IR^3: INDEX ONCE, RETRIEVE, RESTORE, AND REDUCE FOR MULTI-QUERY LONG-VIDEO UNDERSTANDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Popular long videos often receive many questions from different users, yet conventional systems repeatedly perform query-independent visual processing for every question. Existing efficiency approaches mainly follow two directions. Query-conditioned keyframe retrieval improves evidence relevance, but typically introduces additional online scoring and selection latency and may fragment temporal context, omitting event development or temporal order. Visual-token compression, in contrast, reduces downstream inference cost by removing redundant visual tokens while largely preserving task performance, but usually operates only on already selected evidence. Moreover, learning fine-grained token-retention decisions from answer-level supervision creates a difficult credit-assignment problem. We introduce IR^3(**I**ndex Once, **R**etrieve, **R**estore, and **R**educe) for efficient multi-query long-video understanding. IR^3 constructs a reusable, query-independent visual index once per video and uses a hybrid global–local retriever to efficiently locate sparse question-relevant anchors. It then restores their contiguous temporal context and reduces the completed evidence to the exact visual-token budget of direct sparse inference. To address diffuse credit assignment, IR^3 replaces fine-grained token selection with structured semantic-block budget allocation, learning how to distribute an exact token budget across semantically coherent regions while resolving representative selection and residual fusion within each block deterministically. Causal retained-content memory further conditions later allocations on previously preserved evidence. Across 6,196 questions from three long-video benchmarks, IR^3's retrieval-only path maintains nearly identical accuracy to WFS32 while reducing steady-state online latency from 8.450 to 0.510 seconds. On videos of at least 30 minutes, IR^3 provides accuracy gains while preserving the same final visual-token budget. Matched-training ablations further show that semantic-block actions provide more stable optimization than token-wise actions under answer-level supervision.

1 INTRODUCTION

On video platforms, popular long videos attract users who ask large vision-language models different questions about the same content. For example, instructional videos prompt questions about concepts, derivations, or procedural steps (Fig. 1). Repeated query-independent visual processing incurs redundant online computation. In this *single-video, multi-query* setting, visual reuse should reduce online latency without compromising question-answering accuracy.

Long-video systems improve efficiency through query-conditioned keyframe retrieval or visual-token compression. Retrieval reduces LVLM input but adds online scoring and selection for each question (Shen et al., 2025; Chen et al., 2026). However, relevant frames alone may lack the context needed for interpretation. Sparse frame selection can still fragment event development, temporal order, and local temporal transitions (Chen et al., 2026). Token compression, in contrast, reduces downstream inference cost while largely preserving task performance (Tao et al., 2025; Fu et al., 2025b), but operates on already selected evidence rather than recovering temporal context omitted by retrieval.

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.

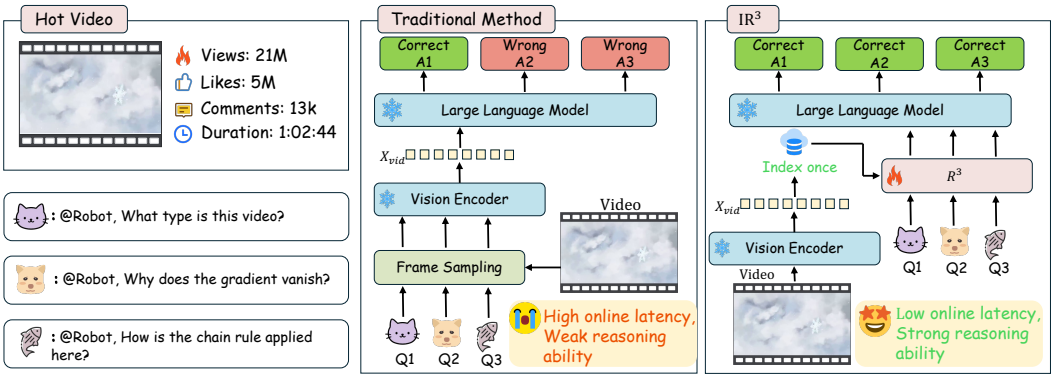


Figure 1: IR^3 for multi-query long-video understanding. Unlike repeated per-query sampling and encoding, IR^3 indexes each video once and reuses it across questions. R^3 retrieves, restores, and reduces evidence before the frozen language model; answer outcomes are schematic.

We introduce **IR^3**—**I**ndex once, **R**etrieve, **R**estore, and **R**educe. One-time query-independent visual indexing supports hybrid global–local retrieval of question-relevant sparse anchors. These support direct answering when evidence is sufficient. For long-video enhancement, IR^3 restores the anchors' contiguous local context and compresses the completed evidence to the exact sparse-inference token budget.

Restoration may introduce tens of thousands of post-ViT tokens. Fine-grained token selection typically makes retention decisions at the token level, yielding a large combinatorial space of possible subsets. When optimized using a single answer-level reward, these decisions are difficult to credit individually, making policy optimization unstable. Compression instead learns question-conditioned budget allocation over connected semantic blocks, with an exact global token budget and deterministic within-block representative selection and residual merging. Full-context answer supervision warm-starts the policy before group-relative DAPO-style alignment.

Across three long-video benchmarks (Wu et al., 2024; Zhou et al., 2025; Fu et al., 2025a), sparse Hybrid32 achieves overall performance comparable to WFS32, while IR^3 modestly improves accuracy under the same final visual-token budget. Clearer Restore–Reduce gains on longer videos motivate direct sparse answering for short videos with sufficient evidence and Restore–Reduce for long videos. Our contributions are:

- We propose IR^3 for multi-query long-video understanding, combining one-time offline visual indexing, cross-query reuse, and hybrid retrieval to substantially reduce online latency while maintaining QA performance comparable to WFS32.
- We propose a retrieve–restore–reduce pipeline that expands sparse keyframes into contiguous segments before compression to the original sparse-inference token budget. This enhances long-video reasoning with low online overhead.
- We design a compressor that replaces fine-grained token selection with semantic-block budget allocation to reduce decision dimensionality. Experiments show more stable policy optimization and higher cross-dataset QA accuracy than token-wise policies.

2 Related Work

2.1 Evidence Retrieval for Long-Video Understanding

Sparse evidence selection includes learned query-aware frame selection (Hu et al., 2025), query-aware frame and resolution adaptation (Zhang et al., 2025), and list-wise optimization of relevance, diversity, and sequentiality (Sun et al., 2025b). Closest to our motivation, WFS-SB detects semantic boundaries from query–frame similarities to preserve narrative structure (Chen et al., 2026). These methods use the selected frames as model input. IR^3 instead treats retrieved frames as anchors,

restores their contiguous local context, and returns the expanded evidence to the original sparse-inference token budget.

2.2 Visual Token Compression for Video LVLMs

FrameFusion and DyCoke exploit cross-frame redundancy (Fu et al., 2025b; Tao et al., 2025); LLaVA-Scissor uses semantic connected components for spatiotemporal compression (Sun et al., 2025a), and OTT-Vid applies optimal transport with adaptive temporal budgets (Kang et al., 2026). CaCoVID and SCORE learn task-aware token selection through reinforcement learning (Ma et al., 2026; Wang et al., 2026), optimizing token combinations and token-level retention, respectively. IR^3 restores contiguous evidence first, then learns how an exact token budget is distributed across connected semantic blocks, with deterministic within-block representative selection and residual merging.

3 Method

3.1 Problem Formulation

The task is long-video question answering (VideoQA) in a multi-query setting. Given a video collection $\mathcal{V} = \{v_i\}_{i=1}^{N}$, each video v_i is associated with a question set $\mathcal{Q}_i = \{q_{i,j}\}_{j=1}^{M_i}$, where N is the number of videos and M_i is the number of questions associated with v_i. The task is to generate a prediction $\hat{y}_{i,j}$ for each question $q_{i,j}$ based on the video content, matching the correct answer $y^*_{i,j}$. Questions arrive independently and are answered separately; query-independent video processing can be performed in advance and reused across questions. The objective is to preserve QA accuracy while reducing online latency from question arrival to answer completion. For an individual video–question pair, subscripts are omitted below, using v, q, and y^* for the video, question, and correct answer, respectively.

3.2 Method Overview

As illustrated in the left panel of Fig. 2, IR^3 comprises one-time offline indexing (**I**ndex Once) and three online stages: **R**etrieve, **R**estore, and **R**educe.

In the offline stage, frozen vision and CLIP image encoders extract visual tokens and frame-level semantic features, respectively, to build indices reusable across questions.

In the online stage, when a question q arrives, the local and global branches of the hybrid evidence retriever produce candidate keyframe sets $\mathcal{A}_q^{\mathrm{loc}}$ and $\mathcal{A}_q^{\mathrm{glo}}$, respectively. These sets are fused into an anchor set $\mathcal{A}_q$ under a fixed frame budget, completing **Retrieve**.

To strengthen reasoning over continuous evidence in long videos, **Restore** expands the anchors to their containing local temporal segments and gathers the corresponding indexed visual tokens. **Reduce** then compresses the restored evidence through semantic-block budget allocation, producing a visual-token sequence $\widetilde{\mathbf{Z}}_q$. Its length matches the actual visual-token count of direct anchor inference. Finally, $\widetilde{\mathbf{Z}}_q$ is organized according to the original spatiotemporal positions and passed to the frozen language model together with the question to generate an answer.

3.3 Hybrid Evidence Retriever

We combine local retrieval (Local) and global retrieval (Global) to select keyframes relevant to question q. Global directly matches the question against indexed frames throughout the video, whereas Local first locates relevant temporal windows and then extracts candidate frames from them. The branches produce the keyframe sets $\mathcal{A}_q^{\mathrm{glo}}$ and $\mathcal{A}_q^{\mathrm{loc}}$ shown in Fig. 2, which are fused into the temporally ordered anchor set $\mathcal{A}_q$. The total frame budget is $B_a = B_g + B_l$, where B_g and B_l control the numbers of global and local frames, respectively.

