FORTE: ADAPTIVE SCORING AND EXACT KEYFRAME SELECTION FOR LONG-VIDEO QUESTION ANSWERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Query-aware keyframe selection enables multimodal large language models (MLLMs) to process long videos using only a small set of question-relevant frames. Existing score-based methods, however, typically search within a fixed, uniformly sampled candidate pool, preventing evidence outside this pool from ever being selected. Given a limited relevance-scoring budget, the key challenge is to allocate evaluations adaptively to promising frames while continuing to explore underrepresented temporal regions. We introduce FORTE, a training-free framework that addresses this challenge through two stages: adaptive relevance scoring and global keyframe optimization. Starting from sparse, uniformly distributed observations, our efficient Gaussian-process relevance predictor estimates relevance for unscored frames, exploiting temporal locality and the approximately banded kernel structure to reduce the core computation from cubic to linear time in the number of frames for fixed bandwidth. The scoring stage then selects which frames to score next by balancing predicted relevance with temporal coverage, prioritizing promising regions while also exploring less-represented parts of the video. The optimization stage selects the final keyframes by maximizing an objective that jointly captures measured relevance and temporal coverage. We derive an exact algorithm that leverages the logarithmic coverage structure to identify the optimal subset of the scored candidate pool in time linear in the pool size, for a fixed final-frame budget. Experiments on four long-video question-answering benchmarks show that FORTE achieves the highest observed mean accuracy among the compared selectors under every tested scoring budget. Further evaluations demonstrate its consistent effectiveness across different relevance scorers and downstream MLLMs.

1 INTRODUCTION

Long-video understanding requires locating question-relevant evidence and integrating information across extended temporal contexts. For multimodal large language models (MLLMs), directly encoding large numbers of frames results in excessively long visual-token sequences, incurring prohibitive memory and computational costs and potentially exceeding the model's context window. Existing work mitigates this challenge through long-context modeling, visual compression, and memory mechanisms (Song et al., 2024; He et al., 2024; Chen et al., 2025; Shen et al., 2025; Shu et al., 2025). In addition to these model-side approaches, *keyframe selection* approaches the problem through another dimension, by determining which video frames are highly relevant to the question and hence should be supplied to the MLLM in the first place, i.e., preserving question-relevant evidence of the video within a limited *frame budget*.

Training-free, query-aware keyframe selection typically uses pretrained vision–language models, such as CLIP or BLIP (Radford et al., 2021; Li et al., 2022), to estimate frame–question relevance without additional training or changes to the downstream MLLM. Existing methods commonly sample frames uniformly across the video or within temporal clips, evaluate their relevance to the question, and then determine which frames to provide to the MLLM, constrained by the frame budget, for example, by balancing relevance with temporal coverage or visual diversity (Tang et al., 2025; Tan et al., 2026; Sun et al., 2025; Zhang et al., 2025b; Zhu et al., 2026).

However, uniform sampling can miss question-relevant frames that appears between sampled timestamps, as illustrated in Figure 1(a). A straightforward way to alleviate this problem is to sample

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.

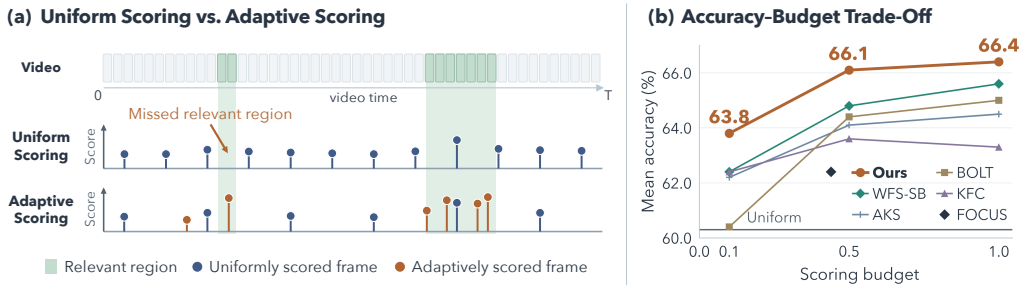


Figure 1: Budgeted keyframe selection for long-video understanding. (a) Adaptive scoring can capture relevant content missed by uniform scoring under the same evaluation budget. (b) Four-benchmark mean QA accuracy with Qwen3-VL-8B and 64 input frames. The scoring budget counts frame–query evaluations per second of video.

more densely, but this increases the number of frames that must be evaluated for relevance, leading to higher computation and memory costs. Therefore, an effective keyframe selection framework has to operate under another constraint, a *relevance-scoring budget*, which limits the number of frames to be evaluated.

This constraint motivates us to adaptively select which frames to be scored rather than fixing their locations uniformly, to meet the relevance-scoring budget while capturing the question relevant frames, as illustrated in Figure 1. Additionally, simply retaining the highest-scoring frames can concentrate the final input in a few regions, even if scoring has covered the video broadly. As such, a coherent keyframe selection framework must optimize both relevance-scoring and temporal frame coverage.

