MULTIPLE QUESTIONS, SHARED EVIDENCE: BUDGET-AWARE GROUPED INFERENCE FOR LONG-VIDEO QA

Anonymous authors
Paper under double-blind review

ABSTRACT

Long videos are frequently asked about using multiple questions instead of a single one, and a fixed visual-token budget must cover the entire batch; naively re-encoding video content for every question wastes this budget as the batch grows, turning batch video question answering into a resource-allocation problem. Existing methods sit at two extremes: answering each question with its own context repeatedly re-encodes overlapping evidence, while answering the whole batch with one shared context discards question-specific details, and neither adapts to how much the questions actually overlap. We introduce `QShare`, a batch VideoQA framework that treats *which questions should share a context* and *how the shared budget is spent* as one joint decision rather than committing to either extreme. `QShare` forms an initial partition from question–frame evidence overlap, allocates frames across groups with a submodular greedy procedure that provably approximates the optimal coverage under the shared budget, and iteratively refines the partition — moving, splitting, or merging questions — based on the contexts produced by the allocation; it then answers each group with a single multimodal language-model call. Across `CinePile`, `CG-Bench`, and `LVBench` with two backbones, `QShare` improves average accuracy by 4.0 points over the strongest baseline under matched batch budgets, reduces the number of LMM calls and the amount of repeated visual prefilling, and keeps its accuracy advantage as the number of questions per video grows.

1 INTRODUCTION

Long-form videos are increasingly treated as a persistent source of information, where users frequently issue multiple questions about the same video instead of making a single query. This multi-question setting is already reflected in long-video benchmarks: LVBench (Wang et al., 2025) pairs each video with about 15 questions on average. As the number of questions grows, processing each question independently could repeatedly consume overlapping visual evidence, causing the total visual-token cost to grow across the batch. This turns batch VideoQA into a resource allocation problem: a limited visual-token budget must be distributed across questions with potentially overlapping evidence needs.

This batch VideoQA task is challenging because different questions often require partially overlapping visual evidence. As shown in Figure 1, allocating frames independently to each question preserves question-specific evidence, but repeatedly processes frames that are relevant to multiple questions. On the other hand, processing all questions jointly over a single shared context avoids such redundancy, but forces every question to operate over frames selected for the entire batch, including frames irrelevant to that particular question, while potentially missing question-specific details. The key challenge is therefore to jointly determine which questions should be executed together and which visual frames should be assigned to each resulting group, so as to maximize overall answer quality under the shared batch-level budget.

Existing methods can be categorized by how they share visual computation across questions. Query-agnostic methods (Song et al., 2024; Zhang et al., 2025a; Shu et al., 2025; Yang et al., 2025a; Shang et al., 2025) compress the video once into a universal context that all questions reuse, but because the context must be general enough to cover all questions, it often wastes tokens on irrelevant content and may miss details needed for a specific question. Query-conditioned methods (Chen et al., 2024a;

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.

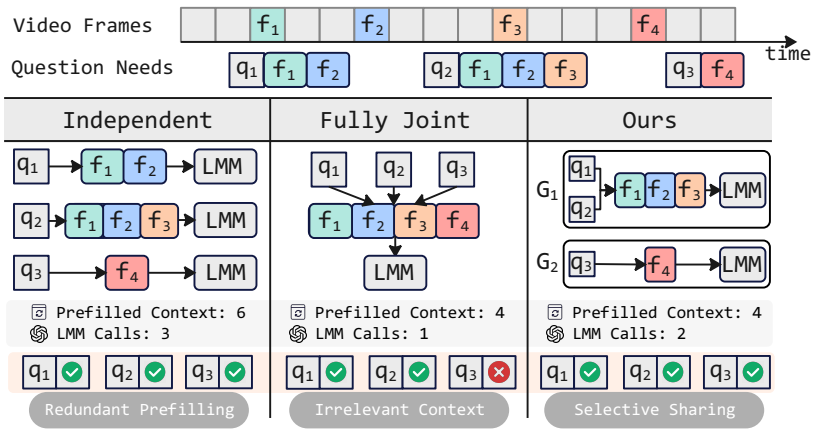

Figure 1: **Batch VideoQA calls for adaptive question grouping.** f_1–f_4 are sets of video frames at different moments of a long video; each question needs a different subset. *Independent* execution processes the overlapping f_1, f_2 twice and issues three LMM calls. *Fully joint* execution processes all frames once but makes every question attend to irrelevant frames. Our method forms groups G_1, G_2 (q_1, q_2 share overlapping evidence, q_3 is separate): shared frames are processed once, unrelated frames stay apart, in two calls.

Zhang et al., 2025b; Huang et al., 2025; Liu et al., 2025; Qian et al., 2024; Shen et al., 2024) select or prune evidence for each question, which improves relevance; however, they must re-select the context and redo prefilling for every question, duplicating computation. Cross-query reuse methods (Di et al., 2025; Khaki et al., 2025; Yang et al., 2025b; Xiao et al., 2026) sit in between by caching a video representation and retrieving query-relevant information from it, yet each question still selects its evidence and is decoded in a separate inference process. None of these methods decides which questions should share a context and which should not, so the sharing pattern is fixed in advance rather than chosen for the specific batch.

To address this gap, we introduce `QShare`, which constructs an execution plan for the entire question batch by jointly deciding which questions should be executed together, how much of the global visual-token budget each group should receive, and which visual evidence should be included in each group's context. Specifically, `QShare` first scores each candidate visual frame against every question and constructs an initial question partition based on the overlap of their top-ranked frames. Given this partition, `QShare` jointly allocates the global token budget and selects group-specific frames by greedily assigning each frame to the group for which it provides the largest marginal increase in evidence coverage. This naturally prioritizes frames that support multiple questions while allocating additional frames to questions that remain under-covered. Since the initial partition is constructed without considering the available budget, `QShare` then refines the question groups based on the evidence that can actually be supported under that budget, considering question moves, singleton splits, and group merges. Each candidate partition is re-evaluated after reallocating its frames, and the refinement continues until no further improvement is found. Finally, `QShare` answers all questions in each resulting group in a single LMM call over its selected visual frames, allowing shared evidence to be processed only once within the group.

- We formulate batch VideoQA under a shared visual-token budget as two coupled decisions: how to partition questions into execution groups, and which frames to allocate to each group. This makes execution granularity a decision variable, rather than fixing it to per-question or fully joint execution.
- We propose `QShare`, a budget-aware grouped inference framework that jointly organizes questions and visual evidence at the batch level. `QShare` initializes groups by overlap in question-relevant evidence, allocates frames across groups by marginal coverage gain, and then iteratively refines the partition via question moves, singleton splits, and group merges. Each final group is answered in a single LMM call using its shared context.
- We evaluate `QShare` on three real-world benchmarks with two LMM backbones. Under the same budget, `QShare` improves over per-question and fully joint execution by 7.3 and

4.4 points on average, respectively, and outperforms the strongest baseline by 4.0 points in average accuracy, consistently across budget levels and backbones.

2 Related Work

Visual Token Reduction. A key challenge in long video question-answering is efficiently compressing the content from lengthy videos. Query-agnostic methods compress a video without access to downstream questions. MovieChat (Song et al., 2024) and Flash-VStream (Zhang et al., 2025a) maintain a compact visual memory offline to retain long-range content, while Video-XL (Shu et al., 2025) compresses short temporal intervals into concise visual summaries. VisionZip (Yang et al., 2025a) and LLaVA-PruMerge (Shang et al., 2025) select salient visual tokens and merge or drop redundant ones. Because the resulting context is independent of a particular question, it can be reused across queries. It must nevertheless preserve sufficiently broad visual information to support different possible questions, which may allocate tokens to irrelevant content and still risk discarding details that are critical for specific questions.

