes are sampled at 0.5 FPS and longer videos at 0.2 FPS unless otherwise specified.

For VideoMME, we use the no-subtitle evaluation setting. Decoding uses batch size one, greedy generation, one beam, and at most 16 new tokens. For multiple-choice questions, the first valid answer letter is used for scoring. The reported VideoMME Overall score is computed from the total number of correctly answered questions over all Short, Medium, and Long examples; it is therefore a question-level aggregate rather than the unweighted arithmetic mean of the three subset percentages.

For streaming evaluation, each answer is scored using only the visual and audio prefix available at its query timestamp. Category scores and the overall result are computed as question-level accuracies.

D Training–Evaluation Data Separation

AudioRouter is trained on external video data collected from InternVid, WebVid, YouCook2, Ego4D/EgoQA, VideoChatGPT, and Kinetics. These datasets contain diverse dynamic video con-

tent, including human activities, instructional procedures, egocentric interactions, and general real-world events, providing broad audio–visual supervision for learning visual routing in continuous video streams.

For each downstream benchmark, we train a separate AudioRouter bottleneck using this external training corpus and evaluate the resulting checkpoint on the corresponding benchmark. VideoMME, MLVU, and RTVU are used only for evaluation and are not included in the training data.

During training, the base VLM, vision encoder, visual projector, and audio encoder remain frozen, and only the AudioRouter bottleneck is optimized. We retain videos with valid audio tracks and construct the audio context using only observations available up to the current video timestamp, following the same causal setting used during streaming inference. The training objective follows Sec. 3, combining visual knowledge preservation and audio–visual alignment.

D.1 Causal Controls for Audio Conditioning

Our control settings are designed to isolate the contribution of audio within the same causal streaming process. *Full Vision* serves as the uncompressed reference and directly passes all 196 visual tokens to the LLM. In contrast, the vision-only setting $A_\emptyset$ retains the same query-based bottleneck and the same 64-token budget as AudioRouter, while removing audio conditioning. It therefore provides the matched control for measuring the effect of audio on visual routing under identical compression.

We further vary the temporal context available to the router while keeping the visual input and compression architecture unchanged. A_{t-1} uses only the previously observed audio segment, whereas A_t uses the audio available up to the current visual observation. The resulting comparison $A_\emptyset \rightarrow A_{t-1} \rightarrow A_t$ progressively introduces causal audio context with increasing temporal proximity to the current event.

This control design focuses on in-stream information that is available during causal inference. It separates vision-only routing, routing with historical audio context, and routing with audio aligned to the current observation without changing the underlying video stream or introducing an additional input distribution. Accordingly, our analysis targets how temporally available audio affects visual evidence selection under streaming constraints, rather than robustness to artificially corrupted or unrelated audio inputs.

E Reference Routing Distribution

To analyze whether audio changes visual routing in a meaningful direction, we compare the bottleneck distribution with a reference importance distribution obtained from the frozen full-visual model.

For a given question, we differentiate the correct answer score with respect to a detached copy of the native visual tokens. Per-token importance is computed from the norm of the resulting Gradient$\times$Activation signal and normalized independently for each frame. This yields a full-visual reference distribution over the original 196 visual positions.

For AudioRouter, the routing distribution used for visualization is obtained by averaging the M routing rows,

$$\bar{p}_f = \frac{1}{M} \sum_{i=1}^{M} P_{f,i}. \tag{5}$$

We report cross-entropy and Jensen–Shannon divergence between the normalized reference distribution and $\bar{p}_f$. Both metrics use $\epsilon = 10^{-8}$ for numerical stability. Metrics are first averaged over frames within each video and then across videos.

This reference is used only for analysis and visualization; no teacher attention map or future-frame information is required during inference.

F WHY AUDIO CHANGES ROUTING BUT NOT THE VISUAL VALUE SET

The main paper emphasizes that audio conditions visual compression without introducing a separate audio-valued token stream. Here we formalize this property.

Proposition 1 (Visual convex-hull preservation). For any output slot i, the routing weights satisfy

$$p_{f,ij} \geq 0, \qquad \sum_{j=1}^{L_v} p_{f,ij} = 1, \tag{6}$$

because P_f is obtained by a row-wise softmax over the visual-token dimension. Consequently,

$$z_{f,i} = \sum_{j=1}^{L_v} p_{f,ij} v_{f,j} \in \mathrm{conv}\{v_{f,1}, \ldots, v_{f,L_v}\}. \tag{7}$$

Proof. Each $p_{f,i}$ is a probability vector. Therefore $z_{f,i}$ is a convex combination of the native visual tokens of the current frame. It follows immediately that

$$\mathrm{rowspan}(Z_f) \subseteq \mathrm{rowspan}(V_f), \qquad \mathrm{rank}(Z_f) \leq \mathrm{rank}(V_f). \tag{8}$$

Thus audio can change the routing coefficients and hence the location of the compressed representation within the visual hull, but it does not add an independent audio value vector to the representation. □

This distinction is important. In general,

$$Z_f = P(A_f, V_f, b_f)V_f$$

is audio-conditioned, and therefore Z_f need not be statistically independent of A_f. Our claim is instead structural: conditioned on the routing matrix, aggregation is linear in the visual values, and every output token remains a convex combination of native visual tokens.

A related property follows by fixing V_f and comparing two audio inputs A_f and A'_f. For one output slot,

$$\Delta z_{f,i} = z_{f,i}(A_f, V_f) - z_{f,i}(A'_f, V_f) \tag{9}$$

$$= \sum_j \Delta p_{f,ij} v_{f,j}, \tag{10}$$

where $\sum_j \Delta p_{f,ij} = 0$. Choosing $v_{f,1}$ as a reference gives

$$\Delta z_{f,i} = \sum_{j=2}^{L_v} \Delta p_{f,ij} \left(v_{f,j} - v_{f,1}\right). \tag{11}$$

Therefore,

$$\Delta z_{f,i} \in \mathrm{span}\left\{v_{f,j} - v_{f,1}\right\}_{j=2}^{L_v}. \tag{12}$$

Audio-induced changes are consequently restricted to directions already defined by differences among the current visual tokens.

G EFFICIENCY MEASUREMENT

We measure efficiency using the complete inference path rather than theoretical FLOPs alone. Model loading and video-file decoding are excluded from the timed region. Image preprocessing, the vision encoder, visual projector, spatial pooling, audio decoding and BEATs encoding, AudioRouter, LLM prefill, and answer generation are otherwise measured explicitly.

CUDA synchronization is performed before and after each timed GPU stage. A single prefill pass is used to warm up kernels before measurement. LLM prefill contains the visual embeddings together

with 32 text tokens and uses KV caching. TTFT includes decoder prefill and the LM-head projection required to produce the first token. Generation uses greedy decoding with KV caching.

Peak GPU memory is measured using `torch.cuda.max_memory_allocated`. The reported end-to-end latency includes image preprocessing and, for AudioRouter, audio encoding and bottleneck computation. This distinction allows us to separate the reduction in LLM-side prefill cost from the additional cost introduced by audio-guided compression.

The recorded software environment uses PyTorch 2.5.1 with CUDA 12.4, Transformers 4.53.3, Accelerate 1.9.0, FlashAttention 2.8.0.post2, Decord 0.6.0, and PyAV 15.0.0. Python 3.10 is used throughout.