Motivated by these two design principles, we propose *FORTE*, an adaptive, training-free framework for keyframe optimization with relevance prediction and temporal coverage. Its two stages, *adaptive relevance scoring* and *global keyframe optimization*, are coordinated by a unified relevance–coverage objective (Figure 2). The first stage is an efficient Gaussian-process (GP) relevance predictor (Rasmussen & Williams, 2006) for unscored frames, exploiting temporal locality and the resulting approximately banded kernel structure to reduce the core GP computation from cubic to linear time. The predictor is updated as new scores are observed, and its estimates are combined with temporal coverage gains to choose additional frames for evaluation. Once the relevance-scoring budget is exhausted, the second stage optimizes the relevance–coverage objective over the final frame set using measured scores. By exploiting the logarithmic gap rewards and temporal ordering, we derive an exact algorithm whose runtime is linear with regard to the candidate-pool size under a fixed final-frame budget.

Experiments on four long-video QA benchmarks show FORTE achieves the highest observed mean accuracy among compared selectors across all tested relevance-scoring and final-frame budgets. Figure 1(b) shows diminishing accuracy gains as the relevance-scoring budget increases, highlighting the importance of allocating evaluations effectively rather than simply scoring more frames.

Our contributions are summarized as follows:

- We introduce FORTE, an adaptive, training-free framework that coordinates frame-level relevance scoring and final-frame selection under separate budgets. Its unified relevance–coverage objective guides feedback-driven scoring and enables efficient global optimization of the final frame set.
- We develop an efficient GP relevance predictor for unscored frames, exploiting temporal locality and the resulting approximately banded kernel structure to reduce the core GP computation from cubic to linear time.
- For global keyframe optimization, we derive an exact algorithm that uses a monotone upper envelope to solve the proposed objective in $\mathcal{O}(KN)$ time, where N is the scored-pool size and K is the final-frame budget.
- We demonstrate the framework's effectiveness on four long-video QA benchmarks across different relevance-scoring budgets, final-frame budgets, relevance scorers, and downstream models, and examine the trade-off between scoring cost and accuracy.

2 Related Work

2.1 Video LLMs for Long-Video Understanding

Video LLMs must retain useful temporal information while processing long visual sequences under finite context and computation limits. LongVILA extends the supported context through long-context training and efficient sequence parallelism (Chen et al., 2025). MovieChat and MA-LMM aggregate historical observations through memory mechanisms (Song et al., 2024; He et al., 2024), while LongVU and Video-XL compress visual representations (Shen et al., 2025; Shu et al., 2025). While these approaches improve how video information is represented and retained, keyframe selection addresses a complementary question: which frames should be supplied to the model in the first place?

2.2 Keyframe Selection Methods

Training-based methods learn frame-selection policies from supervision or feedback (Yu et al., 2023; 2025; Lee et al., 2026). We focus instead on training-free methods, particularly LLM-guided evidence search and lightweight score-based selection.

LLM-guided evidence search. These methods use language or multimodal reasoning to determine which evidence to inspect. VideoAgent coordinates iterative retrieval with vision–language tools (Wang et al., 2024); VideoTree organizes captioned frames into a query-adaptive hierarchy (Wang et al., 2025c); and A.I.R. combines lightweight candidate proposals with VLM-based analysis (Zou et al., 2026). Such workflows can exploit semantic judgments beyond frame–query similarity, but often incur substantial computational overhead from additional captioning, analysis, or reasoning calls during evidence selection.

Score-based frame selection. A lightweight alternative uses pretrained vision–language models, such as CLIP and BLIP (Radford et al., 2021; Li et al., 2022), to score frame–query relevance. These methods typically score frames sampled uniformly throughout the video and use the resulting relevance scores to select input frames for the MLLM. Beyond simply retaining the highest-scoring frames, BOLT (Liu et al., 2025) samples frames according to temporal relevance mass, Q-Frame (Zhang et al., 2025a) combines relevance-based sampling with resolution adaptation, and AdaQ (Zhang et al., 2026b) adjusts its sampling distribution based on score statistics. AKS (Tang et al., 2025) balances relevance with temporal coverage, while TCS (Tan et al., 2026) combines relevance-focused clips with sparse global context. FOCUS (Zhu et al., 2026) estimates clip-level mean relevance together with a confidence bound. However, these methods do not effectively estimate frame-level relevance while jointly leveraging temporal coverage to adaptively determine which frames should be scored. The concurrent preprint CSES (Zhang et al., 2026a) formulates keyframe selection as a 0–1 integer programming problem with a complex optimization objective. Rather than providing an exact polynomial-time solver, CSES employs a greedy approximation algorithm, which does not guarantee a globally optimal keyframe set with respect to its proposed objective.

In contrast, FORTE performs adaptive frame-level scoring by updating GP predictions between acquisition batches and balancing predicted relevance against marginal temporal coverage. Moreover, its logarithmic temporal-gap objective admits exact keyframe optimization in linear time via a monotone upper-envelope algorithm.

3 Method

3.1 Problem Formulation and Overview of FORTE

As illustrated in Figure 2(a), our framework performs query-aware keyframe selection to provide a downstream MLLM with a small set of input frames for long-video question answering.

Given a video spanning $[0, T]$ and a question q, let $\mathcal{U} \subseteq [0, T]$ denote the timestamps of all frames in the original video, with x_t denoting the frame at timestamp t. We use a pretrained scorer ψ to evaluate frame–question relevance and denote the normalized relevance score of x_t by $r_t \in [0, 1]$. We distinguish two budgets: the *scoring budget* B limits how many frames can be evaluated by ψ, whereas the *final frame budget* K specifies how many frames are supplied to the MLLM.