Query-conditioned methods instead select or prune visual evidence for each question. FastV (Chen et al., 2024a), Q-Frame (Zhang et al., 2025b), PruneVid (Huang et al., 2025), and KVTP (Liu et al., 2025) identify relevant frames or tokens; VideoStreaming (Qian et al., 2024) and LongVU (Shen et al., 2024) combine reusable video representations with query-aware access. Recent agentic approaches, such as VideoAgent (Wang et al., 2024), LongVideo-R1 (Qiu et al., 2026) and LongVT (Yang et al., 2026), further extend this idea by iteratively navigating the video as reasoning progresses. These approaches provide more focused visual evidence, but select visual context independently for every question. Query-agnostic and query-conditioned methods therefore represent two fixed settings: a common context for all questions, or a separate context per question, without determining whether subsets of questions with overlapping evidence should share a context.

Cross-Query Reuse. A related line of work targets the efficiency of serving multiple questions about the same video. ReKV (Di et al., 2025) stores processed video KV caches and retrieves query-relevant entries, while SparseVILA (Khaki et al., 2025) adopts a similar two-stage design by decoupling a shared visual prefill from subsequent query-aware access, so different questions can reuse upstream visual computation. StreamMem (Yang et al., 2025b) and MuKV (Xiao et al., 2026) further improve how persistent video KV caches are compressed and organized to reduce storage and access overhead. Such methods avoid repeatedly constructing the full video representation, but evidence retrieval and answer generation remain per-question. As a result, they neither prevent repeated use of redundant evidence nor decide which questions should be carried out jointly.

Batched Model Execution. Recent works improve efficiency by processing multiple inputs or questions together rather than handling each independently. During training, multiple questions have been jointly exploited (Lei et al., 2020; Pei et al., 2025; Chen et al., 2024b), primarily to improve representation learning or training efficiency. At inference time, Batch Prompting (Cheng et al., 2023) packs multiple inputs into one prompt while benchmarks such as MMBench-Video (Fang et al., 2024) place all video questions into a structured prompt. These approaches operate on fixed batches and shared contexts, without deciding which questions should be executed together or how visual evidence should be allocated across different groups. In contrast, `QShare` dynamically determines both through evidence-aware question grouping and group-specific frame allocation.

3 Method

3.1 Problem Formulation and Framework Overview

Let $\mathcal{F} = \{f_1, \ldots, f_M\}$ denote the set of candidate frames sampled from the video and $Q = \{q_1, \ldots, q_N\}$ denote the set of user questions. The LMM's vision encoder represents each frame using c visual tokens; therefore, for any frame subset $S \subseteq \mathcal{F}$, the token cost is $C(S) = c\,|S|$, and B is the overall visual-token budget for the batch. We refer to the frames required to answer a question as its evidence, and we call the set of frames provided to a single LMM invocation a visual context. `QShare` divides Q into $\mathcal{P} = \{G_1, \ldots, G_K\}$, where K may vary, assigns each group a visual context $S_j \subseteq \mathcal{F}$, and answers every question in G_j with one call: $\{a_i : q_i \in G_j\} = \mathrm{LMM}(G_j, S_j)$. Let $Acc(q_i, G_j, S_j)$ be the resulting answer quality for q_i. The objective is:

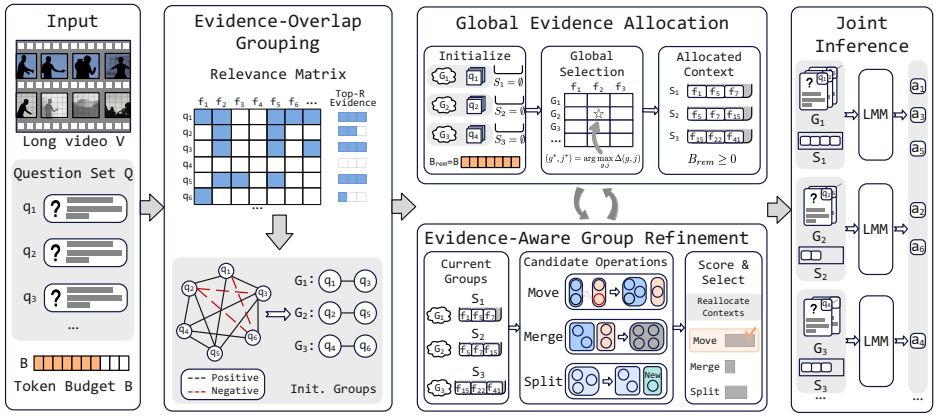

Figure 2: **Overview of `QShare`.** Given a question batch and global visual-token budget, `QShare` groups questions by overlapping relevant frames, greedily allocates frames to maximize marginal evidence coverage, then refines the partition by moving, merging, or splitting questions with reallocated contexts. Each final group is answered in one LMM call over its shared frames.

$$\max_{K,\,\mathcal{P},\,S_1,\ldots,S_K} \sum_{j=1}^{K} \sum_{q_i \in G_j} Acc(q_i, G_j, S_j) \quad \text{s.t.} \quad \sum_{j=1}^{K} C(S_j) \le B. \tag{1}$$

Equation 1 couples a discrete partitioning decision with budget-constrained context allocation. To handle this, we propose `QShare`, which operates in three stages (Figure 2): it first estimates question–frame relevance and constructs an initial partition using the overlap among the questions' highest-ranked frames (Section 3.2); it then assigns frames to groups based on marginal coverage given the shared budget (Section 3.3); and finally, it iteratively refines the partition and reassigns frames until no local adjustment improves the objective (Section 3.4). Each resulting group is then answered with a single LMM call.

3.2 Evidence-Overlap Grouping

We first sample candidate frames $\mathcal{F}$ at a fixed rate. We encode each question q_i and candidate frame f_j using the text and image encoders of SigLIP (Zhai et al., 2023), respectively, and compute their cosine similarity to estimate question–frame relevance r_{ij}. The resulting relevance vector $\mathbf{r}_i = [r_{i1}, \ldots, r_{iM}]$ is used for both question grouping and group-level frame allocation.

Specifically, for each question q_i, we treat its top-R frames $E_i = \mathrm{TopR}_{f_j \in \mathcal{F}}\, r_{ij}$ as its evidence set, and quantify the evidence overlap between two questions via $\mathrm{IoU}(q_i, q_h) = \frac{|E_i \cap E_h|}{|E_i \cup E_h|}$. We construct a graph with one node per question and edge weights $w_{ih} = \mathrm{IoU}(q_i, q_h) - \tau$, where τ is an overlap threshold: positive weights encourage grouping q_i and q_h, while negative weights encourage separating them. We obtain an initial partition $\mathcal{P}_0$ by approximately solving correlation clustering (Bansal et al., 2004) on this graph. Since $\mathcal{P}_0$ depends only on the top-R frames and not on the budget, we further refine it after allocation (Section 3.4).

3.3 Global Evidence Allocation

Given the initial question groups $\mathcal{P}_0 = \{G_1, G_2, \ldots\}$, we next allocate frames to the groups under the shared token budget B. Assigning an equal share of the budget to every group is unsuitable because different groups contain different numbers of questions and require different amounts of evidence. We therefore treat assigning a frame to a group as a candidate allocation operation and greedily select operations across all groups.