Global uses a frozen CLIP model for frame-level matching. Suppose the offline index contains T frames, with $\mathbf{c}_t$ denoting the precomputed CLIP image feature of frame t. When a question arrives, the CLIP text encoder produces its feature $\mathbf{c}_q$, and we compute cosine similarity against

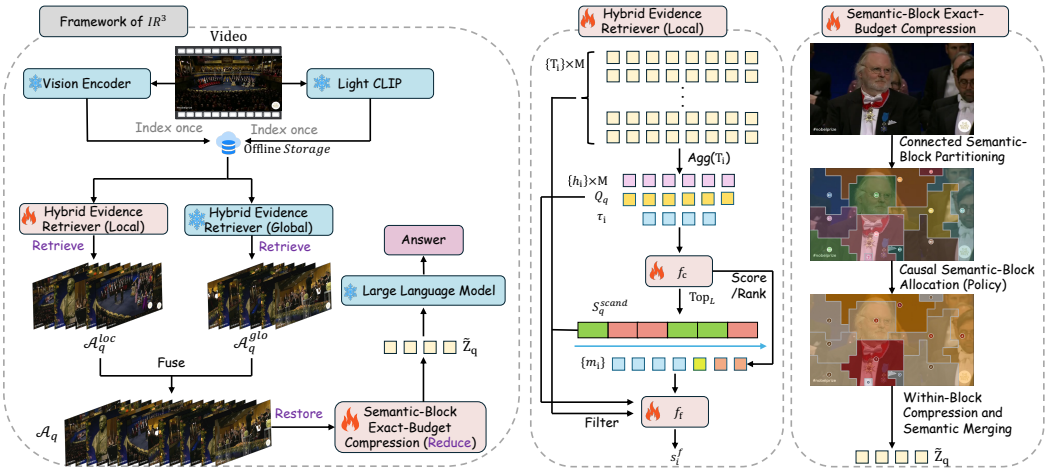


Figure 2: Overview of IR^3. A reusable video index supports learned local retrieval and global CLIP matching; sparse anchors are restored to contiguous context and reduced to an exact token budget by semantic-block compression.

every indexed frame:

$$s_t^{\text{clip}} = \frac{\mathbf{c}_q^\top \mathbf{c}_t}{\|\mathbf{c}_q\|_2 \|\mathbf{c}_t\|_2}. \tag{1}$$

The B_g highest-scoring distinct frames form $\mathcal{A}_q^{\text{glo}}$. This branch searches the complete video timeline without training an additional scoring model.

Local uses fine-grained information in post-ViT visual tokens to locate relevant segments. To balance retrieval quality and online latency, we design a coarse-to-fine structure that first searches compact window representations and then compares a limited candidate set in detail. The indexed timeline is divided into M fixed-duration windows. We denote the visual tokens of window i by $\mathbf{T}_i$ and record its normalized temporal coordinates $\boldsymbol{\tau}_i$, comprising start, end, center, and duration.

To obtain the compact representation $\mathbf{h}_i$ in Fig. 2, we average spatial tokens within each encoder temporal step during offline preprocessing:

$$\mathbf{h}_{i,j} = \frac{1}{P_{i,j}} \sum_{p=1}^{P_{i,j}} \mathbf{z}_{i,j,p}, \qquad \mathbf{h}_i = \text{Agg}(\mathbf{T}_i) = [\mathbf{h}_{i,1}, \ldots, \mathbf{h}_{i,J_i}]. \tag{2}$$

Here, $\mathbf{z}_{i,j,p}$ is the token at spatial position p of encoder temporal step j in window i. $P_{i,j}$ counts spatial tokens at that step, and J_i counts temporal steps in the window. This aggregation reduces spatial redundancy while preserving temporal order; the unaggregated $\mathbf{T}_i$ is retained for fine reranking.

The token-level question representations $\mathbf{Q}_q$ are produced by a frozen language model and are distinct from the CLIP text feature $\mathbf{c}_q$ used by Global. The learned coarse scorer f_c matches $\mathbf{Q}_q$ with each window summary $\mathbf{h}_i$ and incorporates temporal coordinates to compute a relevance score:

$$s_i^c = f_c(\mathbf{Q}_q, \mathbf{h}_i, \boldsymbol{\tau}_i). \tag{3}$$

We retain at most L highest-scoring windows, whose index set is denoted by S_q^{scand} in Fig. 2. Expensive fine-grained comparisons are therefore restricted to this shortlist rather than applied to the raw visual tokens of every window.

The learned fine scorer f_f then reads the unaggregated candidate tokens and jointly compares them using the question representations and candidate metadata:

$$\{s_i^f\}_{i \in S_q^{\text{scand}}} = f_f\left(\mathbf{Q}_q, \{(\mathbf{T}_i, \mathbf{m}_i)\}_{i \in S_q^{\text{scand}}}\right). \tag{4}$$

Here, $\mathbf{m}_i$ corresponds to the candidate metadata in the figure, including temporal coordinates, coarse score and rank, and the token-to-frame-cost ratio; s_i^f is the refined score for window i. Revisiting

fine-grained tokens exposes visual details absent from the compact summaries, while joint comparison assesses relevance across the shortlisted candidates.

The final local-window selection is constrained by a frame-coverage budget B_f. Let c_i count the actual indexed frames in candidate window i, and let $b_i \in \{0, 1\}$ indicate whether it is selected. We solve:

$$\max_{\{b_i \in \{0,1\}\}} \sum_{i \in S_q^{\text{scand}}} b_i \, \sigma(s_i^f) \quad \text{s.t.} \quad \sum_{i \in S_q^{\text{scand}}} b_i c_i \leq B_f, \tag{5}$$

where σ is the sigmoid function. For equal-cost windows, this is equivalent to selecting a fixed number of highest-scoring candidates. The budget B_f limits only the local retrieval pool, not the final keyframe count or the subsequent context completion stage.

We order the selected local windows by their maximum frame-level CLIP scores and traverse them round-robin, taking frames in temporal order within each window. Fusion first retains the B_g Global frames, then skips duplicate positions while accepting B_l additional local frames as $\mathcal{A}_q^{\text{loc}}$. If the local stream is exhausted before the budget is filled, a deterministic uniform traversal supplies the remaining positions. Finally, the selected frames are sorted by time to obtain $\mathcal{A}_q$. Network architectures, training objectives, and the budget-selection algorithm are detailed in Appendix B.

3.4 Anchor-Guided Context Completion

The anchor set $\mathcal{A}_q$ from the preceding stage identifies question-relevant evidence locations. To restore nearby temporal segments, we use simple GOP (Group of Pictures) expansion, mapping each anchor to its containing GOP and retaining the indexed frames within that segment. GOP boundaries are directly available from the video encoding structure, without an additional segmentation model. Unlike the fixed-duration retrieval windows, GOP lengths are determined by the video's encoding structure.

Let $\text{Seg}(a)$ denote the GOP containing anchor a. The completed segment set is:

$$\mathcal{E}_q = \text{TemporalSort}\left(\text{Unique}\{\text{Seg}(a) \mid a \in \mathcal{A}_q\}\right). \tag{6}$$

Here, Unique merges repeated GOPs referenced by multiple anchors, and TemporalSort orders the segments by their source timestamps. For each selected GOP, we retain all frames on its one-frame-per-second index rather than only the retrieved anchors. Completion is therefore guided by anchor locations, without passing every segment of the video to downstream inference.

We then gather the corresponding visual tokens from the offline index and concatenate them in temporal order:

$$\mathbf{Z}_q = \text{TemporalConcat}_{G \in \mathcal{E}_q} \mathbf{Z}_G, \qquad N_q = \sum_{G \in \mathcal{E}_q} N_G. \tag{7}$$

Here, $\mathbf{Z}_G$ is the post-ViT token sequence of segment G, and N_G is its token count. The completed visual sequence is $\mathbf{Z}_q$, with N_q tokens before compression. Each token's original spatiotemporal coordinates and segment membership are retained to support subsequent compression and position reconstruction.

3.5 Semantic-Block Exact-Budget Compression

As shown in the right panel of Fig. 2, the compressor receives all visual tokens $\mathbf{Z}_s$ of a completed segment, the question q, and its budget $K_q^{(s)}$. Semantic partitioning, block-level allocation, and within-block compression produce $K_q^{(s)}$ output tokens. Segment outputs are concatenated in temporal order into $\widetilde{\mathbf{Z}}_q$, with a total budget $K_q = \sum_s K_q^{(s)}$ matching the actual visual-token count of direct anchor inference.

3.5.1 Connected Semantic-Block Partitioning

With answer-level rewards alone, numerous token-wise retention decisions make credit assignment difficult. We reduce the decision dimensionality by grouping adjacent, visually similar tokens into semantic blocks and allocating budgets across regions. Our focus is this reformulation of the decision granularity, so we use a simple connected agglomeration algorithm.

Each encoder temporal step forms a visual slice containing all spatial tokens on an $H \times W$ grid. Initially, every token is a separate region connected to its horizontal and vertical neighbors. The merge affinity of adjacent regions A, B is:

$$s(A, B) = \max_{(i,j)\in\mathcal{E}(A,B)} \cos(\mathbf{z}_i, \mathbf{z}_j), \tag{8}$$

where $\mathbf{z}_i$ is a token feature and $\mathcal{E}(A, B)$ contains four-neighbor edges connecting the regions. To discourage oversized regions, we define the merge priority as:

$$\mathrm{Score}_{\mathrm{merge}}(A, B) = s(A, B) - \lambda_b \left(\frac{|A| + |B|}{HW/R} \right)^2, \tag{9}$$

where $R \leq HW$ is the target block count and λ_b controls the size penalty. We merge the highest-priority adjacent pair and update boundary scores until R blocks remain, independently within each slice. For block B_r, its mean visual feature $\mathbf{b}_r$, capacity $c_r = |B_r|$, and spatiotemporal layout $\mathbf{l}_r$ form the policy input. Implementation details are in Appendix D.

3.5.2 Causal Semantic-Block Allocation

The global budget is distributed across segments in proportion to token capacity. Within each multi-slice segment, a learned temporal head uses question and visual summaries to select the first slice's budget fraction from a fixed candidate set. The remaining budget is distributed across later slices in proportion to their remaining capacities. Capacity-aware rounding produces integer slice budgets $K_q^{(s,u)}$ that sum exactly to the segment budget; detailed rules are in Appendix E. To exploit redundancy across consecutive slices, we process each segment in temporal order, maintaining previously retained representative tokens as causal memory. The decision state at step u is:

$$o_{s,u} = \left(\mathbf{Q}_q,\ \{(\mathbf{b}_r, \mathbf{l}_r, c_r)\}_{r=1}^{R_{s,u}},\ \mathcal{H}_{s,u},\ K_q^{(s,u)} \right), \tag{10}$$

where $\mathbf{Q}_q$ is the token-wise question representation, $R_{s,u}$ is the current block count, and the initially empty memory $\mathcal{H}_{s,u}$ contains earlier retained representatives from the same segment together with their temporal information.

Conditioned on the question, current blocks, and memory, the policy outputs block logits $\mathbf{g}$. The training action is a complete block-quota vector sampled as:

$$\widehat{\mathbf{k}}^{(s,u)} \sim \mathrm{Multinomial}\left(K_q^{(s,u)}, \mathrm{softmax}(\mathbf{g}/\tau) \right), \tag{11}$$

where τ is a temperature parameter. Capacity repair preserves the total budget and produces an executable action $\mathbf{k}^{(s,u)}$, where k_r is the output-token quota of block B_r. After compression, the current representatives update memory, allowing subsequent decisions to account for already retained content.