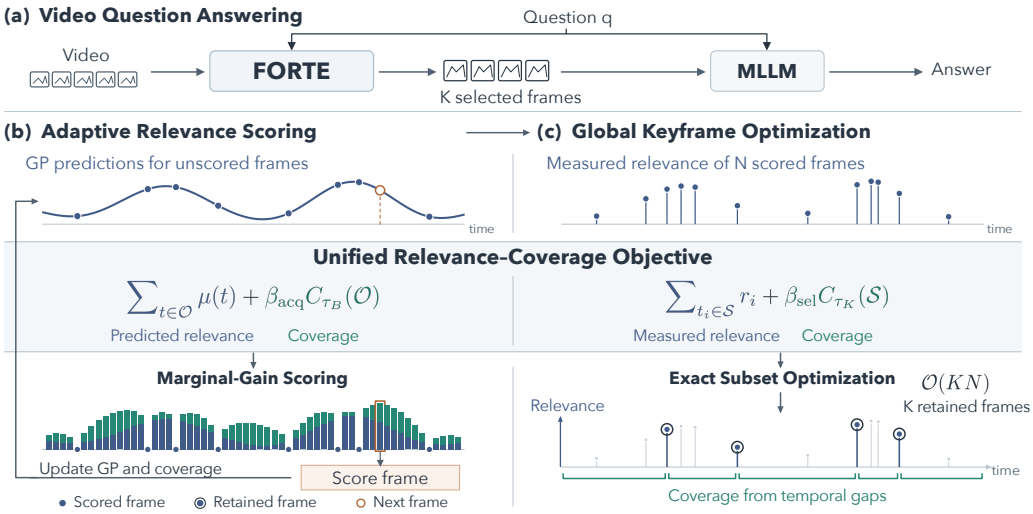


Figure 2: Overview of FORTE. (a) Under a scoring budget B, FORTE supplies K frames and the question to a frozen MLLM. Both stages balance summed relevance with temporal coverage: (b) adaptive scoring follows marginal gains under current GP predictions; (c) global keyframe optimization uses measured scores to select the optimal K-frame subset from N scored candidates.

The task involves two coupled decisions: which frames to score and which scored frames to retain. The first constructs a scored candidate set $\mathcal{O} \subseteq \mathcal{U}$ with $|\mathcal{O}| \leq B$; the second selects $\mathcal{S} \subseteq \mathcal{O}$ with $|\mathcal{S}| = K$. Candidate construction thus determines the frames available for final selection. We describe the case $|\mathcal{O}| \geq K$ below; Appendix C.2 specifies how shorter candidate pools are handled.

As shown in Figure 2, FORTE consists of two stages coordinated by a unified relevance–coverage objective. *Adaptive relevance scoring* begins by scoring a sparse set of uniformly spaced frames. It then predicts relevance for unscored frames and balances these predictions with temporal coverage to allocate the remaining scoring budget. *Global keyframe optimization* determines the final K frames by balancing measured relevance with coverage of the retained subset. Coverage thus guides both the construction of the scored candidate pool and the selection of the final input. The scorer and downstream MLLM remain frozen throughout this training-free process, while B and K can be specified independently.

3.2 Adaptive Relevance Scoring

Uniform initialization. To obtain initial relevance observations without favoring any temporal region, we score a sparse set of uniformly spaced frames. By default, we allocate $B/2$ evaluations to this initialization and reserve the remaining budget for adaptive scoring.

Predicting relevance for unscored frames. To decide where further scoring is useful, we need relevance estimates for frames that have not yet been evaluated. Temporal continuity provides a basis for these estimates: question-relevant content can persist across successive frames, allowing measured scores to inform predictions for nearby frames. We develop an efficient Gaussian-process (GP) relevance predictor (Rasmussen & Williams, 2006) that models this dependence and exploits the temporal kernel structure to accelerate prediction. It estimates relevance from the observed scores and timestamps without additional calls to the scorer.

Let $\mathcal{O}$ denote the timestamps of the frames scored so far and $\mathbf{r}$ the corresponding relevance-score vector. We model temporal dependence with a radial basis function (RBF) kernel matrix $\mathbf{K}$, whose pairwise entries for observed timestamps $t_i, t_j \in \mathcal{O}$ are

$$[\mathbf{K}]_{ij} = k(t_i, t_j) = \exp\left(-\frac{(t_i - t_j)^2}{2\ell^2}\right), \tag{1}$$

where ℓ sets the temporal correlation scale. For an unscored frame at timestamp t, the vector $\mathbf{k}(t)$ relates t to the observed timestamps through $[\mathbf{k}(t)]_i = k(t, t_i)$. The frame's relevance estimate is the

posterior mean of GP regression:

$$\mu(t) = \mu_{\text{obs}} + \mathbf{k}(t)^\top (\mathbf{K} + \epsilon \mathbf{I})^{-1} (\mathbf{r} - \mu_{\text{obs}} \mathbf{1}). \tag{2}$$

Here, $\mathbf{1}$ is the all-ones vector. With a zero-mean GP prior, predicted relevance tends toward zero in regions with no nearby scored frames, where $\mathbf{k}(t)$ is small. This would make unobserved regions appear irrelevant and could discourage further scoring there. We therefore set μ_{obs} to the mean of $\mathbf{r}$, allowing predictions in these regions to revert to the observed average. Nearby timestamps can produce very similar kernel rows, making $\mathbf{K}$ nearly singular. Thus, we add $\epsilon\mathbf{I}$ for numerical stability, where $\mathbf{I}$ is the identity matrix and $\epsilon > 0$ is a small diagonal adjustment.