Evidence Coverage. Let S_g denote the set of frames currently selected for group G_g. We define the evidence coverage of question q_i under S_g as $F_i(S_g) = \min\left(1, \sum_{f_j \in S_g} r_{ij}\right)$, where the question–frame relevance scores r_{ij} are normalized to $[0, 1]$. As more relevant frames are added, coverage increases but saturates once sufficient evidence has been collected. The cap at 1 models diminishing returns: it prevents already well-covered questions from accruing unbounded utility, encouraging

the budget to be spent on questions that remain under-covered. Therefore, when a frame f_j is added to group G_g, its marginal evidence gain is:

$$\Delta(g, j) = \sum_{q_i \in G_g} [F_i(S_g \cup \{f_j\}) - F_i(S_g)] \tag{2}$$

This formulation naturally captures evidence sharing within a group. If f_j is relevant to multiple questions in G_g, it contributes evidence to all of them while being prefilled only once in the group's context. At each step, we select the group–frame pair with the largest marginal gain and continue until the global visual-token budget is exhausted.

Global Greedy Allocation. We initialize $S_g = \emptyset$ for every group and set the remaining budget to $B_{\text{rem}} = B$. At each iteration, we evaluate all feasible group–frame pairs and select:

$$(g^*, j^*) = \arg\max_{(g,j)} \Delta(g, j) \tag{3}$$

We add f_{j^*} to S_{g^*} and decrease B_{rem} by c. A pair is feasible only if f_j is not already in S_g, adding one more frame keeps the group's visual context within the LMM context window limit, and $B_{\text{rem}} \geq c$. We repeat this procedure until the global budget is exhausted. Because allocation is group-specific, the same visual frame may be assigned to different groups, in which case each assignment consumes c tokens of the global budget. The resulting sets $\{S_1, \ldots, S_K\}$ form the group-specific visual contexts used by the subsequent refinement stage.

Approximation guarantee. For a fixed partition, allocation chooses at most $\lfloor B/c \rfloor$ group–frame pairs to maximize the total coverage $\sum_g \sum_{q_i \in G_g} F_i(S_g)$. Each F_i is a sum of non-negative relevance scores capped at 1, which is monotone and submodular, so the total coverage is monotone and submodular as well. Under a single cardinality constraint, greedy selection therefore attains at least $(1 - 1/e)$ of the optimal coverage (Nemhauser et al., 1978).

3.4 Evidence-Aware Group Refinement

The initial groups are constructed from the overlap between the top-R frame sets of individual questions. However, it does not account for the visual frames that are actually selected for each group under the shared token budget. We therefore refine the initial partition using the group-specific contexts produced by Section 3.3. The task objective in Eq. 1 is defined in terms of the post-hoc answer quality $Acc(q_i, G_g, S_g)$, which is only observed after an LMM call and is therefore unavailable while planning. We instead score a candidate plan during partitioning and allocation with a tractable surrogate based on evidence coverage, together with a penalty for interference in large joint executions, and only invoke the LMM once the plan is fixed. Specifically, given a partition $\mathcal{P}$ and the visual contexts $\mathcal{S} = \{S_g\}_{G_g \in \mathcal{P}}$ allocated to its groups, we score the plan $(\mathcal{P}, \mathcal{S})$ using

$$\mathcal{J}(\mathcal{P}, \mathcal{S}) = \sum_{G_g \in \mathcal{P}} \sum_{q_i \in G_g} F_i(S_g) - \lambda \sum_{G_g \in \mathcal{P}} \frac{|G_g|(|G_g| - 1)}{2} \tag{4}$$

The first term measures the total evidence coverage of the questions under their group-specific contexts. The second term penalizes large joint executions because answering too many questions at once can lose question-specific details. The parameter λ controls the trade-off between evidence sharing and the cost of interference in larger groups.

Starting from the initial partition, we consider three local operations: *move* a question to another existing group (removing the source group if it becomes empty); *split* a question out of a group of size at least two into a new singleton group, which lets a poorly matched question leave without enumerating all possible ways of splitting the original group; and *merge* two existing groups, reselecting the merged group's frames rather than taking their union so that shared frames are prefilled once. For every candidate partition $\mathcal{P}'$ we rerun the greedy allocation from empty contexts under the same budget to obtain its contexts $\mathcal{S}'$, reusing the precomputed relevance scores and requiring no video encoding or LMM inference, and record $\Delta\mathcal{J}(\mathcal{P}') = \mathcal{J}(\mathcal{P}', \mathcal{S}') - \mathcal{J}(\mathcal{P}, \mathcal{S})$ together with the corresponding $\mathcal{S}'$. Candidates that exceed the LMM context window are discarded; among the rest, we apply the operation with the largest positive $\Delta\mathcal{J}$, replacing $(\mathcal{P}, \mathcal{S})$ with its $(\mathcal{P}', \mathcal{S}')$, and repeat until no candidate remains or no candidate improves $\mathcal{J}$.

Algorithm 1 `QShare`: Evidence-Aware Grouped Inference

Input: Questions Q, candidate frames $\mathcal{F}$, per-frame token cost c, budget B, evidence size R, overlap threshold τ, trade-off λ
Output: Answers $\mathcal{A}$

$\mathbf{r} \leftarrow \textsc{Relevance}(Q, \mathcal{F})$ ▷ SigLIP similarity, Sec. 3.2
$\{E_i\} \leftarrow \textsc{TopR}(\mathbf{r}, R)$
$\mathcal{P} \leftarrow \textsc{CorrelationCluster}(\{\mathrm{IoU}(E_i, E_h) - \tau\})$ ▷ initial partition $\mathcal{P}_0$
$\mathcal{S} \leftarrow \textsc{GreedyAllocate}(\mathcal{P}, \mathbf{r}, B)$ ▷ Sec. 3.3
$\mathcal{P}, \mathcal{S} \leftarrow \textsc{Refine}(\mathcal{P}, \mathcal{S}, \mathbf{r}, B, \lambda)$ ▷ Sec. 3.4
$\mathcal{A} \leftarrow \emptyset$
for $G_g \in \mathcal{P}$ **do**
    $\mathcal{A} \leftarrow \mathcal{A} \cup \mathrm{LMM}(G_g, S_g)$ ▷ $S_g \in \mathcal{S}$
return $\mathcal{A}$

function $\textsc{GreedyAllocate}(\mathcal{P}, \mathbf{r}, B)$
    $S_g \leftarrow \emptyset$ for every $G_g \in \mathcal{P}$; $B_{\mathrm{rem}} \leftarrow B$
    while $B_{\mathrm{rem}} \geq c$ **and** $\exists$ feasible (g, j) **do**
        $(g^*, j^*) \leftarrow \arg\max_{(g,j)} \Delta(g, j)$ ▷ Eq. 3
        $S_{g^*} \leftarrow S_{g^*} \cup \{f_{j^*}\}$
        $B_{\mathrm{rem}} \leftarrow B_{\mathrm{rem}} - c$
    return $\mathcal{S} \leftarrow \{S_g\}_{G_g \in \mathcal{P}}$

function $\textsc{Refine}(\mathcal{P}, \mathcal{S}, \mathbf{r}, B, \lambda)$
    while true do
        $\mathcal{C} \leftarrow \textsc{Candidates}(\mathcal{P})$ ▷ move, split, merge; Sec. 3.4
        if $\mathcal{C} = \emptyset$ **then**
            break
        for $\mathcal{P}' \in \mathcal{C}$ **do**
            $\mathcal{S}[\mathcal{P}'] \leftarrow \textsc{GreedyAllocate}(\mathcal{P}', \mathbf{r}, B)$ ▷ store allocation per candidate
            $\Delta\mathcal{J}(\mathcal{P}') \leftarrow \mathcal{J}\big(\mathcal{P}', \mathcal{S}[\mathcal{P}']\big) - \mathcal{J}(\mathcal{P}, \mathcal{S})$
        $\mathcal{P}^* \leftarrow \arg\max_{\mathcal{P}' \in \mathcal{C}} \Delta\mathcal{J}(\mathcal{P}')$
        if $\Delta\mathcal{J}(\mathcal{P}^*) \leq 0$ **then**
            break
        $(\mathcal{P}, \mathcal{S}) \leftarrow \big(\mathcal{P}^*, \mathcal{S}[\mathcal{P}^*]\big)$
    return $\mathcal{P}, \mathcal{S}$