We warm-start the policy by supervised ranking of candidate allocations, then use DAPO (Yu et al., 2025) to optimize group-relative returns from different samples for the same question. The principal reward is the frozen LVLM's log probability of the correct answer y^*:

$$R(\mathbf{k}) = \log p_{\mathcal{M}}\left(y^* \mid q, \widetilde{\mathbf{Z}}_q(\mathbf{k}) \right), \tag{12}$$

where $\mathbf{k}$ denotes the allocations composing the full compressed output. Inference assigns budgets deterministically from the policy weights instead of sampling. Training objectives, capacity repair, and inference rules are detailed in Appendices F and E.

3.5.3 Within-Block Compression and Semantic Merging

For quota k_r, we start with the token most similar to the block mean and use farthest-first traversal to select k_r representatives. All source tokens in the block are assigned to their most similar representative by cosine similarity, forming clusters $\mathcal{C}_m$. Residual merging gives:

$$\widetilde{\mathbf{z}}_m = (1 - \alpha)\mathbf{z}_{a_m} + \alpha \frac{1}{|\mathcal{C}_m|} \sum_{i\in\mathcal{C}_m} \mathbf{z}_i, \tag{13}$$

where a_m is the representative's token index and α controls merging strength. Merging stays within each block, and outputs retain their representatives' original spatiotemporal coordinates. Segment outputs are restored to their original spatiotemporal order and concatenated into $\widetilde{\mathbf{Z}}_q$ with exactly K_q tokens for the frozen LVLM. Selection details, edge cases, and hyperparameters are in Appendix D.

Table 1: Video-QA accuracy (%). "Pooled" aggregates all 6,196 questions and is not a macro-average over datasets. Within each backbone, all methods use the same answering LVLM; our final visual-token count is matched per question to direct Hybrid32. Each Δ is the accuracy change in percentage points relative to Uniform32 under the same backbone.

		LVB		MLVU-Dev		Video-MME		Pooled	
Backbone	Method	Acc.	Δ	Acc.	Δ	Acc.	Δ	Acc.	Δ
Qwen2.5-VL-7B	Uniform32	56.54	0.00	61.36	0.00	59.85	0.00	59.67	0.00
	CLIP32	61.71	+5.17	**73.05**	**+11.69**	62.53	+2.68	66.04	+6.37
	WFS32	62.68	**+6.14**	72.40	+11.04	63.20	+3.35	66.32	+6.65
	Hybrid32	62.30	+5.76	72.08	+10.72	63.28	+3.43	66.16	+6.49
	IR^3	62.00	+5.46	71.67	+10.31	**65.03**	**+5.18**	**66.70**	**+7.03**
LLaVA-OneVision-7B	Uniform32	55.50	0.00	65.00	0.00	58.70	0.00	60.22	0.00
	CLIP32	59.01	+3.51	71.67	+6.67	60.48	+1.78	64.09	+3.87
	WFS32	**61.93**	**+6.43**	70.70	+5.70	61.04	+2.34	64.62	+4.40
	Hybrid32	61.33	+5.83	71.21	+6.21	61.01	+2.31	64.65	+4.43
	IR^3	61.18	+5.68	**71.80**	**+6.80**	**61.08**	**+2.38**	**64.86**	**+4.64**

Table 2: Qwen2.5-VL-7B accuracy (%) on videos of at least 30 minutes. IR^3 matches Hybrid32's exact-K budget. Bold denotes the best result in each row.

Dataset	n	Uniform	CLIP	WFS	Hybrid	IR^3
LongVideoBench	108	54.63	55.56	**56.48**	53.70	**56.48**
MLVU-Dev	48	52.08	**75.00**	56.25	60.42	66.67
Video-MME	882	49.43	52.38	54.20	53.29	**55.22**
Pooled	1,038	50.10	53.76	54.53	53.66	**55.88**

4 Experiments

4.1 Experimental Setup

Benchmarks and evaluation. We evaluate on LongVideoBench validation (1,337 questions) (Wu et al., 2024), MLVU development (2,174) (Zhou et al., 2025), and a 2,685-question Video-MME subset (Fu et al., 2025a) after exact-ID decontamination, totaling 6,196 multiple-choice questions. Retriever training uses LVBench annotations (Wang et al., 2024); compressor alignment uses a separate Video-R training manifest, with corresponding video-disjoint held-out splits for model selection. We report question-level accuracy on full sets and long-video subsets using a 30-minute threshold fixed before inspecting predictions.

Baselines and controlled protocol. Comparisons use frozen Qwen2.5-VL-7B (Bai et al., 2025) and LLaVA-OneVision-7B (Li et al., 2024), with the same answering model within each backbone. Methods include Uniform32, global CLIP Top-32 (Radford et al., 2021), released WFS32 selections (Chen et al., 2026), direct Hybrid32, and full IR^3. Hybrid32 and IR^3 share ordered anchors and, for every feasible question, the exact final visual-token count. Latency comparisons also cache WFS32's query-independent BLIP-2 (Li et al., 2023) visual features across questions, timing query-conditioned scoring and selection with the same answering pipeline. Dataset provenance, overlap checks, category-level results, and input-manifest and exact-K audits are provided in Appendix G.

4.2 Main Results

We evaluate hybrid retrieval and the contribution of completion and compression to long-video reasoning across three benchmarks and two backbones. As shown in Table 1, retrieval-only Hybrid32 achieves pooled accuracy comparable to WFS32, differing by -0.16 and $+0.03$ percentage points

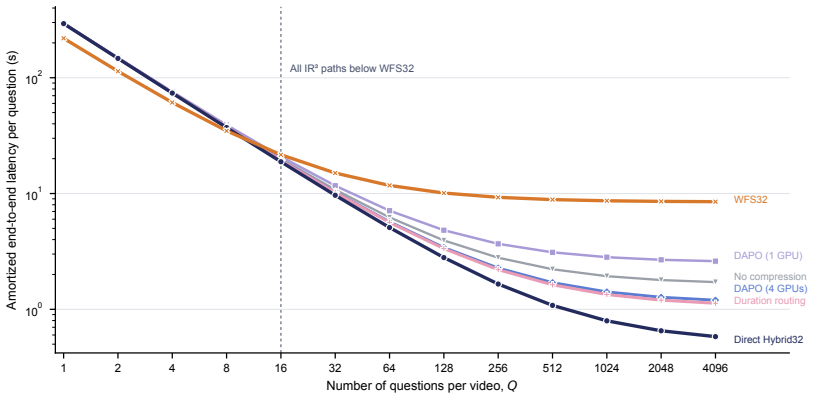


Figure 3: Amortized end-to-end latency for Q questions about the same video. Each point is $L_{\text{offline}}/Q + L_{\text{online}}$, averaged over four duration-stratified videos. The dashed line marks the first evaluated Q at which every IR^3 path is below WFS32; details are in Appendix J.

on the two backbones, respectively. Adding completion and compression raises pooled accuracy to 66.70% and 64.86%, the highest values in the table for each backbone. These results show that hybrid retrieval maintains overall performance comparable to a strong frame-selection baseline, while full IR^3 further improves pooled VideoQA accuracy without increasing the final visual-token count.

To examine the role of completion and compression on longer inputs, we evaluate Qwen2.5-VL-7B on 1,038 questions from videos lasting at least 30 minutes across the three benchmarks. As shown in Table 2, IR^3 outperforms Hybrid32 on all three long-video subsets, increasing pooled accuracy from 53.66% to 55.88% and exceeding WFS32 by 1.35 percentage points. The larger gain over Hybrid32 than on the full evaluation supports restoring local continuous evidence and compressing it within the original token budget to strengthen long-video question answering.

To assess the response cost associated with these accuracy results, we measure latency on four videos of different durations and amortize one-time offline processing over the number of questions per video. Figure 3 shows that the shared index's per-question cost decreases as the number of questions grows; at 16 questions, all measured IR^3 paths are below WFS32. With index reuse, Hybrid32's steady-state online latency is 0.510 seconds, compared with 8.450 seconds for WFS32 with the same offline-precomputation accounting, a speedup of approximately $16.6\times$ at comparable accuracy.

Full IR^3 parallelizes compression across segments, achieving an online latency of 1.129 seconds on four GPUs while improving long-video QA. With sufficient segments, additional workers or better scheduling and communication could further accelerate compression. Current measurements use short answers; compared with uncompressed completed evidence, shorter visual contexts may offer greater computational savings for longer outputs or larger answering models. Stage-wise timings and configurations are in Appendix J.

Overall, IR^3 uses reusable indexing to reduce online latency at comparable QA accuracy, while completion and compression further strengthen long-video reasoning, balancing answer quality and response efficiency in multi-query settings.

4.3 Ablation Studies

To assess the contributions of retrieval and compression, we replace the hybrid retriever with CLIP32 or the learned compressor with uniform exact-K retention on Video-MME (Table 3). Under the same final token budget, full IR^3 improves overall and long-video accuracy over direct Hybrid32 by 1.75 and 1.93 percentage points, respectively. Replacing retrieval or compression reduces overall accuracy by 0.63 or 0.74 points. These comparisons support the contribution of both components to the completed-evidence pipeline. Native completion supplies an uncompressed reference with a larger token budget.

Table 3: Whole-method ablation on the decontaminated Video-MME split (2,685 questions). The four exact-K rows match direct Hybrid32's per-question visual-token count; Native retains all completed tokens.

Configuration	All	<30 min	≥ 30 min
Direct Hybrid32	63.28	68.16	53.29
CLIP32 + completion + DAPO	64.39	**70.05**	52.83
Hybrid32 + completion + uniform	64.28	69.05	54.54
Full IR^3	**65.03**	69.83	**55.22**
Hybrid32 + completion, Native	67.23	71.99	57.48

Table 4: Matched-training action-granularity comparison. AUC is the held-out log-probability gain from initialization. QA budgets are 10% on LVBench and Hybrid32-matched K on LongVideoBench. "Warm start" is supervised initialization; "Full Semantic" adds residual merging to semantic-block DAPO. Protocols are in Appendix I.4.

Metric	Warm start	Flat Token	Grid Block	Semantic	Full Semantic
Positive-AUC seeds $(/3)\uparrow$	-	2	1	**3**	**3**
Mean relative AUC $(\times 10^{-3})\uparrow$	-	21.949	-0.018	0.991	0.711
AUC std. $(\times 10^{-3})\downarrow$	-	52.228	0.363	0.696	**0.126**
LVBench-Train+Dev diagnostic (%) $\uparrow$	46.57	45.21	47.31	47.11	**47.65**
LongVideoBench accuracy (%) $\uparrow$	61.33	59.91	61.26	61.71	**62.60**

Table 5: Compressor comparison on LongVideoBench under matched completed evidence and identical per-question final visual-token budgets.