Dense GP fitting requires cubic time in the number of observed frames. This cost can be reduced by exploiting the temporal locality induced by the kernel. The RBF kernel decays rapidly with temporal separation, giving the time-ordered kernel matrix an approximately banded structure. We exploit this structure by retaining a band of local interactions and restricting prediction to neighboring observations. For a fixed neighborhood width, the banded solve requires linear time and memory in the number of observations, and each prediction requires constant arithmetic work. Unless otherwise stated, GP prediction in our experiments uses this efficient approximation. Appendix B.1 provides the implementation and complexity details.

Balancing relevance and temporal coverage. Predicted relevance helps identify promising frames, but relying on it alone may concentrate evaluations in a small part of the video. The same issue arises when selecting the final input from measured scores. We therefore design the coverage term to serve both stages, with two requirements. First, at a fixed frame count, it should favor balanced temporal spacing. Second, the coverage gained by adding a frame should diminish as nearby frames accumulate, encouraging choices from less represented regions.

For an ordered timestamp set $\mathcal{S} = \{s_1 < \cdots < s_m\} \subseteq [0, T]$, let $s_0 = 0$ and $s_{m+1} = T$ denote the video boundaries. The temporal gaps are $g_i(\mathcal{S}) = s_{i+1} - s_i$ for $i = 0, \ldots, m$. We propose the following gap-based coverage objective:

$$C_\tau(\mathcal{S}) = \sum_{i=0}^{m} \log\left(1 + \frac{g_i(\mathcal{S})}{\tau}\right), \tag{3}$$

where $\tau > 0$ sets the reference time scale. The first property follows from the concavity of the logarithm, as formalized below.

Theorem 1 (Uniform-spacing optimality)**.** *Among all m-point subsets of $[0, T]$, C_τ is maximized when the $m + 1$ boundary-inclusive gaps are equal: $g_i = T/(m + 1)$.*

Although coverage alone is maximized by evenly spaced frames, adaptive scoring balances this preference with predicted relevance, allowing more evaluations in promising regions. For each candidate timestamp t, we evaluate the coverage gained by adding it to the current set, $\Delta C_\tau(t \mid \mathcal{S}) = C_\tau(\mathcal{S} \cup \{t\}) - C_\tau(\mathcal{S})$. This gain depends only on the gap containing t, so it can be computed without reevaluating coverage over the entire set. It satisfies the second property:

Theorem 2 (Coverage gains and diminishing returns)**.** *Given an interval $[0, T]$, a scale $\tau > 0$, and a finite timestamp set $\mathcal{U} \subseteq [0, T]$, adding $t \in \mathcal{U} \setminus \mathcal{S}$ to $\mathcal{S} \subseteq \mathcal{U}$ yields a nonnegative marginal coverage gain $\Delta C_\tau(t \mid \mathcal{S})$ that depends only on the gap containing t. For any $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{U}$ and $t \in \mathcal{U} \setminus \mathcal{B}$, the gains satisfy $\Delta C_\tau(t \mid \mathcal{A}) \geq \Delta C_\tau(t \mid \mathcal{B})$. Consequently, C_τ is monotone and submodular on $\mathcal{U}$.*

As neighboring frames are added, further relevance evaluations in the same region receive less coverage reward. The local expression for this gain and all theorem proofs are given in Appendix A.

Given the current observations, we choose the next frame by its estimated marginal contribution to the relevance–coverage objective, combining predicted relevance with the additional coverage it would provide:

$$t^\star = \arg\max_{t \in \mathcal{U} \setminus \mathcal{O}} \left[\mu(t) + \beta_{\text{acq}} \Delta C_{\tau_B}(t \mid \mathcal{O})\right], \tag{4}$$

where $\beta_{\text{acq}} \geq 0$ weights the coverage gain and $\tau_B = T/(B + 1)$ sets the reference spacing for B scored frames. We evaluate $\psi(x_{t^\star}, q)$, add $t^\star$ to $\mathcal{O}$, and incorporate its measured relevance into the GP. Before choosing another frame, we update the predictions and coverage gains for the remaining unscored frames. Each new observation thus informs the next scoring decision, while the coverage

reward decreases around newly scored frames. This select–score–update process, as illustrated in Figure 2(b), continues until the scoring budget is exhausted.

For computational efficiency, our experiments use batched scoring, with GP updates only between batches. Within each batch, frames are chosen sequentially using fixed GP predictions, but coverage gains are updated after every choice to discourage selecting nearby frames within the same batch. Each candidate's coverage gain depends only on its neighboring selected timestamps and can be computed using $\mathcal{O}(1)$ arithmetic operations. We then score the chosen frames in parallel through batched forward passes and update the GP with their measured relevance before constructing the next batch. The batch schedule and observation updates are detailed in Appendix B.