Complexity. Let $L = \lfloor B/c \rfloor$ be the number of frames. With current coverages $F_i(S_g)$, computing $\Delta(g, j)$ costs $O(|G_g|)$, so one greedy step evaluates all $K \times M$ pairs in $O(NM)$ time and allocation costs $O(LNM)$. One refinement round considers at most $N(K–1)$ moves, N splits, and $K(K–1)/2$ merges; re-allocating frames for each candidate yields $O\big((NK + K^2)\, LNM\big)$ per round. Since partitions are finite and $\mathcal{J}$ strictly increases on each accepted step, no partition repeats and the algorithm terminates. All terms are computed from the precomputed $N \times M$ relevance matrix via vectorized operations, with no LMM forward passes.

Joint Inference. For each final group G_g, `QShare` processes its visual context S_g once in temporal order and jointly answers all assigned questions in a single LMM call. The full prompts are provided in Appendix C.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate `QShare` on three video-understanding benchmarks with multiple questions per video. `CG-Bench` (Chen et al., 2025) includes 1,219 videos (10–80 min) and 12,129 questions (∼10/video). `LVBench` (Wang et al., 2025) includes 103 videos (avg. 68 min) and 1,549 questions across diverse domains (∼15/video). `CinePile` (Rawal et al., 2024) contains 9,396 videos and 303,828 question–answer pairs (∼32/video).

Evaluation Models. We use Qwen3-VL-8B-Instruct (Bai et al., 2025) and LLaVA-OneVision-7B (Li et al., 2024) as the backbones. Both models use the same question order and answer format.

Baselines. We organize the baselines into three groups. (1) Visual reduction methods. `VisionZip` (Yang et al., 2025a) constructs one query-agnostic visual context. `Q-Frame` (Zhang

Table 1: Comparison with existing baselines under the same global visual-token budget $B = 12{,}544$ per video batch. Qwen denotes Qwen3-VL-8B, and LLaVA denotes LLaVA-OneVision-7B.

Method	CG-Bench		LVBench		CinePile		Avg.
	Qwen3	LLaVA	Qwen	LLaVA	Qwen	LLaVA	
Our method variants							
Uniform Sampling	44.10	34.65	42.32	40.75	44.54	42.20	41.43
Fully Joint	48.25	39.04	42.95	41.42	45.92	43.47	43.51
Independent	45.61	37.28	39.81	40.44	43.57	37.13	40.64
Fixed-Size Grouping	45.18	31.58	36.36	35.11	44.92	35.78	38.16
Semantic Grouping	46.93	32.89	37.30	35.74	46.46	36.86	39.36
Random Grouping	43.42	32.02	32.60	35.42	45.01	35.41	37.31
Cross-query reuse baselines							
ReKV	38.60	29.82	35.74	36.05	34.42	40.21	35.81
SparseVILA	42.98	31.58	33.54	35.42	40.22	35.14	36.48
Flash-VStream	46.49	36.84	42.63	41.07	45.64	42.57	42.54
Visual-reduction baselines							
VisionZip	48.50	39.91	43.57	41.83	46.05	43.56	43.90
Q-Frame	39.91	31.14	34.80	34.48	44.02	35.68	36.67
PruneVid	39.47	30.26	35.97	36.68	42.48	37.77	37.11
Ours	**53.44**	**44.04**	**48.45**	**46.26**	**49.38**	**46.03**	**47.93**

et al., 2025b) independently selects question-relevant frames. `PruneVid` (Huang et al., 2025) combines query-agnostic video token merging with question-guided token pruning. (2) Cross-query reuse baselines. `ReKV` (Di et al., 2025) applies sliding-window attention over video frames to build and store a video-level KV cache, then, for each question, retrieves the cache segments most relevant to the query. `SparseVILA` (Khaki et al., 2025) prunes visual tokens in the prefill stage, keeps the remaining tokens as a reusable visual cache, and during decoding fetches a question-specific sparse subset from this cache. `Flash-VStream` (Zhang et al., 2025a) incrementally compresses the video into a fixed-size visual memory from which subsequent questions are answered. (3) Variants of our method. These variants share `QShare`'s relevance scoring, greedy allocation, and batch budget B, and differ only in how frames are selected or how questions are partitioned. `Uniform Sampling` is the query-agnostic extreme: it samples frames uniformly over the full video to fill the budget B and answers all questions in one LMM call over this single shared context. `Fully Joint` keeps the same one-call, one-context setting but selects frames with `QShare`'s relevance scores, treating the whole batch as a single group. `Independent` is the per-question extreme: each question forms its own group, the budget is distributed across questions by the same greedy allocation, and each question is answered in a separate LMM call. `Fixed-Size Grouping` uses preset group sizes. `Random Grouping` randomly partitions questions to match our group count and size distribution. `Semantic Grouping` clusters questions using only text embeddings.

Implementation Details. All methods share the same candidate frames (sampled at 1 FPS over the full video), answering prompt, decoding settings, and batch budget B=12,544 visual tokens; with c=196 tokens per frame for both backbones, this equals 64 frames per video batch. Every baseline is configured to consume exactly B: single-context methods build one context of B tokens, per-question methods split B across a video's questions, and cross-query reuse methods retrieve B tokens in total from a cache whose construction cost is reported separately (Appendix A). `QShare` uses R=16, τ=0.4, and λ=0.5; sensitivity to these values is reported in Appendix B. All experiments run on NVIDIA V100 40 GB GPUs, and all latencies are measured on a single GPU.

Evaluation Metrics. We report accuracy, total visual-token cost, the number of LMM calls, and end-to-end batch latency. Visual-token cost sums c over every frame in every LMM call; a frame selected by several groups is counted once per group, while a frame shared by questions within a group is counted once.

4.2 Comparison with Existing Baselines

Accuracy under a Fixed Token Budget. Table 1 compares methods under the same batch-level visual-token budget, $B = 12{,}544$. `QShare` obtains the best average accuracy, improving over the strongest baseline by 4.0 points on average. Across our variants, `Independent` assigns each question its own visual context, but it divides the token budget into N portions and repeats prefilling over

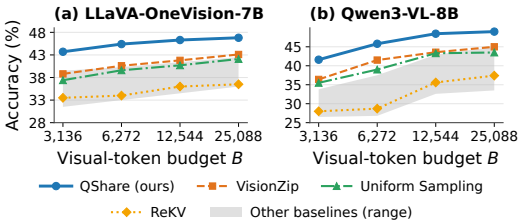

Figure 3: Accuracy across visual-token budgets. Average accuracy over LVBench as the batch budget B varies from $3,136$ to $25,088$ for both backbones.