Retention	OTT-Vid	Ours	Δ
25%	61.11	**61.85**	**+0.74**
10%	58.64	**59.16**	**+0.52**

To test whether semantic-block actions stabilize optimization and whether RL improves on supervised initialization, we compare Warm start, Flat Token, Grid Block, Semantic, and Full Semantic under matched training conditions (Table 4). Full Semantic achieves positive integrated held-out gains in all three seeds and the lowest cross-seed AUC standard deviation (0.126×10^{-3}), compared with 52.228×10^{-3} for Flat Token. Relative to Warm start, it improves accuracy from 46.57% to 47.65% on the LVBench diagnostic set and from 61.33% to 62.60% on LongVideoBench. These results support more stable optimization with structured block actions and show that RL provides additional QA gains beyond supervised initialization.

To isolate compressor quality from evidence selection, we compare our compressor with training-free OTT-Vid on LongVideoBench using identical completed evidence and per-question final token budgets (Table 5). Our method improves accuracy by 0.74 and 0.52 percentage points at 25% and 10% retention, respectively. The gains at both budgets support the effectiveness of learned semantic-block compression on the selected continuous evidence.

Together, these ablations support the complementary roles of hybrid retrieval, context completion, and learned semantic-block compression. Additional sensitivity and retriever analyses are provided in Appendix I; per-seed optimization curves and stage-wise timings are in Appendices I.4 and J.

5 Conclusion

We presented IR^3, a reusable-index framework for multi-query long-video understanding. IR^3 uses Hybrid retrieval to localize sparse evidence, then applies Restore–Reduce to recover contiguous context and compress it to the same visual-token budget as Hybrid32 through semantic-block DAPO. Across three long-video benchmarks and two LVLM backbones, IR^3 improved pooled performance and showed clearer benefits on longer videos. Multi-query latency results further showed that one-time indexing amortizes repeated visual processing across questions, providing a practical balance between low online cost and temporally complete evidence.

AI USE STATEMENT

The core ideas and methodological design of this work were developed by the authors. Generative AI tools, including OpenAI Codex, were used to assist with code implementation and debugging, experimental testing and result organization, manuscript translation and language editing, and figure preparation. The authors checked the relevant implementations and outputs through experimental reruns and result verification, and made the final decisions on methodology, interpretation of results, and conclusions. The authors take full responsibility for all content of the paper, including AI-assisted text, code, and figures.

REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025. URL https://arxiv.org/abs/2502.13923.

Wang Chen, Yuhui Zeng, Yongdong Luo, Tianyu Xie, Luojun Lin, Jiayi Ji, Yan Zhang, and Xiawu Zheng. Wavelet-based frame selection by detecting semantic boundary for long video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24052–24061, June 2026.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, Peixian Chen, Yanwei Li, Shaohui Lin, Sirui Zhao, Ke Li, Tong Xu, Xiawu Zheng, Enhong Chen, Caifeng Shan, Ran He, and Xing Sun. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24108–24118, June 2025a.

Tianyu Fu, Tengxuan Liu, Qinghao Han, Guohao Dai, Shengen Yan, Huazhong Yang, Xuefei Ning, and Yu Wang. Framefusion: Combining similarity and importance for video token reduction on large vision language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22654–22663, October 2025b.

Kai Hu, Feng Gao, Xiaohan Nie, Peng Zhou, Son Tran, Tal Neiman, Lingyun Wang, Mubarak Shah, Raffay Hamid, Bing Yin, and Trishul Chilimbi. M-llm based video frame selection for efficient video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13702–13712, June 2025.

Minseok Kang, Minhyeok Lee, Jungho Lee, Minjung Kim, Donghyeong Kim, Dayeon Lee, Heeseung Choi, Ig-jae Kim, and Sangyoun Lee. Ott-vid: Optimal transport temporal token compression for video large language models. *arXiv preprint arXiv:2605.11803*, 2026.

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

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19730–19742. PMLR, 2023. URL https://proceedings.mlr.press/v202/li23q.html.

Yinchao Ma, Qiang Zhou, Zhibin Wang, Xianing Chen, Hanqing Yang, Jun Song, and Bo Zheng. Contribution-aware token compression for efficient video understanding via reinforcement learning, 2026. URL https://arxiv.org/abs/2602.01649.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina

Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Xiaoqian Shen, Wenxuan Zhang, Jun Chen, and Mohamed Elhoseiny. Vgent: Graph-based retrieval-reasoning-augmented generation for long video understanding, 2025. URL https://arxiv.org/abs/2510.14032.

Boyuan Sun, Jiaxing Zhao, Xihan Wei, and Qibin Hou. Llava-scissor: Token compression with semantic connected components for video llms, 2025a. URL https://arxiv.org/abs/2506.21862.

Hui Sun, Shiyin Lu, Huanyu Wang, Qing-Guo Chen, Zhao Xu, Weihua Luo, Kaifu Zhang, and Ming Li. Mdp3: A training-free approach for list-wise frame selection in video-llms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 24090–24101, October 2025b.

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

Shida Wang, YongXiang Hua, Zhou Tao, Haoyu Cao, and Linli Xu. Dynamic token compression for efficient video understanding through reinforcement learning, 2026. URL https://arxiv.org/abs/2603.26365.

Weihan Wang, Zehai He, Wenyi Hong, Yean Cheng, Xiaohan Zhang, Ji Qi, Xiaotao Gu, Shiyu Huang, Bin Xu, Yuxiao Dong, Ming Ding, and Jie Tang. LVBench: An extreme long video understanding benchmark, 2024. URL https://arxiv.org/abs/2406.08035.

Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. Longvideobench: A benchmark for long-context interleaved video-language understanding, 2024. URL https://arxiv.org/abs/2407.15754.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Shaojie Zhang, Jiahui Yang, Jianqin Yin, Zhenbo Luo, and Jian Luan. Q-frame: Query-aware frame selection and multi-resolution adaptation for video-llms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22056–22065, October 2025.

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

APPENDIX CONTENTS

The appendix is divided into a methodological supplement and an experimental supplement. Section titles and page numbers below are clickable.

Part I: Methodological Supplement

1. Reusable offline video index 13
2. Hybrid evidence retrieval 13
3. Anchor-guided completion and budget construction 15
4. Connected semantic blocks and token merging 16
5. Hierarchical exact-budget allocation 17
6. Full-context supervision and group-relative alignment 18

Part II: Experimental Supplement

1. Datasets, baselines, and controlled protocol 20
2. Implementation and reproducibility details 21
3. Additional results and ablations 21
4. Efficiency evaluation 23

PART I: METHODOLOGICAL SUPPLEMENT

A REUSABLE OFFLINE VIDEO INDEX

A.1 INDEX CONTENTS AND CONSTRUCTION

Before question arrival, each video v is indexed as $\mathcal{I}_v = (\mathcal{C}_v, \mathcal{T}_v, \mathcal{M}_v)$. $\mathcal{C}_v$ stores normalized frame-level CLIP embeddings; $\mathcal{T}_v$ stores post-ViT tokens; and $\mathcal{M}_v$ maps token ranges to timestamps, retrieval windows, codec GOPs, encoder temporal cells, spatial coordinates, and byte ranges. The token index has two logical views:

$$\mathcal{T}_v = \left(\mathcal{T}_v^{\mathrm{ret}}, \mathcal{T}_v^{\mathrm{cmp}}\right). \tag{14}$$

The retrieval view groups the one-FPS timeline into non-overlapping 12-second windows; the completion view is addressed by variable-length codec GOPs. Each view uses its designated frozen visual encoder (Appendix H). Token ranges are shared when encoder weights, preprocessing, frame grouping, temporal pairing, and chunk boundaries coincide; otherwise, compatible ranges are stored separately. All groups are fixed without access to questions.

Index construction consists of four steps:

1. Decode timestamps and codec I-frame boundaries, sample the one-FPS timeline, and assign each position to its retrieval window and GOP. Windows and GOPs may cross each other's boundaries.
2. Encode indexed frames with CLIP and apply each view's frozen visual encoder to its predefined input groups. Identical compatible groups are encoded once.
3. Store post-ViT values without an additional learned projection, together with source indices, temporal pairing, `grid_thw`, post-merge spatial coordinates, dtype, resolution rule, and token order. Spatial mean pooling within temporal cells supplies the coarse summaries defined in Section 3.3.
4. Serialize values and metadata under deterministic video, window, GOP, and token-range keys.

A full retrieval window contains 12 indexed frames and six temporal cells after paired-frame merging. Boundary windows retain their true durations, valid cells, and frame costs. GOP expansion occurs only after anchor selection.

A.2 TOKEN GATHERING AND POSITION RECONSTRUCTION

Sparse answering gathers the indexed ranges associated with its anchors. Restore–Reduce deduplicates selected GOPs, orders them by source time, and gathers their completion-view ranges without decoding or re-encoding RGB. Every token retains its encoder coordinate (t, h, w); compressed representatives inherit their anchor coordinates and are ordered by segment, temporal cell, and spatial position.

Visual-encoder rotary operations are already reflected in stored token values. The injection path reconstructs the visual layout and LVLM-side three-dimensional position identifiers from the metadata, preserving visual boundary tokens, attention masks, and the position-id/rotary-delta convention.

B HYBRID EVIDENCE RETRIEVAL

The retriever returns the ordered anchors $\mathcal{A}_q$ from the indexed CLIP features, retrieval-view tokens, and window metadata. This section specifies the architectures, supervision, and decoding rules underlying Section 3.3.

B.1 Coarse Search

The coarse ranker scores all retrieval windows using compact temporal summaries and normalized start, end, center, and width coordinates. Only the Top-50 candidates load unpooled tokens for fine reranking.

More precisely, write the tokens of window i as $\mathbf{T}_i = \{\mathbf{z}_{i,j,p}\}$, where j indexes its visual temporal cells, p indexes the post-merge spatial positions in that cell, and $P_{i,j}$ is their count. The coarse aggregation used in the implementation is:

$$\mathbf{h}_{i,j} = \frac{1}{P_{i,j}} \sum_{p=1}^{P_{i,j}} \mathbf{z}_{i,j,p}, \qquad \mathrm{Agg_c}(\mathbf{T}_i) = \mathbf{h}_i = [\mathbf{h}_{i,1}, \ldots, \mathbf{h}_{i,J_i}]. \tag{15}$$

Spatial aggregation preserves temporal order, giving $J_i = 6$ for a full 12-second window and retaining only valid cells for boundary windows.