3.3 Global Keyframe Optimization

Even when the scored candidates are distributed across the video, retaining only the highest-scoring frames can concentrate the final input in a few temporal regions. We therefore optimize the unified relevance–coverage objective over the retained subset, balancing measured relevance with coverage of the frames actually supplied to the MLLM. Let $\mathcal{O} = \{t_1 < \cdots < t_N\}$ be the scored pool, and write $r_i = r_{t_i}$ for the measured relevance at t_i. We define the final selection objective F and maximize it over K-frame subsets:

$$F(\mathcal{S}) = \sum_{t_i \in \mathcal{S}} r_i + \beta_{\text{sel}} C_{\tau_K}(\mathcal{S}), \qquad \mathcal{S}^\star = \arg\max_{\mathcal{S} \subseteq \mathcal{O},\, |\mathcal{S}|=K} F(\mathcal{S}), \tag{5}$$

where $\beta_{\text{sel}} \geq 0$ balances relevance and coverage, and $\tau_K = T/(K+1)$ is the reference spacing for the final K frames. Both stages thus use an additive relevance term and the same coverage function: Equation 4 estimates the marginal gain from scoring another frame, while F evaluates the complete retained subset using measured scores.

Exact algorithm. The objective F decomposes into per-frame relevance and coverage rewards for consecutive temporal gaps. This structure allows us to optimize it exactly using dynamic programming, which extends optimal partial selections in temporal order. Let $D[m, i]$ be the best partial objective for m selected frames ending at t_i, excluding the final gap to T. Appending t_i to a subset ending at t_j adds its relevance r_i and the coverage reward for the gap $t_i - t_j$, giving the initialization and recurrence

$$\begin{aligned} D[1,i] &= r_i + \beta_{\text{sel}} \log\left(1 + \frac{t_i}{\tau_K}\right), && 1 \leq i \leq N, \\ D[m,i] &= r_i + \max_{m-1 \leq j < i} \left[D[m-1,j] + \beta_{\text{sel}} \log\left(1 + \frac{t_i - t_j}{\tau_K}\right) \right], && 2 \leq m \leq K, \\ & && m \leq i \leq N. \end{aligned} \tag{6}$$

We add the final-gap reward $\beta_{\text{sel}} \log(1 + (T - t_i)/\tau_K)$ to each feasible $D[K, i]$ and backtrack from the best final state to recover the selected frames.

Linear-time evaluation of each DP layer. Scanning every predecessor in Equation 6 would require $\mathcal{O}(KN^2)$ time. We avoid this scan by exploiting the logarithmic gap reward. For $\beta_{\text{sel}} > 0$, exponentiation preserves the ordering of transition values, so finding the best predecessor is equivalent to finding the largest value among the lines

$$L_j(x) = \exp\left(\frac{D[m-1,j]}{\beta_{\text{sel}}}\right)(x + \tau_K - t_j) \quad \text{at } x = t_i. \tag{7}$$

An earlier predecessor with an equal or larger partial objective always dominates a later one, so only predecessors that establish a new maximum partial objective need to be retained. Their line slopes increase in temporal order, as do the query timestamps. We therefore maintain their upper envelope, the pointwise maximum of these lines, in a double-ended queue. Each predecessor enters and leaves the queue at most once per layer, giving $\mathcal{O}(N)$ work per layer and $\mathcal{O}(KN)$ time overall.

Theorem 3 (Global optimality and complexity)**.** *For a fixed, temporally ordered scored pool with $N \geq K$ and $\beta_{\text{sel}} \geq 0$, the exact algorithm returns a global maximizer of F over all K-frame subsets in $\mathcal{O}(KN)$ time, using $\mathcal{O}(KN)$ storage including backtracking indices.*

Appendix A.3 gives the derivation, proof, and pseudocode, including evaluation without forming large exponentials and the $\beta_{\text{sel}} = 0$ case. After optimization, the selected frames are passed to the downstream MLLM in chronological order to generate the answer.

Table 1: Long-video QA accuracy (%) with Qwen3-VL-8B and $K = 64$ input frames. Budget ρ counts frame–query evaluations per second of video; FOCUS is shown at its effective budget, and "Uniform" requires no scoring. Avg. is the unweighted four-benchmark mean. Best results within each budget block are bold, including ties; green parentheses show gains over "Uniform".