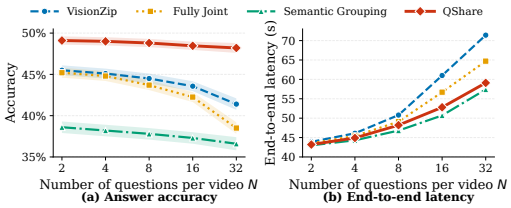

Figure 4: Accuracy and end-to-end latency at different question batch sizes on LVBench with Qwen3-VL-8B-Instruct. (a) Answer accuracy. (b) End-to-end latency from the raw video and question batch to the generation of all answers.

Table 2: Component ablation of QShare over CG-Bench, LVBench and CinePile under different global visual-token budgets.

Budget	Qwen				LLaVA			
	Per-Union	Proportional	w/o Refine	QShare	Per-Union	Proportional	w/o Refine	QShare
6,272	43.41	42.83	43.01	**46.58**	40.11	38.89	39.23	**43.22**
12,544	47.82	46.65	46.71	**50.42**	42.81	39.78	40.89	**45.44**
25,088	48.19	47.31	47.56	**51.20**	43.66	42.20	42.53	**47.03**

N separate calls. As a result, it is less effective under a strict token limit. Fully Joint instead uses the entire budget to create a single shared context in one call, but that context must accommodate all questions and cannot be customized for any particular one. Uniform Sampling lags behind since QShare relies on relevance-based frame selection rather than sampling uniformly. Fixed-Size Grouping and Random Grouping reduce some redundant computation, yet the questions grouped together may not actually share evidence. Semantic Grouping also falls short because similarity between question text does not consistently correspond to sharing visual evidence. The visual-reduction baselines exhibit a complementary pattern. VisionZip uses one visual context for the whole batch and loses question-specific contexts. Q-Frame and PruneVid retain query relevance but repeatedly prefill overlapping frames across questions. QShare combines selective sharing with group-specific evidence and therefore reaches higher accuracy at the same budget. For cross-query reuse, ReKV and SparseVILA amortize the construction of the cached video representation, and Flash-VStream keeps a compact visual memory across questions; however, they do not decide which questions should share a visual context. In contrast, QShare jointly uses the question batch to choose the sharing pattern and allocate frames to each group, reducing redundant processing without forcing unrelated questions to share one context. Appendix D shows QShare's actual groupings and frame allocations on three real benchmark videos.

Performance Across Token Budgets. Figure 3 sweeps the batch budget from $3,136$ to $25,088$ for both backbones. QShare delivers the highest accuracy across all budgets because it simultaneously determines which questions can share a visual context and how many frames to assign to each group under the overall budget, promoting reuse among related queries while retaining evidence needed for each question. This benefit is strongest at small budgets, where per-question baselines spend tokens on repeated computation and single-context baselines fail to capture question-specific details; as the budget grows, performance rises for all methods, but QShare remains consistently ahead.

4.3 Ablation Study

We evaluate the contributions of QShare's key components by ablating global budget allocation, group-level evidence selection, and evidence-aware refinement. Specifically, we compare QShare against two variants that keep its final partition and total budget fixed. Proportional allocates frames to each group in proportion to its question count and then greedily selects frames using QShare's group-level coverage. Per-Union keeps QShare's per-group frame counts, but selects frames per question by relevance, takes the union of the top frames across questions, and fills any shortfall with the next-best frames. We also compare QShare with No-Refine, which disables refinement and treats the initial partition as final after allocating group contexts.

As shown in Table 2, all three variants reduce accuracy relative to the full QShare. Proportional performs worse because a group's evidence demand is not necessarily propor-

Table 3: End-to-end latency (s) breakdown under the same global visual-token budget (B=12,544) for Qwen3-VL-8B-Instruct on LVBench. All latency is measured from the raw video and question batch to the generation of all answers.

Method	Calls	Video Process	Plan	LMM Inference	E2E Latency
Independent	15.0	40.9 ± 0.03	4.3 ± 0.01	6.5 ± 0.04	51.7 ± 0.08
Fully Joint	1.0	40.9 ± 0.03	4.3 ± 0.01	11.5 ± 0.04	56.7 ± 0.08
ReKV	15.0	40.9 ± 0.03	6.8 ± 0.03	6.5 ± 0.04	54.2 ± 0.1
VisionZip	1.0	40.9 ± 0.03	8.2 ± 0.02	11.9 ± 0.04	61.0 ± 0.09
Semantic Grouping	6.4	40.9 ± 0.03	4.3 ± 0.01	6.5 ± 0.04	51.7 ± 0.08
Separate execution	15.0	40.9 ± 0.03	7.4 ± 0.01	17.5 ± 0.04	65.8 ± 0.08
QShare	6.4	40.9 ± 0.03	7.4 ± 0.02	6.5 ± 0.04	54.8 ± 0.09

tional to its number of questions, while Per-Union cannot explicitly prioritize evidence shared across questions. Disabling refinement causes a further accuracy drop, indicating that the initial overlap-based partition does not fully capture the group-specific evidence that can be supported under the allocated budget. Overall, these results validate the complementary benefits of global evidence allocation, group-level evidence selection, and evidence-aware group refinement.

4.4 End-to-End Efficiency and Scalability

End-to-end efficiency. We evaluate QShare's end-to-end latency and provide a breakdown of its components under the same overall visual-token budget. Table 3 compares QShare with representative per-query, fully joint, and cross-query reuse baselines, as well as a Separate Execution variant that keeps the same groups and visual contexts as QShare but answers each question independently. QShare completes each video batch in 54.8 s, comparable to ReKV (54.2 s) and faster than Fully Joint (56.7 s) and VisionZip (61.0 s), while retaining the highest accuracy reported in Table 1. Its additional planning stages (including question–frame relevance estimation, evidence-aware grouping, global evidence allocation, and group refinement) account for only 13.5% of the total latency. Compared with Separate Execution, QShare reduces the average number of answer-generation calls from 15.0 to 6.4, lowers LMM inference latency by 62.9% (17.5 s to 6.5 s), and reduces end-to-end latency by 16.7% (65.8 s to 54.8 s). These results demonstrate the efficiency benefit of sharing visual prefilling within each group.

Scalability with question batch size. We evaluate how QShare scales with the question batch size N on LVBench with Qwen3-VL-8B-Instruct, for the same video. We use $b = 4$ frames per question to set the total visual-token budget as $B = Nbc$, where $c = 196$ is the visual-token cost per frame. We vary $N \in \{2, 4, 8, 16, 32\}$, giving total budgets of $\{8, 16, 32, 64, 128\}$ frames. For each video, we use the same pool of questions for all batch sizes. We randomly shuffle the questions and take the first N, so each smaller batch is included in the larger batches: $Q_2 \subset Q_4 \subset Q_8 \subset Q_{16} \subset Q_{32}$. We repeat this process five times with different shuffles and report the average results.

Figure 4(a) shows that QShare achieves the highest accuracy at every batch size. Its accuracy decreases by only about one percentage point as N increases from 2 to 32. In comparison, VisionZip and Fully Joint show larger drops. Semantic Grouping changes only slightly, but its accuracy remains much lower than that of QShare. These results show that QShare keeps high answer accuracy as the question batch grows. Figure 4(b) shows that end-to-end latency increases with N for all methods. QShare has lower latency than VisionZip and Fully Joint across all batch sizes. Semantic Grouping is slightly faster than QShare, but its accuracy is much lower. Overall, QShare provides a better balance between accuracy and latency as the question batch grows.