In the current ranker, contextual question states from frozen language-layer indices 1, 12, and 18 are RMS-normalized and mixed with learned weights to obtain query tokens $\mathbf{Q} = \{\mathbf{q}_a\}$. Query and vision use shared-base projections with modality-specific low-rank residuals. A depth-wise temporal convolution over the window summaries produces a contextual vector, which is appended to the projected vectors of that window. Denote this visual set by $\mathbf{V}_i = \{\mathbf{v}_{i,b}\}$. The semantic component retains the strongest visual match for every question token and averages the strongest quarter of the resulting query-token scores:

$$e_{i,a} = \max_b \langle \mathbf{q}_a, \mathbf{v}_{i,b} \rangle, \qquad s_i^{\mathrm{sem}} = \frac{1}{k_q} \sum_{a \in \mathrm{Top}_{k_q}(\mathbf{e}_i)} e_{i,a}, \qquad k_q = \max(1, \lceil 0.25|\mathbf{Q}| \rceil)\,. \tag{16}$$

The normalized start, end, center, and width $\boldsymbol{\tau}_i$ are mapped by continuous sinusoidal features to a temporal embedding $\mathbf{e}_i^t$; the pooled question gives a temporal query $\mathbf{q}^t$ and gate g_q. The coarse logit is:

$$s_i^{\mathrm{c}} = \exp(\lambda_s) s_i^{\mathrm{sem}} + \sigma(g_q) \exp(\lambda_t) \langle \mathbf{e}_i^t, \mathbf{q}^t \rangle, \tag{17}$$

where the two scales are learned. This equation defines the coarse scorer f_{c} used in the main text; all video-side quantities come from the reusable index.

Let $\mathcal{P}$ denote the windows overlapping the annotated evidence interval, o_i their overlap ratios, and s_i^{c} the coarse logits. For interval annotations, o_i is the maximum intersection duration divided by window duration over the annotated intervals; point annotations use window membership. We normalize the graded positive labels as:

$$t_i = \frac{\mathbf{1}[i \in \mathcal{P}](0.25 + o_i)}{\sum_j \mathbf{1}[j \in \mathcal{P}](0.25 + o_j)} \tag{18}$$

and use $\mathcal{L}_{\mathrm{list}} = -\sum_i t_i \log \mathrm{softmax}(\mathbf{s}^{\mathrm{c}})_i$. Writing $p = \max_{i \in \mathcal{P}} s_i^{\mathrm{c}}$, ν_k for the k-th largest negative logit (or the lowest available negative if fewer than k exist), and $\mathcal{H}_{50}$ for the 50 hardest negatives, the budget-boundary and hard-negative terms are:

$$\mathcal{L}_{\mathrm{bnd@}k} = \mathrm{softplus}(0.5 - p + \nu_k), \quad k \in \{20, 50\}, \qquad \mathcal{L}_{\mathrm{hard}} = \frac{1}{|\mathcal{H}_{50}|} \sum_{j \in \mathcal{H}_{50}} \mathrm{softplus}(0.5 - p + s_j^{\mathrm{c}}). \tag{19}$$

For an available wrong question from the same video whose annotation is disjoint from the target interval, let s_i^- be its window logits and set $\mathcal{L}_{\mathrm{cf}} = \mathrm{softplus}(0.25 - p + \max_{i \in \mathcal{P}} s_i^-)$. The deployed coarse checkpoint is trained with:

$$\mathcal{L}_{\mathrm{coarse}} = 0.15\mathcal{L}_{\mathrm{list}} + 0.25\mathcal{L}_{\mathrm{bnd@20}} + 0.45\mathcal{L}_{\mathrm{bnd@50}} + 0.15\mathcal{L}_{\mathrm{hard}} + 0.15\mathcal{L}_{\mathrm{cf}}. \tag{20}$$

The temporal auxiliary coefficient is zero in the selected run.

B.2 Fine Reranking and Budgeted Decoding

For each coarse candidate, the fine ranker receives (i) its unpooled post-ViT tokens, (ii) the contextual question states, and (iii) seven normalized scalar features: start, end, center, duration, coarse

score, coarse rank, and token-to-frame-cost ratio. A shared projection maps question and visual features to 256 dimensions, with rank-32 modality residuals. Eight question-seeded slots read the raw projected visual tokens and all question tokens directly. A four-layer 256-dimensional timeline transformer then compares the at-most-50 candidate vectors. Candidate-signal features are enabled and the frozen coarse score is retained as a residual with scale 0.4. The output is one fine logit s_i^{f} per window. Only these logits decide local-window membership.

Membership is decoded under the candidate-coverage budget $B_f = 240$, measured on the one-FPS search grid. If candidate i has true frame cost c_i, the decoder solves:

$$\max_{\mathbf{b}\in\{0,1\}^L} \sum_{i=1}^{L} b_i\, \sigma(s_i^{\mathrm{f}}) \quad \text{s.t.} \quad \sum_{i=1}^{L} b_i c_i \le B_f. \tag{21}$$

We use exact 0/1 dynamic programming with deterministic candidate-order tie breaking. For ordinary 12-frame windows this is equivalent to retaining the Top-20 candidates; the explicit cost formulation correctly handles the shorter boundary window.

Fusion extracts $B_l = 15$ local anchors from this pool. The B_f constraint applies to retrieval coverage; completion is governed by the final token budget K_q.

The reranker is trained with temporal interval supervision available only on the retriever-training split. Let p_f be the highest positive fine logit, ν_f the 20th largest negative fine logit (the lowest available negative when fewer than 20 exist), and p_o the fine logit of the positive window having maximum temporal overlap. With margin $m = 0.2$ and temperature $\tau_f = 0.2$, the selected checkpoint minimizes:

$$\mathcal{L}_{\mathrm{fine}} = \tau_f \,\mathrm{softplus}\!\left(\frac{\nu_f - p_f + m}{\tau_f}\right) + 0.1\tau_f \,\mathrm{softplus}\!\left(\frac{\nu_f - p_o + m}{\tau_f}\right). \tag{22}$$

The first term makes at least one positive cross the actual 20-window budget cutoff; the second supplies a light graded anchor when several positives exist. The implementation also supports an ℓ_2 penalty on the learned residual, but its coefficient is zero in the selected run; the coarse residual itself is frozen. A training shortlist containing no valid positive or no valid negative contributes no fine-ranking loss. At test time, no temporal label or answer is read.

B.3 Global–Local Fusion

The independent CLIP branch computes $s_t^{\mathrm{clip}} = \mathrm{sim}(\mathbf{c}_q, \mathbf{c}_t)$ over the complete one-FPS grid. The deployed anchor budget is $B_a = 32$, divided into $B_g = 17$ global and $B_l = 15$ local positions. Fusion is deterministic:

1. insert the 17 highest-scoring unique CLIP frames;
2. for every fine-selected window, compute its maximum frame-level CLIP score and order windows by this value, breaking ties by window index;
3. traverse the ordered windows round-robin and insert their one-FPS frames in increasing time within each window until 15 additional unique positions have been accepted;
4. if the local stream is exhausted, fill the residual budget with a deterministic uniform traversal of the indexed video; and
5. sort the 32 unique anchors by source time only after fusion.

A window rejected by the learned fine stage cannot re-enter through local CLIP ordering. For videos shorter than the indexed anchor budget, the fixed short-video route bypasses learned window retrieval and returns the shared deterministic uniformly sampled source-frame sequence. It also bypasses GOP expansion for the controlled compressor comparison, treating the same ordered anchors as one identity-compression unit; no frame is duplicated merely to reach 32 positions.

C Anchor-Guided Completion and Exact Budget Construction

Let $g(a)$ map anchor a to its containing codec GOP. Completion forms:

$$\mathcal{E}_q = \mathrm{sort}_{\mathrm{time}}\left(\mathrm{unique}\{g(a) : a \in \mathcal{A}_q\}\right). \tag{23}$$

For each retained GOP, all one-FPS indexed entries assigned to that GOP are gathered. Repeated GOPs and shared boundary entries are removed before token loading. The gather operation outputs the token sequence $\mathbf{Z}_q$, a segment/slice partition, and a reverse map from each token to its source timestamp and (t, h, w) coordinate. GOPs are used because their boundaries are available without an additional segmentation model; the compressor does not rely on GOP optimality and can accept another deterministic segmenter.

The final budget is measured rather than estimated. For each question, K_q is the actual post-ViT token count of its direct Hybrid32 reference input under the same frozen target-encoder contract. This count is recorded in the comparison manifest before compression. In indexed deployment, it is obtained from compatible token-range metadata rather than additional query-time RGB encoding. If completion produces $N_q > K_q$ tokens, the compressor must return exactly K_q. If $N_q = K_q$, compression is an identity operation. If $N_q < K_q$, the input audit flags the shortfall and excludes that input from an exact-budget comparison; no token is copied to fabricate an exact count.

Input and budget matching across comparison arms is specified in Appendix G.

D Connected Semantic Blocks and Token Merging

Each visual temporal slice is partitioned into connected blocks for learned quota allocation, followed by deterministic within-block compression.

D.1 Connected Agglomeration

After the visual encoder's spatial merge, a temporal slice has an $H \times W$ token grid. We begin with one component per token and connect horizontal and vertical neighbors only. For adjacent components A and B, let s_{AB} be the strongest cosine similarity on their shared boundary and let n_A, n_B be their sizes. With target block count R and ideal size $n_{\text{ideal}} = HW/R$, the merge priority is:

$$\pi(A, B) = s_{AB} - 0.12 \left(\frac{n_A + n_B}{n_{\text{ideal}}} \right)^2 . \tag{24}$$

The highest-priority adjacent pair is merged, boundary scores are updated, and the process continues until exactly $R = 12$ components remain (or $R = HW$ for a smaller grid). A lazy priority queue implements the same rule without recomputing all pairs. Ties are resolved by component index. Components are ordered by their earliest raster index. No edge or merge crosses a temporal slice or completed-segment boundary.

For the quota actor, the visual descriptor of block B_r is its mean post-ViT feature:

$$\mathbf{b}_r = \frac{1}{c_r} \sum_{j \in B_r} \mathbf{z}_j, \qquad c_r = |B_r|, \tag{25}$$

and $\mathbf{l}_r$ contains its normalized segment/time index, spatial center, relative area, and retention-rate context. Thus the descriptor, layout, and capacity used in the main text are deterministic functions of the connected partition rather than additional learned labels.

D.2 Deterministic Representative Merging

Consider block r with token values $\{\mathbf{z}_j\}_{j \in r}$ and assigned quota k_r. For $k_r = 0$ the output is empty; for $k_r \geq |r|$ all tokens are retained. Otherwise, the first representative is the token with maximum cosine similarity to the block mean. Subsequent representatives are chosen by farthest-first traversal: at each step we select the token whose maximum cosine similarity to the already selected set is smallest. Previously selected indices are masked, so identical features cannot select the same anchor twice.

Every source token is assigned to its most similar selected representative. If C_m is the cluster assigned to anchor a_m, the output value is:

$$\widetilde{\mathbf{z}}_m = (1 - \alpha)\mathbf{z}_{a_m} + \alpha \frac{1}{|C_m|} \sum_{j \in C_m} \mathbf{z}_j, \qquad \alpha = 0.25. \tag{26}$$

An empty cluster falls back to its anchor token. Each output keeps the original coordinate of a_m, and outputs are restored to anchor order. Thus merging aggregates complementary content while preserving a valid position for every retained token.