Method	Budget ρ	LongVB	Video-MME	LVBench	MLVU	Avg.
Uniform	0	62.4	67.7	41.0	70.3	60.3
Top-K	0.1	63.4	68.7	46.6	70.8	62.4
AKS (CVPR '25)	0.1	63.2	69.1	45.6	71.0	62.2
BOLT (CVPR '25)	0.1	59.3	63.4	46.7	71.9	60.4
Q-Frame (ICCV '25)	0.1	62.9	69.1	42.9	70.9	61.5
AdaQ (arXiv '26)	0.1	63.4	68.9	46.7	70.8	62.4
WFS-SB (CVPR '26)	0.1	63.2	68.3	46.9	71.3	62.4
KFC (ICLR '26)	0.1	62.5	68.2	47.6	71.4	62.4
Ours	0.1	**64.9** (+2.5)	**69.2** (+1.5)	**49.0** (+8.0)	**72.0** (+1.7)	**63.8** (+3.5)
Top-K	0.5	65.7	68.1	50.6	73.8	64.6
AKS (CVPR '25)	0.5	65.1	69.0	47.8	74.5	64.1
BOLT (CVPR '25)	0.5	63.9	69.4	49.2	75.1	64.4
Q-Frame (ICCV '25)	0.5	64.4	67.6	43.5	72.2	61.9
AdaQ (arXiv '26)	0.5	66.2	68.3	50.2	73.7	64.6
WFS-SB (CVPR '26)	0.5	65.5	68.4	50.6	74.7	64.8
FOCUS (ICLR '26)	~ 0.44	64.1	65.3	48.7	71.3	62.4
KFC (ICLR '26)	0.5	65.5	67.8	46.4	74.9	63.6
Ours	0.5	**68.1** (+5.7)	**69.9** (+2.2)	**50.7** (+9.7)	**75.7** (+5.4)	**66.1** (+5.8)
Top-K	1.0	65.9	67.5	51.2	74.1	64.7
AKS (CVPR '25)	1.0	66.2	69.0	49.1	73.6	64.5
BOLT (CVPR '25)	1.0	65.6	69.6	49.5	75.1	65.0
Q-Frame (ICCV '25)	1.0	64.4	68.2	43.9	72.3	62.2
AdaQ (arXiv '26)	1.0	66.6	67.8	50.9	74.3	64.9
WFS-SB (CVPR '26)	1.0	67.6	68.5	51.0	75.5	65.6
KFC (ICLR '26)	1.0	66.0	67.8	45.4	73.8	63.3
Ours	1.0	**68.5** (+6.1)	**69.9** (+2.2)	**51.4** (+10.4)	**75.8** (+5.5)	**66.4** (+6.1)

4 Experiments

4.1 Experimental Setup

Benchmarks and metric. We evaluate on four complementary long-video QA benchmarks. LongVideoBench (Wu et al., 2024) emphasizes evidence retrieval and referring reasoning; Video-MME (Fu et al., 2025) spans diverse domains and video durations; LVBench (Wang et al., 2025a) focuses on information integration over hour-scale videos; and MLVU (Zhou et al., 2025) covers multiple understanding tasks and video genres. Together, they test both localized evidence retrieval and reasoning across extended temporal contexts. We report multiple-choice accuracy (%) over all evaluated questions in each benchmark. *Avg.* denotes the unweighted four-benchmark mean, also used in the ablation and generalization studies.

Models and relevance scorers. We evaluate four representative video LLMs: LLaVA-OneVision (LLaVA-OV) (Li et al., 2025), LLaVA-Video (Zhang et al., 2025c), Qwen3-VL (Bai et al., 2025), and InternVL3.5 (Wang et al., 2025b). Relevance scorers include BLIP (Li et al., 2022), BLIP-2 (Li et al., 2023), and CLIP (Radford et al., 2021). Unless otherwise stated, we use Qwen3-VL-8B-Instruct as the downstream model and BLIP as the relevance scorer in our experiments. We use greedy decoding for downstream answer generation, without stochastic token sampling.

Scoring and frame budgets. For a T-second video, we allocate $B \approx \rho T$ frame–query evaluations, where ρ denotes the scoring allowance per second of video. The main comparison uses $\rho \in \{0.1, 0.5, 1.0\}$ and a final frame budget of $K = 64$ for the MLLM. Baseline scoring protocols and budget accounting are detailed in Appendix C.3.

Table 2: Ablation of the two stages.

Stage 1	Stage 2	Avg.
Uniform scoring	Top-K	64.6
Uniform scoring	Global opt.	65.2
w/o relevance	Global opt.	65.2
w/o coverage	Global opt.	64.2
Full	Top-K	64.9
Full	Global opt.	**66.1**

Table 3: Ablation of the final frame budget.

K	Uniform	Top-K	AKS	WFS-SB	KFC	**Ours**
8	51.4	55.9	56.2	57.3	56.3	**58.3**
16	54.4	58.4	58.3	59.6	59.1	**60.5**
32	58.2	60.8	60.7	62.2	60.9	**63.6**
64	60.3	64.6	64.1	64.8	63.6	**66.1**

Table 4: Generalization across scorers.

Scorer	Top-K	AKS	WFS-SB	KFC	**Ours**
BLIP	64.6	64.1	64.8	63.6	**66.1**
CLIP	64.4	62.7	63.9	63.8	**65.0**
BLIP-2	64.4	62.4	64.3	64.1	**65.1**

Table 5: Generalization across downstream MLLMs.

MLLM	Uniform	Top-K	AKS	WFS-SB	KFC	**Ours**
Qwen3-VL-8B	60.3	64.6	64.1	64.8	63.6	**66.1**
LLaVA-OV-7B	55.6	58.3	58.0	58.3	58.2	**59.3**
LLaVA-Video-7B	59.4	62.0	61.3	61.8	61.0	**62.6**
InternVL3.5-8B	60.7	62.8	62.7	62.9	62.5	**63.8**