5 Conclusion

We propose QShare for batch VideoQA under a shared visual-token budget. Instead of using one context per question or per batch, QShare groups questions by evidence overlap, allocates a visual context to each group, iteratively refines groups, and answers each group jointly. This enables selective sharing: related questions share one visual context, while questions with distinct evidence are kept separate. Extensive experiments on three benchmarks and two backbones show that QShare improves average accuracy by 4.0 points over the strongest baseline under the same batch budget, and the gain persists across budget levels.

AI USE STATEMENT

In this work, we used generative AI tools only to improve the readability of this paper, for example, by correcting grammar and enhancing the clarity of sentences we had already written. We did not use generative AI tools for any activity that requires disclosure under the ICLR 2027 AI Policy for Authors: they were not used to create synthetic data, develop theoretical models or conceptual frameworks, state or prove mathematical claims, propose or refine hypotheses, design the methodology or experiments or provide feedback on them, implement methods, clean or reformat data, analyze or interpret results, or support translation. They were also not used for any other tasks for which disclosure is recommended, such as literature searches, writing code, creating figures, or drafting content. All AI-assisted edits were reviewed by the authors, and we accept full responsibility for the final work, including the text, claims, and any artifacts produced with the help of generative AI.

REPRODUCIBILITY STATEMENT

All datasets used in our experiments (`CG-Bench`, `LVBench`, and `CinePile`) are publicly available benchmarks, and both backbones (Qwen3-VL-8B-Instruct and LLaVA-OneVision-7B) are publicly released models. Section 4.1 describes the benchmarks, backbones, baselines, and evaluation metrics used in our experiments. Code is available at `https://anonymous.4open.science/r/QShare-repo`.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine learning*, 56(1): 89–113, 2004.

Guo Chen, Yicheng Liu, Yifei Huang, Baoqi Pei, Jilan Xu, Yuping He, Tong Lu, Yali Wang, and Limin Wang. Cg-bench: Clue-grounded question answering benchmark for long video understanding. In *International Conference on Learning Representations*, volume 2025, pp. 45647–45682, 2025.

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

Qishen Chen, Minjie Bian, and Huahu Xu. Mmql: Multi-question learning for medical visual question answering. In *International conference on medical image computing and computer-assisted intervention*, pp. 480–489. Springer, 2024b.

Zhoujun Cheng, Jungo Kasai, and Tao Yu. Batch prompting: Efficient inference with large language model apis. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 792–810, 2023.

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

Xinyu Fang, Kangrui Mao, Haodong Duan, Xiangyu Zhao, Yining Li, Dahua Lin, and Kai Chen. Mmbench-video: A long-form multi-shot benchmark for holistic video understanding. *Advances in Neural Information Processing Systems*, 37:89098–89124, 2024.

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

Samir Khaki, Junxian Guo, Jiaming Tang, Shang Yang, Yukang Chen, Konstantinos N Plataniotis, Yao Lu, Song Han, and Zhijian Liu. Sparsevila: Decoupling visual sparsity for efficient vlm inference. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 23784–23794. IEEE, 2025.

Chenyi Lei, Lei Wu, Dong Liu, Zhao Li, Guoxin Wang, Haihong Tang, and Houqiang Li. Multi-question learning for visual question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 11328–11335, 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.

Yudong Liu, Jingwei Sun, Yueqian Lin, Jianyi Zhang, Jingyang Zhang, Ming Yin, Qinsi Wang, Hai Li, and Yiran Chen. Keyframe-oriented vision token pruning: Enhancing efficiency of large vision language models on long-form video processing. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20802–20811. IEEE, 2025.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

Baoqi Pei, Yifei Huang, Guo Chen, Jilan Xu, Yali Wang, Limin Wang, Tong Lu, Yu Qiao, and Fei Wu. Guiding audio-visual question answering with collective question reasoning. *International Journal of Computer Vision*, 133(10):6912–6929, 2025.

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

Jihao Qiu, Lingxi Xie, Xinyue Huo, Qi Tian, and Qixiang Ye. Longvideo-r1: Smart navigation for low-cost long video understanding. *arXiv preprint arXiv:2602.20913*, 2026.

Ruchit Rawal, Khalid Saifullah, Miquel Farré, Ronen Basri, David Jacobs, Gowthami Somepalli, and Tom Goldstein. Cinepile: A long video question answering dataset and benchmark. *arXiv preprint arXiv:2405.08813*, 2024.

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

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

Yan Shu, Zheng Liu, Peitian Zhang, Minghao Qin, Junjie Zhou, Zhengyang Liang, Tiejun Huang, and Bo Zhao. Video-xl: Extra-long vision language model for hour-scale video understanding. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26160–26169. 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.

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

Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. Videoagent: Long-form video understanding with large language model as agent. In *European Conference on Computer Vision*, pp. 58–76. Springer, 2024.

Junbin Xiao, Jiajun Chen, Tianxiang Sun, Xun Yang, and Angela Yao. Mukv: Multi-grained kv cache compression for long streaming video question-answering. *arXiv preprint arXiv:2605.22269*, 2026.

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

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*, 2025b.

Zuhao Yang, Sudong Wang, Kaichen Zhang, Keming Wu, Sicong Leng, Yifan Zhang, Bo Li, Chengwei Qin, Shijian Lu, Xingxuan Li, et al. Longvt: Incentivizing" thinking with long videos" via native tool calling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 33816–33826, 2026.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11941–11952. IEEE, 2023.

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

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*, pp. 22056–22065, 2025b.

A Baseline Configuration and Cost Accounting

All methods operate on the same candidate frames, sampled uniformly at 1 FPS over the full video and encoded once by the backbone's vision encoder. Each frame contributes c=196 visual tokens, and every method consumes exactly B=12,544 tokens per video batch. Table 4 summarizes how each method spends this budget.

Single-context methods. `VisionZip` and `Uniform Sampling` build one visual context of B tokens and answer all N questions of a video in a single LMM call. `VisionZip` selects tokens with its own query-agnostic saliency criterion; `Uniform Sampling` takes 64 uniformly sampled frames. `Fully Joint` uses the same single-call setting but selects frames with `QShare`'s relevance scores.

Per-question methods. `Q-Frame` and `PruneVid` keep their native per-question execution. Each question receives a context of $\lfloor B/N \rfloor$ tokens, rounded down to whole frames for `Q-Frame`, so the N calls sum to at most B. `Independent` distributes B across questions with `QShare`'s greedy allocation instead of an equal split.

Cross-query reuse methods. `ReKV` and `SparseVILA` build their cached video representation once over all candidate frames. For each question they retrieve $\lfloor B/N \rfloor$ tokens from the cache, so the total retrieved equals B. Building the cache requires a full LMM forward pass over the video; this cost is excluded from B. `Flash-VStream` is configured analogously, with its visual memory sized to B/N tokens per question.

Grouping variants. `Fixed-Size Grouping`, `Random Grouping`, `Semantic Grouping`, and `QShare` all produce K groups, allocate frames to groups with the same greedy procedure under the shared budget B, and issue one LMM call per group.

Hardware. All experiments run on a server with four NVIDIA V100 40 GB GPUs. Each video batch is processed end-to-end on a single GPU; the four GPUs are used only to parallelize across videos, so all reported latencies correspond to single-GPU execution.

Table 4: How each method spends the batch budget B. N is the number of questions per video and K the number of groups produced by the method.