E HIERARCHICAL EXACT-BUDGET ALLOCATION

The allocation hierarchy follows the actual implementation: there is no learned transfer between completed GOPs. Learning begins only after a fixed segment budget has been established. Fixing these budgets removes inter-GOP allocation dependencies, enabling independent compression workers with causal retained-content memory local to each GOP (Appendix J.3).

E.1 SEGMENT AND VISUAL-SLICE BUDGETS

First, K_q is distributed across nonempty completed segments in proportion to their token capacities. We use floor allocation followed by largest-deficit rounding, enforce capacity constraints, and keep at least one token per segment whenever the global budget permits. This produces segment budget $K_q^{(s)}$.

A segment may contain several encoder-defined visual temporal slices. For a multi-slice segment, a temporal budget head chooses a first-slice fraction from:

$$\mathcal{F} = \{0.20,\ 1/3,\ 0.50,\ 0.65\}. \tag{27}$$

Its input concatenates the pooled question, first-slice and later-slice visual summaries; their differences and elementwise query interactions; normalized capacities; first–later similarities; and summary statistics of the frozen block actor's evidence logits. In the Qwen2.5-VL-7B configuration this is a 1,562-dimensional vector. The head is a layer-normalized two-hidden-layer MLP with width 256, GELU activations, and 0.1 dropout.

Given selected fraction f and slice capacities $\mathbf{c}$, allocation is deterministic. If feasible, every slice first receives one token. The first slice is raised toward $\mathrm{round}(fK_q^{(s)})$ without exceeding its capacity or consuming the minima already reserved for other slices. When the budget is smaller than the slice count, this minimum reservation is omitted and the same first-slice and residual rules operate from zero quotas. The residual is distributed among later slices in proportion to their remaining capacities using floor-plus-largest-deficit rounding; the first slice is revisited only after later slices saturate. This yields exact slice budgets $K_q^{(s,u)}$.

E.2 CAUSAL BLOCK-QUOTA ACTOR

Within a slice, the actor receives one mean post-ViT feature per connected block, seven layout scalars, all valid question embeddings, the block capacities, and the representatives actually retained from earlier slices of the same segment. The layout scalars are normalized segment index, temporal index, block center (y, x), relative block size, current baseline retention, and global retention. The output is one logit per active block, interpreted jointly as a quota distribution rather than as independent token scores.

For Qwen2.5-VL-7B, the actor projects 3,584-dimensional question and visual inputs to 256 dimensions, applies eight-head question attention and a two-layer contextual transformer, and adds a log-area prior. For later slices, a separate causal attention branch reads only previously retained representatives. A normalized temporal-age embedding and a centered novelty term correct the current logits; current or future tokens are explicitly rejected from the history. Zero initialization of the causal gates makes the first slice exactly match the warm-started non-causal policy.

At deterministic inference, with slice budget K and block capacities $\mathbf{c}$, the target counts are $\mathbf{t} = K\,\mathrm{softmax}(\boldsymbol{\ell})$. We floor $\mathbf{t}$, cap it by $\mathbf{c}$, and repeatedly assign each residual token to the unsaturated block with the largest deficit $t_r - k_r$. The result is an integer quota vector satisfying:

$$0 \le k_r \le c_r, \qquad \sum_r k_r = K. \tag{28}$$

E.3 EXACT-K GUARANTEE

Every level receives a feasible integer budget no larger than the sum of its child capacities. Floor allocation never exceeds a child capacity, and every rounding or repair step adds a token only to an unsaturated child. Therefore, while a residual exists, at least one legal recipient exists. Summing the exact child budgets first recovers each slice budget, then each segment budget, and finally K_q. The only exception is the explicitly reported $N_q < K_q$ input-shortfall case described in Appendix C.

F FULL-CONTEXT SUPERVISION AND GROUP-RELATIVE ALIGNMENT

F.1 COMPLETE-ACTION WARM START

Warm-start supervision scores complete allocations in the full answer context. Starting from the proportional exact-budget allocation, we construct feasible complete actions by transferring quota from a donor block to an unsaturated receiver while preserving every fixed segment and slice budget. Each candidate is capacity-repaired, composed with the unchanged allocations of all other slices and segments, merged with the same deterministic operator, and inserted into the full video context. The frozen LVLM then assigns:

$$R(\mathbf{k}) = \log p_{\mathcal{M}}\left(y^* \mid q, \widetilde{\mathbf{Z}}_q(\mathbf{k})\right). \tag{29}$$

For each training example we score 20 complete actions, including the proportional baseline. Here $K = K_q^{(s,u)}$ is the target slice budget. Let $g_a = R(\mathbf{k}_a) - R(\mathbf{k}_0)$ and let $\mathbf{m}_a = \mathbf{k}_a/K$ be action a's normalized block-mass vector. We retain the four highest-gain actions and form the answer-reward teacher. With $\bar{g}_a = \mathrm{clip}(g_a/0.025, -8, 8)$, the teacher weights and distribution are:

$$w_a = \frac{\mathbf{1}[a \in \mathrm{Top}_4(\mathbf{g})]\exp(\bar{g}_a)}{\sum_b \mathbf{1}[b \in \mathrm{Top}_4(\mathbf{g})]\exp(\bar{g}_b)}, \qquad \mathbf{p}^* = \sum_a w_a \mathbf{m}_a. \tag{30}$$

If $\boldsymbol{\ell}$ are the policy logits, the primary warm-start loss is $\mathcal{L}_{\mathrm{block}} = -\sum_r p_r^* \log \mathrm{softmax}(\boldsymbol{\ell})_r$. We also require the learned residual $\boldsymbol{\delta}$ to rank the complete actions. With d_a measuring displacement from the proportional baseline, its score is:

$$v_a = 12\left[(\mathbf{m}_a - \mathbf{m}_0)^\top \boldsymbol{\delta} - 0.1 d_a\right], \qquad \mathcal{L}_{\mathrm{act}} = -\sum_a w_a \log \mathrm{softmax}(\mathbf{v})_a. \tag{31}$$

Let $\mathbf{p} = \mathrm{softmax}(\boldsymbol{\ell})$ and let $\mathbf{p}_{\mathrm{area}}$ be normalized block area. Examples are weighted by $\gamma = \mathrm{clip}(\max_a g_a/0.025, 0.15, 2)$, giving:

$$\mathcal{L}_{\mathrm{warm}} = \mathbb{E}[\gamma\,(\mathcal{L}_{\mathrm{block}} + 0.30\mathcal{L}_{\mathrm{act}})] + 0.015 D_{\mathrm{KL}}(\mathbf{p} \,\|\, \mathbf{p}_{\mathrm{area}}). \tag{32}$$

The implementation evaluates the finite displacement penalty as $d_a = \sum_r m_{a,r}[\log \max(m_{a,r}, 10^{-9}) - \log \max(m_{0,r}, 10^{-9})]$. Candidate actions require $K > 0$; zero-budget slices have empty outputs and no allocation-learning example. The residual logits are $\boldsymbol{\delta} = \boldsymbol{\ell} - \log \mathbf{p}_{\mathrm{area}}$. Thus the policy learns from complete-context answer likelihood while the area prior prevents unstable collapse during warm start. No tokenwise pseudo-label or independently evaluated block reward is used.

The temporal head is similarly warm-started by evaluating each candidate first-slice fraction $f \in \mathcal{F}$ while holding the semantic actor and all other allocations fixed. For their complete-context rewards R_f, the target and supervised loss are:

$$u_f = \mathrm{softmax}\left(\frac{R_f - \max_{f'} R_{f'}}{0.05}\right), \qquad \mathcal{L}_{\text{head-sup}} = -\sum_{f \in \mathcal{F}} u_f \log \mathrm{softmax}(\mathbf{h})_f, \tag{33}$$

where $\mathbf{h}$ are the head logits. This trains the head to choose among complete feasible allocations, rather than regress a heuristic duration ratio.

F.2 STRUCTURED ACTION AND CAPACITY REPAIR

For a slice budget K, a rollout samples K categorical draws from the block-logit distribution. Their histogram $\mathbf{n} = (n_1, \ldots, n_R)$ is the latent policy action. If some $n_r > c_r$, capacity repair clamps

overflowing counts and reallocates the missing mass to unsaturated blocks by the same largest-deficit rule used at inference. The exact multinomial log probability of the pre-repair action is:

$$\log \pi(\mathbf{n}) = \log \frac{K!}{\prod_r n_r!} + \sum_{r:n_r>0} n_r \log p_r, \qquad p_r = \mathrm{softmax}(\boldsymbol{\ell}/T)_r. \tag{34}$$

The reward is computed after repair and merging, but policy likelihood is computed for $\mathbf{n}$. The training objective is consequently a latent policy objective; it does not claim to evaluate the marginal likelihood of all latent vectors that repair to the same allocation.

The causal block actor is aligned with four same-question rollouts of these latent quota vectors. A rollout samples the temporally ordered slice decisions of one target GOP while upper-level budgets and other GOP allocations stay fixed. Within a rollout, each slice's retained representatives update the history used by later slices. Every repaired action is merged and evaluated inside the complete LVLM context. For reward R_i, the group advantage is:

$$A_i = \frac{R_i - \mathrm{mean}(R)}{\max(\mathrm{std}(R), 10^{-4})}. \tag{35}$$

Let $r_{i,d} = \exp(\mathrm{clip}(\log \pi_\theta(a_{i,d}) - \log \pi_{\mathrm{old}}(a_{i,d}), -5, 5))$ be the probability ratio for decision d. Here d indexes a slice-level quota decision in rollout i's target-GOP trajectory. The DAPO loss used for the block actor is:

$$\mathcal{L}_{\mathrm{actor}} = -\mathbb{E}_{i,d}[\min(r_{i,d}A_i, \mathrm{clip}(r_{i,d}, 0.80, 1.28)A_i)] + 0.01\,\mathbb{E}_d D_{\mathrm{KL}}(\mathbf{p}_\theta^d \,\|\, \mathbf{p}_{\mathrm{ref}}^d). \tag{36}$$

Here $\mathbf{p}^d$ is the categorical block-probability vector, not the multinomial count distribution: the likelihood ratio uses count likelihoods, whereas KL is averaged over categorical decision distributions. Repaired allocations, retained histories, and answer rewards are fixed throughout the two update epochs; only their policy log probabilities are recomputed. The old-policy log probability is detached, and the reference policy is the frozen warm-start checkpoint. Groups with reward standard deviation below 0.005 and range below 0.015 are skipped because they contain no reliable relative preference. Sampling uses temperature 1.0 and two update epochs per group. This answer-alignment stage updates quota decisions rather than a per-token keep classifier.