4.2 Main Results

Baselines. For controlled comparisons, we reimplement baseline frame-selection rules within a common evaluation pipeline, using the same downstream MLLM, relevance scorer, and final-frame budget. The Uniform baseline samples K evenly spaced frames without relevance scoring. Top-K retains the K highest-scoring observed frames. AKS (Tang et al., 2025) recursively balances query relevance and temporal coverage. BOLT (Liu et al., 2025) uses deterministic inverse-transform sampling of normalized relevance scores. Q-Frame (Zhang et al., 2025a) applies Gumbel-Max sampling to query–frame relevance. AdaQ (Zhang et al., 2026b) adapts its sampling distribution to score variance and samples from a top-p relevance set. WFS-SB (Chen et al., 2026) detects semantic boundaries in the relevance trajectory and selects diverse frames within the resulting segments. KFC, the frame-selection component of Nar-KFC (Fang et al., 2026), jointly optimizes query relevance and visual diversity. FOCUS (Zhu et al., 2026) uses clip-level relevance estimates to guide frame selection. Its official scoring schedule yields an effective budget of approximately 0.44 frame–query evaluations per second of video, so we report it alongside the $\rho = 0.5$ results. Appendix D.1 specifies an additional comparison matching FORTE to FOCUS's scoring count for each question.

Analysis. Table 1 compares selectors at scoring budgets $\rho \in \{0.1, 0.5, 1.0\}$. Our method achieves the highest observed accuracy on all four benchmarks at every budget. Top-K becomes increasingly competitive as more frames are scored, outperforming several more complex selectors at higher budgets. However, the benefits of additional scoring diminish: Increasing ρ from 0.5 to 1.0 brings only small gains in average accuracy despite roughly doubling the number of scorer evaluations. These results suggest that $\rho = 0.5$ offers a practical balance between accuracy and scoring cost.

4.3 Ablation and Generalization Studies

Motivated by the diminishing returns beyond $\rho = 0.5$, we use this scoring budget and $K = 64$ input frames for all ablation and generalization studies unless otherwise stated.

Adaptive relevance scoring and global keyframe optimization. Table 2 compares variants of the two stages. In Stage 1, "Full" uses both terms in Eq. 4, while "w/o" indicates removal of the specified term. With Stage 2 fixed to global keyframe optimization, "Full" outperforms uniform scoring and either single-term variant, supporting the combined use of predicted relevance and coverage. In Stage 2, global keyframe optimization ("Global opt.") maximizes the joint relevance–coverage objective in Eq. 5, whereas Top-K uses relevance alone. Global keyframe optimization improves on Top-K for both uniformly and adaptively scored pools, showing that coverage also matters for final frame selection. Appendix D provides per-benchmark ablations and paired uncertainty estimates.

Final frame budget. Table 3 compares Uniform, Top-K, AKS, WFS-SB, KFC, and our method across final frame budgets. All six improve with more input frames, but ours achieves the highest observed mean accuracy at every tested K. With 16 frames, it reaches the same reported mean

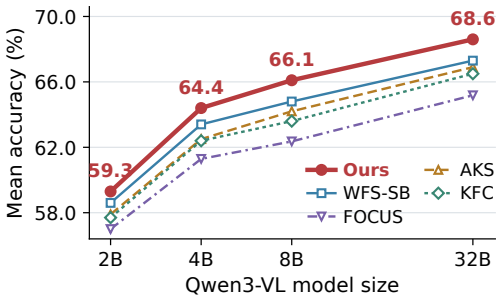


Figure 3: Generalization across model scales. Points show four-benchmark mean accuracy.

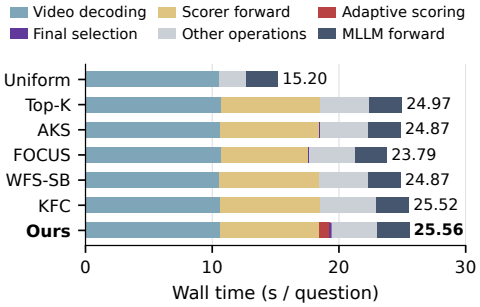


Figure 4: Wall-time decomposition of the entire pipeline for long-video question answering.

accuracy as Uniform with 64, suggesting that question-aware selection can preserve accuracy with substantially fewer input frames.

Parameter sensitivity. We use a single shared configuration of heuristic defaults without development-set tuning (Appendix C) and keep it fixed across all main experiments. To assess sensitivity, we vary the GP length scale, both coverage weights, the uniform initialization fraction, and the number of adaptive scoring batches one at a time, reporting the unweighted mean accuracy across the four evaluation benchmarks (Appendix D.5).

Relevance scorers. We replace BLIP with CLIP (Radford et al., 2021) and BLIP-2 (Li et al., 2023), keeping selector hyperparameters fixed. Our method outperforms Top-K, AKS, WFS-SB, and KFC in mean accuracy with all three scorers (Table 4), showing that its gains are not specific to BLIP.

Downstream MLLMs. We evaluate the same frame selections with LLaVA-OneVision-7B (Li et al., 2025), LLaVA-Video-7B (Zhang et al., 2025c), and InternVL3.5-8B (Wang et al., 2025b). While the strongest baseline varies across models, our method achieves the highest observed mean accuracy among the compared selectors on each of the four MLLMs (Table 5).

Model scales. With $K = 64$ input frames, accuracy increases with Qwen3-VL model size from 2B to 32B parameters. Our method remains ahead of AKS, WFS-SB, KFC, and FOCUS at every tested scale (Figure 3), showing that its advantage persists with larger downstream MLLMs.