Method	Context	Calls	Tokens / call
`Uniform Sampling` `Fully Joint` `VisionZip`	one shared	1	B
`Q-Frame`, `PruneVid`	one per question	N	$\lfloor B/N \rfloor$
`Independent`	one per question	N	greedy, $\sum = B$
`ReKV`, `SparseVILA`	one per question	N	$\lfloor B/N \rfloor$
`Flash-VStream`	one per question	N	B/N
`Random Grouping` `Fixed-Size Grouping` `Semantic Grouping` `QShare`	one per group	K	greedy, $\sum = B$

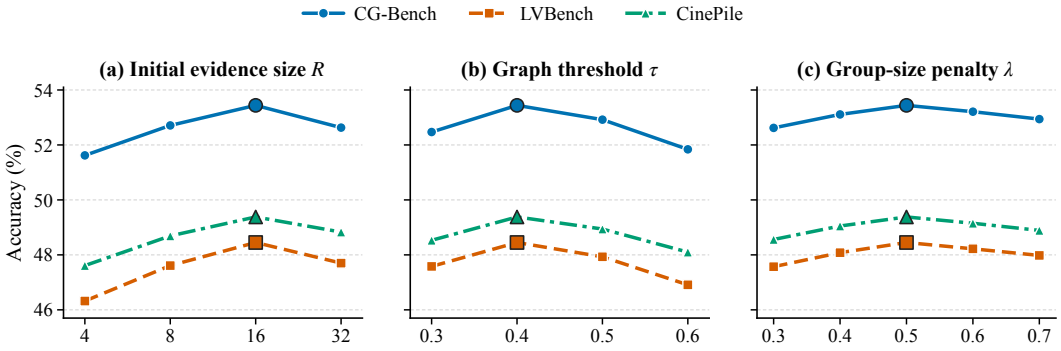

Figure 5: **Hyperparameter sensitivity of `QShare`.** Results on three benchmarks using Qwen3-VL-8B-Instruct under $B = 12{,}544$. We vary one hyperparameter at a time while fixing the others to $R = 16$, $\tau = 0.4$, and $\lambda = 0.5$.

B Hyperparameter Sensitivity

Initial evidence size R. We study initial evidence sizes $R \in \{4, 8, 16, 32\}$ (Figure 5). A small R may omit frames needed to reliably estimate evidence overlap, while an excessively large R lets the top-R sets of unrelated questions overlap on weakly relevant frames, blurring the question graph and producing overly broad groups. We therefore use $R = 16$ in all other experiments.

Question-graph threshold τ. We evaluate graph thresholds $\tau \in \{0.3, 0.4, 0.5, 0.6\}$ (Figure 5). A low threshold connects questions with weak evidence overlap, producing overly broad groups and unnecessarily large contexts, while a high threshold fragments questions that could share evidence, increasing the number of groups and repeated prefills. The intermediate value $\tau = 0.4$ gives the best accuracy–efficiency trade-off and is used as the default.

Group-size penalty λ. We study group-size penalties $\lambda \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$ (Figure 5). A small λ favors large groups, increasing reuse but risking incompatible evidence and cross-question interference, while a large λ yields many small groups and reduces sharing. The default $\lambda = 0.5$ best balances these trade-offs, preserving most token/latency savings without hurting accuracy.

C Prompts

Group prompt. For each final group G_g, `QShare` makes one LMM call. Its input consists of the visual context S_g and a text prompt containing all questions assigned to G_g. The selected frames are placed in their original temporal order. Each frame is preceded by its timestamp in seconds, allowing the model to identify the time gaps between frames that are not adjacent in the video. We use the same timestamp format for both backbones.

Questions follow their original benchmark order and are labeled `[Q1]`, `[Q2]`, and so on. The prompt instructs the model to treat each question independently and return one option for each question in

the same order. All questions share the same visual prefix, which is prefilled once, and their answers are produced in a single generation pass. Therefore, the visual tokens of a frame used by several questions appear only once in the group prefix. The prompt template is:

```
You are given frames sampled from a long video. Each frame is preceded
by its timestamp in seconds.

Answer each multiple-choice question using only the video content.
Treat each question independently and select exactly one option
(A, B, C, or D) for each question.

[Q1] Question: <question 1>
    Options: (A) <...>  (B) <...>  (C) <...>  (D) <...>
[Q2] Question: <question 2>
    Options: (A) <...>  (B) <...>  (C) <...>  (D) <...>
...

Return exactly one answer per question, one per line, and in the same
order as the questions. Copy each question identifier exactly and do
not include explanations or any other text.

[Q1]: <option>
[Q2]: <option>
...
```

Single-question prompt. For `Independent`, `Separate Execution`, and all per-question baselines, we use the same template with only one question. The instruction, option order, and output format remain unchanged; only the number of questions in the prompt differs. For baselines with their own visual-processing pipelines, including `ReKV`, `SparseVILA`, and `Flash-VStream`, we keep their original visual processing and use the same single-question text and output format.

D Case Studies

We present three real-video case studies to illustrate `QShare`'s grouping and frame allocation. For each case, we run the full `QShare` pipeline on a source video and its associated question batch, and then report (i) the inferred question group and (ii) the shared temporal context selected by `QShare`. Where available, we additionally include the benchmark's released temporal evidence annotations as an external reference for what the shared context should cover. These examples show on real videos that `QShare` can identify shared temporal context across related questions and allocate frames jointly, consistent with the accuracy gains in Table 1. In particular, `LVBench` and `CG-Bench` provide temporal evidence annotations that allow a direct cross-check: in both cases, `QShare`'s selected shared context matches the annotated evidence interval. We therefore report the resulting frame count and compare it against what `Independent` would spend if it encoded the same evidence separately for each focal question. At 1 FPS with c=196 tokens per frame, an evidence interval of length L seconds corresponds to L frame assignments and $196L$ visual tokens. `CinePile` provides question and video identifiers but no temporal evidence labels; for this dataset we instead report `QShare`'s selected frames together with manually verified scene phases, without an external annotation to compute savings against.

D.1 Selective Sharing on LVBench

Figure 6 shows a 47:26 tennis video. Questions 3961 (who served first), 3962 (who scored the first point), and 3963 (why the start was considered poor) have different answer targets, but they share the same official evidence interval (10:58–11:16). The corresponding labels are Stefanos Tsitsipas, Novak Djokovic, and a Tsitsipas service error. `QShare` groups these three questions and assigns them a shared 18-frame context that covers this interval. If executed independently, the system would encode the same 18-second window three times (once per question), requiring 54 frame assignments; in contrast, `QShare` uses 18 frames in total, saving 36 assignments (7,056 visual tokens), i.e., 56.25% of a 64-frame batch budget. `QShare` then reallocates the saved capacity to questions with temporally distant evidence, e.g., question 3966 (33:28–34:51), instead of mixing unrelated frames into the tennis-point context. Overall, this case illustrates both aspects of the design