F.3 Video-Level Temporal-Head Alignment

The final temporal-head alignment samples one fraction action for every multi-slice completed segment in the video, composes them with the frozen semantic actor, and evaluates the resulting complete video once. Four same-question rollouts form a group. It uses the same normalized advantage, asymmetric $[0.80, 1.28]$ clipped surrogate, bounded log ratio, and categorical reference-KL objective defined above, but its decisions are fraction choices rather than block draws. Groups whose reward standard deviation and range are both below 10^{-5} contribute zero policy gradient. The deployed joint alignment uses reference-KL coefficient 0.01, sampling temperature 1.25, two update epochs per group, learning rate 2×10^{-5}, weight decay 10^{-3}, and unit gradient-norm clipping. Only the temporal head is updated in this final stage; the visual encoder, answer LVLM, semantic actor, and reference head are frozen.

At inference, teacher scoring, sampling, answer labels, and the reference model are removed. The temporal head takes an argmax action, the causal block actor uses deterministic exact-budget rounding, and the within-block merger produces the final temporally ordered K_q tokens.

PART II: EXPERIMENTAL SUPPLEMENT

G DATASETS, BASELINES, AND CONTROLLED PROTOCOL

G.1 DATASETS AND EVALUATION

We evaluate on the 1,337-question LongVideoBench validation split (Wu et al., 2024), the 2,174-question MLVU development split (Zhou et al., 2025), and the 2,700-question Video-MME split (Fu et al., 2025a). Each item is identified by a dataset-qualified question UID, and predictions are scored by exact match between the parsed option label and the ground-truth label. We report both the number of correct answers and question-level accuracy. Video durations are read from source metadata before predictions are inspected. The main long-video analysis fixes the boundary at 30 minutes; Table 7 additionally reports < 10, 10–30, 30–60, and ≥ 60 minute bins.

LVBench training annotations (Wang et al., 2024) are used to learn the retriever, and its video-disjoint development portion is reserved for retriever model selection. Compressor policy alignment instead uses the separate `Video-R1-llava_onevision_filtered.json` manifest and its video-disjoint held-out examples; external benchmark questions are not used to select checkpoints. An exact identifier audit found no overlap with LongVideoBench or MLVU, but it found three LVBench-training video IDs and two LVBench-development video IDs in Video-MME, corresponding to nine and six Video-MME questions, respectively. The exact-ID filter therefore retains 2,685 questions from 895 videos. We additionally screen five fixed relative frame positions per video using perceptual hashes, requiring at least three informative matching positions and comparable frame counts. The retriever-source screen recovers all five known overlaps and finds no additional candidates among 99 available LVBench videos; four training-source videos lack the required assets. A separate screen of all 16,222 Video-R manifest videos against 753 LongVideoBench, 1,242 MLVU, and 900 Video-MME videos has complete asset coverage and yields no candidates under this rule. These sparse screens cannot exclude crops, temporal shifts, or partial reuse, and the Video-R manifest is a superset of optimization inputs. All Video-MME entries in the main tables use the common five-video-filtered subset.

G.2 BASELINES

Uniform32. Frames are distributed uniformly over the available timeline. For videos with fewer than 32 distinct sampled positions, all available positions are used and the actual resulting token count is recorded rather than fabricating duplicate evidence.

CLIP32. All indexed frames are ranked by CLIP (Radford et al., 2021) image–question similarity, and the top 32 distinct frames are restored to temporal order before answering.

WFS32. We evaluate downstream QA using the released WFS (Chen et al., 2026) 32-frame selections as the frame-selection baseline.

Hybrid32. Our sparse retriever combines B_g globally ranked CLIP anchors with B_l anchors supplied by coarse-to-fine local retrieval, where $B_g + B_l = B_a = 32$. This is the direct sparse-inference variant, without context completion or token compression.

Completion + DAPO. The proposed full method starts from the exact ordered Hybrid32 anchors, completes the temporal unit containing every anchor, and compresses the corresponding target-LVLM post-ViT tokens to that question's direct Hybrid32 token count. In indexed deployment, these tokens are gathered from compatible ranges in the reusable video index. Its policy optimization adapts DAPO (Yu et al., 2025) to exact-budget semantic-block compression.

G.3 CONTROLLED INPUT AND BUDGET PROTOCOL

Within each answering backbone, comparison arms share the same frozen checkpoint, pixel bounds, prompt template, decoding rule, maximum output length, and empty system prompt. For each

question, an immutable selection record stores the ordered anchor indices, completed temporal-unit IDs, expanded RGB-frame indices, raw post-ViT-token count, retained-token count, positional metadata, and hashes of all upstream sources. Direct Hybrid32 defines the per-question reference budget K_q using its *actual* visual-token count, including short-video fallback cases. The compressor is accepted only when its retained count is exactly K_q.

Completion arms share ordered RGB inputs, and post-ViT compressors additionally share raw token tensors. Encoder-internal compressors are matched at RGB input and final budget. The following invariants are checked before scoring: UID uniqueness and coverage; valid video paths and frame indices; identical anchors between direct and completion arms; ordered and deduplicated GOP expansion; identical raw RGB frames among completion-based arms; consistent raw-token accounting; valid joint post-ViT partitions; exact-K_q retention; and identical model, resolution, prompt, and decoding configuration. Answer files are append-only, and interrupted runs resume from the set of completed UIDs. LongVideoBench and MLVU checks cover source hashes, recorded inputs, joint partitions, and exact-K. The Video-MME checks cover recorded anchor/GOP/RGB metadata and token counts, supplemented by fresh smoke parity.

H Implementation and Reproducibility Details

The main comparison uses frozen Qwen2.5-VL-7B (Bai et al., 2025) and LLaVA-OneVision-7B (Li et al., 2024); settings are held fixed across comparison arms within each backbone. For Qwen2.5-VL-7B, `min_pixels` $= 100{,}352$, `max_pixels` $= 200{,}704$, and `max_new_tokens` $= 16$, with deterministic decoding and an empty system prompt.

The shared retrieval chain uses frozen 3B-derived representations and the same coarse/fine checkpoints across datasets. Its fixed configuration is one FPS, 12-second windows, Top-50 coarse candidates, a 240-frame local retrieval pool, and 32 anchors comprising 17 global and 15 local positions. Completion and answering use the corresponding target backbone. The Qwen completion view stores BF16 tokens with grid and temporal-position metadata; retrieval and completion therefore use model-specific representations. Index construction and per-question budget accounting are specified in Appendices A and C.

I Additional Results and Ablations

I.1 Complete Results

Table 6 gives the integer counts underlying the Qwen2.5-VL-7B rows of Table 1. The pooled value aggregates 6,196 questions rather than averaging dataset percentages.

I.2 Duration-Routing Sensitivity

We re-score fixed predictions at $t \in \{5, 10, 15, 20, 25, 30, 45, 60\}$ minutes, using direct Hybrid32 below t and completion + DAPO otherwise, without retraining. Figure 4 reports the full sweep. Evaluation-set maxima occur at 25/10/5 minutes for LongVideoBench/MLVU-Dev/Video-MME, yielding 4,154/6,196 correct answers (67.04%). The best shared threshold is five minutes, yielding 4,143/6,196 (66.87%). These are post-hoc sensitivity results; the prespecified 30-minute analysis split remains unchanged.

The dataset-specific thresholds route 1,178/1,337 LongVideoBench, 1,682/2,174 MLVU-Dev, and 1,011/2,685 Video-MME questions through Hybrid32: 62.48% overall, with 2,325 questions using completion + DAPO. Threshold calibration is separate from the retrieval and compression modules.

I.3 Retriever Diagnostics

The evidence-hit diagnostic in Table 8 uses annotated LongVideoBench evidence intervals. A prediction is a hit when at least one of the 240 retained frames intersects an annotated interval. Coarse retrieval raises hit rate from 61.93% to 70.46%; learned fine reranking further raises it to 74.83%, adding 4.37 percentage points within the same frame budget. The Top-50 oracle measures headroom and is not a deployable baseline.

Table 6: Qwen2.5-VL-7B correct-answer counts on the main evaluation manifests. The routed row uses the post-hoc thresholds evaluated in Fig. 4.

Method	LVB / 1,337	MLVU / 2,174	Video-MME / 2,685	All / 6,196
Uniform32	756	1,334	1,607	3,697
CLIP32	825	1,588	1,679	4,092
WFS32	838	1,574	1,697	4,109
Hybrid32	833	1,567	1,699	4,099
Completion + DAPO	829	1,558	1,746	4,133
Duration routing	842	1,572	1,740	4,154

Table 7: Qwen2.5-VL-7B pooled accuracy (%) by video duration. Native retains all completed tokens; completion + DAPO matches each question's direct Hybrid32 token budget. The bins partition the same 6,196 questions.

Duration (min)	Questions	Hybrid32	Native	Completion + DAPO
< 10	3,948	70.47	73.35	70.31
10–30	1,210	62.81	63.80	64.21
30–60	995	53.27	57.49	55.48
≥ 60	43	62.79	67.44	65.12

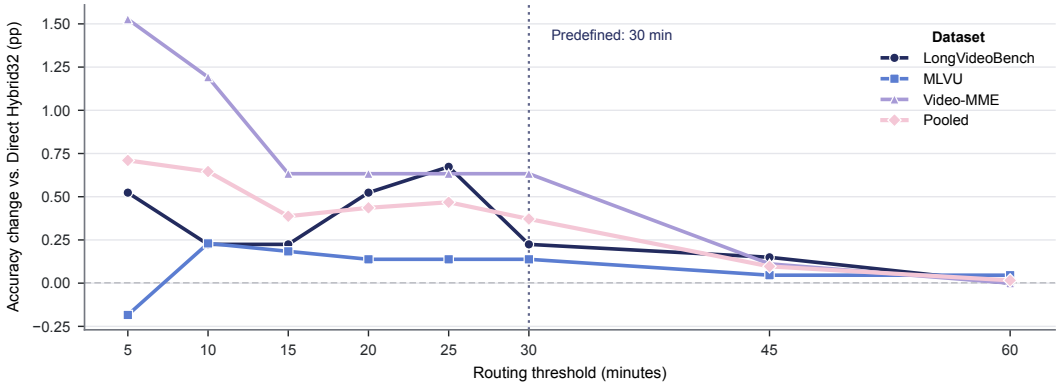


Figure 4: Sensitivity of duration-based routing. For threshold t, videos shorter than t use direct Hybrid32 and the others use completion + DAPO. Curves show accuracy change relative to always using direct Hybrid32 on the aligned 6,196-question evaluation. The dotted line marks the prespecified 30-minute analysis split; Table 6 reports the corresponding per-dataset maxima from this post-hoc sweep.

Table 8: LongVideoBench evidence hit rate under a 240-frame retrieval budget.

Candidate construction	Any-hit rate (%)
Uniform 240	61.93
Learned coarse retrieval	70.46
Coarse + learned fine reranking	74.83
Top-50 oracle	87.36

I.4 Credit-Assignment Granularity

Table 4 summarizes the action-granularity comparison; Fig. 5 provides its training-progress evidence. Every method uses its own partition-matched Stage-1 initialization; all curves use the same 512 video-disjoint validation examples, optimization steps, rollout count, teacher-call budget, and global exact-K constraint. The Flat Token result is dominated by seed variation, whereas both semantic-block configurations have positive relative AUC in all three seeds. Residual merging further reduces the semantic policy's AUC standard deviation from 0.696×10^{-3} to 0.126×10^{-3}.

For each method and seed, let $g(t)$ be the held-out mean gold-answer log-probability minus its step-0 value. Relative AUC is the trapezoidal integral of $g(t)$ over steps $\{0, 16, 32, 48, 64\}$ divided by 64. The reported standard deviation is the across-seed dispersion of this normalized gain.

The QA rows of Table 4 use seed 20260825. LongVideoBench uses 1,337 questions and direct-Hybrid32-matched budgets; LVBench uses 1,471 train+dev questions at 10% retention. The latter is a conditional compressor diagnostic on retriever-development data. These matched-training checkpoints are separate from the historical main-method checkpoints.

I.5 Compressor Sensitivity

Retention ratio. On 294 video-disjoint LVBench development questions, Full Semantic obtains 46.60, 48.64, 46.26, and 45.92% at 25, 20, 15, and 10% exact post-ViT retention, respectively. All anchors, completed evidence, raw tokens, and other settings are fixed. Figure 6 visualizes this result together with the residual-fusion sweep.

Residual-fusion robustness. Holding the learned actions, per-GOP quotas, retained positions, inputs, and exact budgets fixed, Fig. 6 shows nearly flat accuracy for $\alpha \in \{0, .10, .25, .50\}$, indicating that performance does not depend on a narrowly tuned fusion coefficient.

I.6 Completion and Retention Diagnostics

Completion segments. We compare codec GOPs with non-overlapping 12-second windows on 512 answer-independent questions while reusing identical anchors. The alternatives have closely matched expansion cost (211.55 versus 212.03 frames on average; median 216 for both). GOP and fixed-window completion obtain 68.95 and 68.36% accuracy, respectively, with 98.63% answer agreement. Thus, codec boundaries are a simple, low-overhead choice whose performance is competitive with a fixed-window alternative in this uncompressed completion diagnostic.

Uniform and random retention. On the same 512-question subset, we replace the learned compressor with either chronologically uniform or fixed-seed random token retention while preserving the completed RGB input, joint post-ViT tokens, 3D positions, and each question's direct-Hybrid32 budget. Direct Hybrid32, uncompressed completion, uniform retention, random retention, and Full Semantic DAPO obtain 68.16, 68.95, 68.36, 68.95, and 69.73% accuracy, respectively. DAPO improves the uniform and random point estimates by 1.37 and 0.78 points, respectively.

J Efficiency Evaluation

J.1 Measurement Protocol

The reusable index in Appendix A is constructed before question arrival and shared across questions. Figure 3 counts decoding and visual-index computation as offline cost; serialization and persistent disk writes are excluded consistently across methods. Steady-state timing starts before question encoding and includes retrieval, token gathering, H2D transfers, partition and position preparation, compression, input assembly, prefill, and answer decoding. One-time index loading is excluded from steady-state latency.

Measurements use A800 GPUs and four videos lasting approximately 47, 680, 2,417, and 7,979 seconds. After warm-up, each configuration is measured five times per video; we average the four per-video medians. Original query-batching contexts are preserved, charging the full batch wall time to the target question without division by batch size. Total latency is measured separately from stage-wise instrumentation. Short multiple-choice generation uses `max_new_tokens` $= 16$; decoding takes approximately 0.039 seconds and is included in the total.

Hybrid32 retrieval includes question encoding, coarse and fine ranking, CLIP scoring, and anchor fusion. WFS32 likewise reuses offline BLIP-2 (Li et al., 2023) visual features; its timed retrieval includes question preprocessing, index transfer, query-conditioned Q-Former/ITM scoring, and released wavelet–MMR selection. Both paths then use selected-token gathering, position reconstruction, and the same frozen 7B answerer. Input and budget controls follow Appendix G.

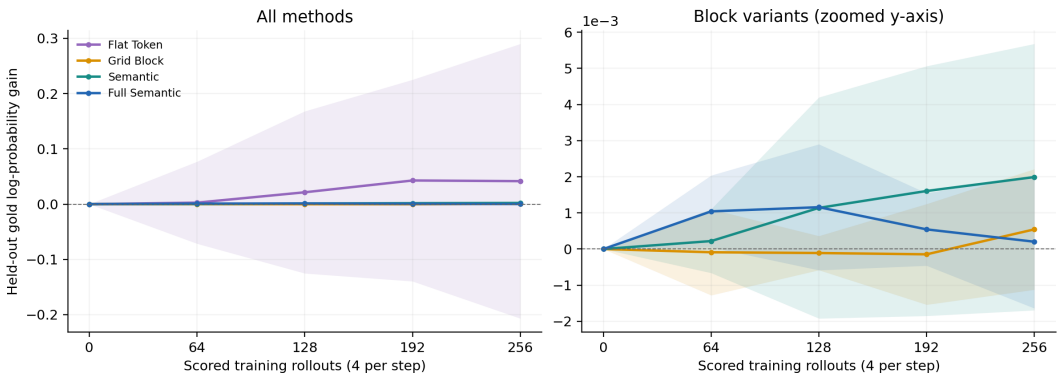
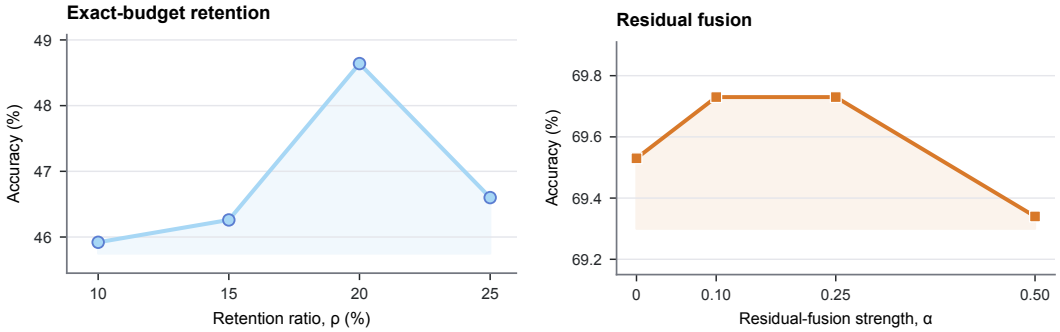


Figure 5: Held-out gold-answer log-probability gain relative to each method's own initialization. Lines are three-seed means; shading is a pointwise Student-t 95% interval (2 degrees of freedom). The right panel explicitly zooms the block variants. The horizontal axis counts four training rollouts per step, excluding initialization and validation.



(a) Exact-budget retention

(b) Residual fusion

Figure 6: Compressor sensitivity. (a) Exact-K retention on 294 LVBench development questions. (b) Residual-fusion strength on the fixed 512-question LVB+MLVU diagnostic subset. Other settings are fixed within each panel.

J.2 Stage-Wise and Duration-Routed Latency

Across the four videos, WFS32 selection ranges from 0.155 to 23.255 seconds, compared with 0.006 to 0.286 seconds for Hybrid32. Table 9 reports stage-wise and full-path measurements.

Four-GPU compression uses persistent workers for separate GOPs, with retrieval and answering on the primary GPU. Communication, result collection, and worker waits are included. Single- and four-GPU runs produce identical actions, allocations, token tensors, and 3D positions on these examples, and pass anchor, exact-K, and answer-option checks.

Duration routing applies the thresholds in Appendix I.2 to measured branch executions. The 47-second Video-MME example uses Hybrid32; the 680-second MLVU, 2,417-second Video-MME, and 7,979-second MLVU examples use four-GPU completion + DAPO. Averaging their per-video medians yields 1.015 seconds to the first token and 1.056 seconds to answer completion. Compression averages 0.537 seconds over this one-direct/three-completion mix.

J.3 Parallelism and Inference-Cost Analysis

Fixed GOP budgets and GOP-local causal memory permit independent compression across GOPs, with sequential decisions within each GOP. Four workers reduce compression time from 2.009 to 0.604 seconds ($3.33\times$); full-path latency is 1.129 seconds, versus 1.647 seconds for single-GPU uncompressed completion. These configurations use the GPU counts reported in Table 9; the comparison concerns latency rather than equal-resource throughput.

Table 9: Steady-state online latency (seconds) with a reusable, query-independent video index. Question-conditioned retrieval, token gathering, and H2D transfers are included in timing.

Configuration	GPUs	Retrieval	Read/H2D	Compress	Prefill	Total
Direct Hybrid32	1	0.167	0.003	—	0.292	**0.510**
WFS32	1	8.116	0.003	—	0.295	8.450
Completion, no compression	1	0.168	0.011	—	1.416	1.647
Completion + DAPO, optimized	1	0.168	0.011	2.009	0.291	2.534
Completion + DAPO, parallel	4	0.168	0.011	0.604	0.292	1.129
Duration routing, parallel	4	0.168	0.011	0.537	0.292	**1.056**

A schematic compression cost on P workers is:

$$C(P) \approx H(P) + \max_{1 \le p \le P} \sum_{s \in \mathcal{S}_p} c_s, \tag{37}$$

where c_s is a GOP's compression time, $\mathcal{S}_p$ its worker assignment, and $H(P)$ scheduling and communication overhead. More workers can shorten the critical path when independent GOPs are sufficiently numerous and balanced.

For Native completion with $N_q > K_q$, let $F_m(n)$ and $D_m(n, L)$ denote prefill and decoding latency for model m, input size n, and output length L, holding the text prompt and decoding configuration fixed. Let $\Delta C(P, m)$ be IR^3's additional non-answering cost, including compression, data movement, and input assembly. The latency saving is:

$$\begin{aligned} \Delta T(P, m, L) &= T_{\text{Native}} - T_{\text{IR}^3} \\ &\approx F_m(N_q) - F_m(K_q) \\ &\quad + D_m(N_q, L) - D_m(K_q, L) - \Delta C(P, m). \end{aligned} \tag{38}$$

A shorter visual prefix can reduce attention and KV-cache access during decoding. Longer outputs or larger models may therefore favor compression when answering-stage savings exceed its overhead. These are analytical extensions of the measured short-answer, one- and four-GPU configurations.

Native completion retains all N_q tokens and provides an uncompressed quality reference. Direct Hybrid32 and IR^3 both retain K_q tokens; their comparison tests evidence enrichment at a matched final budget. Peak system memory additionally depends on intermediate tensors and worker-resident models.