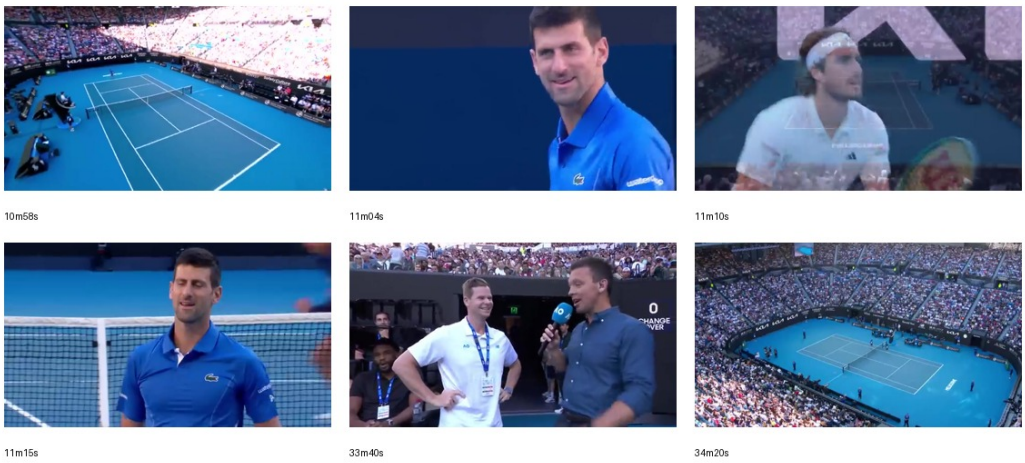

Figure 6: Real LVBench evidence. The first four frames depict QShare's actual shared context for questions 3961–3963, which fall within the shared 10:58–11:16 evidence window; the final two frames show the more distant interview portion assigned to question 3966's separate group. The timestamps correspond to times in the source video. These frames indicate what evidence is included in QShare's context, not which frames the model internally focused on or any causal account of how it produced its answer.

on a single video: nearby questions share one prefix, while distant questions are kept in separate groups.

D.2 Shared Local Evidence and Distributed Evidence in CG-Bench

Figure 7 shows a CG-Bench video (23:49) recorded from a first-person view during airport ground operations. Questions 544 and 2179 ask, respectively, where the aircraft is relative to the vehicle and the direction it is heading. Both questions share the label "directly ahead" and the same official clue interval (919.71–939.61 s). QShare therefore groups them and uses a shared context of about 20 frames for this interval, which avoids assigning roughly 20 duplicate frames (about 3,920 tokens) compared with encoding the interval separately for each question. In contrast, other questions in the same batch require different evidence and are handled separately: question 954 uses 779.32–827.96 s, question 4128 uses 986.03–1012.56 s, and question 2804 counts five exits from the vehicle across five widely separated intervals. If one universal context were used for the whole batch, it would need to cover all these events, and the aircraft-approach segment would be duplicated for questions 544 and 2179 under independent execution. QShare's grouping instead reuses the local evidence for those two questions while reserving separate context capacity for the dispersed counting question.

D.3 High-Question-Count Sharing on CinePile

Figure 8 presents the video clip in CinePile. The 2:24 night-time sequence is accompanied by 32 questions that follow a clear four-phase structure: (i) inspecting a broken sensor, (ii) reacting to an approaching helicopter, (iii) staying hidden during the spotlight and snake sequence, and (iv) resuming movement afterward. QShare's grouping follows these phases: within each phase, questions target different attributes, actions, temporal order, and character relations, so they are grouped to share the same visual evidence instead of using one undifferentiated context for all questions. This example also highlights the high-question-count regime: with a 64-frame batch budget, an equal independent split would allocate only two frames per question, whereas QShare's group-level sharing allows many questions to benefit from a denser shared prefix. Since CinePile does not provide temporal evidence intervals, we cannot compute token savings against external annotations (unlike LVBench and CG-Bench); instead, we report QShare's selected frames together with manually verified phase timestamps.

Overall, these three cases show why a shared budget needs careful grouping, based on QShare's actual grouping and allocation on these videos. In LVBench, three questions refer to the same

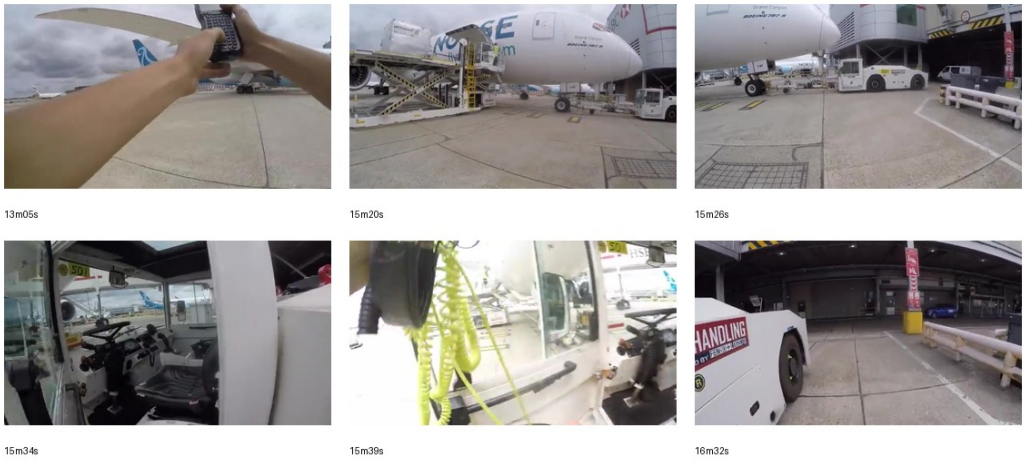

Figure 7: Real CG-Bench evidence. Frames at 15:20–15:39 are part of QShare's shared context for questions 544 and 2179 and lie in the common annotated clue interval; 13:05 and 16:32 illustrate distinct evidence regions allocated to other questions in the same batch.

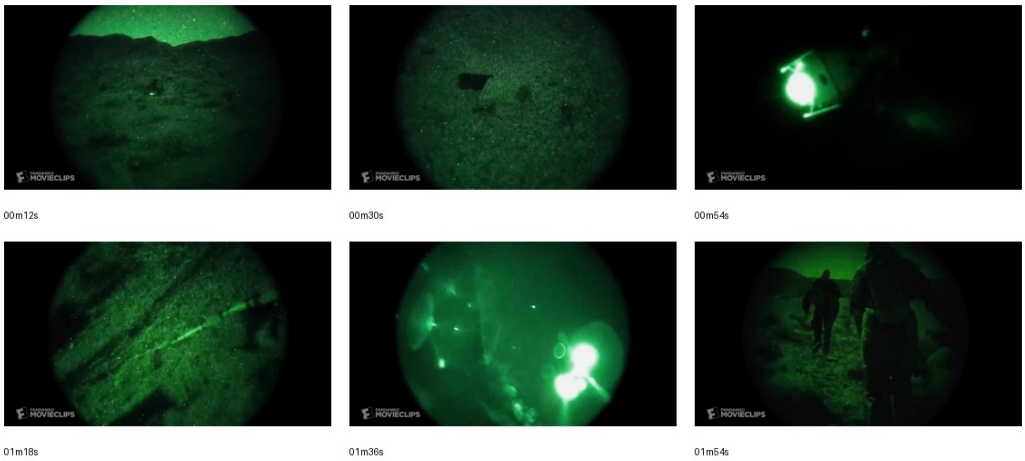

Figure 8: Real frames from a CinePile clip. The panel covers the sensor, helicopter/spotlight, and aftermath phases and visualizes the frames QShare selects for this batch. Because CinePile provides no temporal evidence annotations, the phase boundaries are shown only for illustration and were manually verified.

short moment, so QShare lets them share one context, while a later question is put in its own group. In CG-Bench, QShare shares a small local clue between two questions, but keeps another question separate because its evidence is spread across the video. In CinePile, the batch has many questions, so splitting the budget per question would leave too few frames; grouping lets related questions share a denser prefix. This matches the improvements over Independent and Fully Joint in Table 1: sharing reduces repeated prefilling, but QShare still keeps different evidence in different groups instead of forcing one context for all questions.