4.4 Computational Cost

We evaluate runtime and peak GPU memory on 100 videos with three repeats per method, using Qwen3-VL-8B, BLIP, and $K = 64$ on an A100-80GB GPU. Figure 4 reports the end-to-end breakdown at the default scoring budget. Video decoding and scorer inference account for approximately 72% of FORTE's runtime, whereas adaptive scoring and final optimization together take only 0.88 seconds (3.4%). Its total latency is 25.56 seconds per question, close to Top-K, AKS, WFS-SB, and KFC (24.87–25.52 seconds). Thus, the gains from adaptive scoring and optimization come with modest additional overhead relative to other score-based selectors. Our goal is not to reduce end-to-end cost relative to uniform sampling, which requires no relevance scoring, but to achieve higher accuracy under a fixed relevance-scoring budget. All scored methods reach the same selection-stage GPU memory peak of 2.52 GiB, compared with 23.46 GiB for MLLM answering. Appendix D.7 provides the measurement protocol and full numerical results.

5 Conclusion

We present FORTE, a training-free framework for long-video QA that decides which frames to score and which to retain under separate budgets. Both decisions balance relevance with temporal coverage, using efficient GP predictions during scoring and measured relevance during final selection. By exploiting the objective's logarithmic gap structure, our exact algorithm finds the globally optimal subset of the scored pool in $\mathcal{O}(KN)$ time. Across four benchmarks, FORTE achieves the highest observed mean accuracy among the compared methods under the tested budgets, consistently generalizing across the evaluated relevance scorers and downstream MLLMs.

AI USE STATEMENT

In this work, we used generative AI tools only to assist with drafting and polishing the manuscript for clarity and readability. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To support reimplementation of FORTE, we describe its scoring and optimization procedures in Section 3, with details of the GP approximation and batched scoring in Appendix B. Appendix A.3 provides the derivation, pseudocode, and numerical evaluation details of the $\mathcal{O}(KN)$ exact algorithm. The assumptions and complete proofs underlying the theoretical guarantees are provided in Appendix A.

To support reproduction of the experiments, we specify the benchmarks, models, relevance scorers, and evaluation protocol in Section 4, with hyperparameters, preprocessing, and baseline configurations documented in Appendix C. For computational-cost comparisons, we additionally report the hardware configuration and timing protocol in Appendix D.7.

For additional implementation details, please refer to our code provided in the supplementary materials.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

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, 2026.

Yukang Chen, Fuzhao Xue, Dacheng Li, Qinghao Hu, Ligeng Zhu, Xiuyu Li, Yunhao Fang, Haotian Tang, Shang Yang, Zhijian Liu, Ethan He, Hongxu Yin, Pavlo Molchanov, Jan Kautz, Jim Fan, Yuke Zhu, Yao Lu, and Song Han. LongVILA: Scaling long-context visual language models for long videos. In *International Conference on Learning Representations (ICLR)*, volume 2025, pp. 18227–18246, 2025.

Janghoon Cho, Jungsoo Lee, Munawar Hayat, Kyuwoong Hwang, Fatih Porikli, and Sungha Choi. FLoC: Facility location-based efficient visual token compression for long video understanding. In *International Conference on Learning Representations (ICLR)*, volume 2026, pp. 102973–102996, 2026.

Yang Ding, Xin Lai, Yizhen Zhang, Wei Li, Ruihang Chu, and Yujiu Yang. VideoZoomer: Reinforcement-learned temporal focusing for long video reasoning. In *International Conference on Learning Representations (ICLR)*, volume 2026, pp. 20087–20111, 2026.

Bo Fang, Yuxin Song, Haoyuan Sun, Qiangqiang Wu, Wenhao Wu, and Antoni Chan. Threading keyframe with narratives: MLLMs as strong long video comprehenders. In *International Conference on Learning Representations (ICLR)*, volume 2026, pp. 63417–63431, 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, 2025.

Bo He, Hengduo Li, Young Kyun Jang, Menglin Jia, Xuefei Cao, Ashish Shah, Abhinav Shrivastava, and Ser-Nam Lim. MA-LMM: Memory-augmented large multimodal model for long-term video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13504–13514, 2024.

Hosu Lee, Junho Kim, Hyunjun Kim, and Yong Man Ro. ReFoCUS: Reinforcement-guided frame optimization for contextual understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 8291–8302, 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. *Transactions on Machine Learning Research*, 2025.

Jialuo Li, Bin Li, Jiahao Li, and Yan Lu. Divide, then ground: Adapting frame selection to query types for long-form video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11369–11380, 2026a.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pp. 12888–12900. PMLR, 2022.

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 *International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19730–19742. PMLR, 2023.

Xinhao Li, Yi Wang, Jiashuo Yu, Xiangyu Zeng, Yuhan Zhu, Haian Huang, Jianfei Gao, Kunchang Li, Yinan He, Chenting Wang, Yu Qiao, Yali Wang, and Limin Wang. VideoChat-Flash: Hierarchical compression for long-context video modeling. In *International Conference on Learning Representations (ICLR)*, volume 2026, pp. 109089–109117, 2026b.

Hao Liang, Jiapeng Li, Tianyi Bai, Xijie Huang, Linzhuang Sun, Zhengren Wang, Conghui He, Bin Cui, Chong Chen, and Wentao Zhang. KeyVideoLLM: Towards large-scale video keyframe selection. *arXiv preprint arXiv:2407.03104*, 2024.

Shuming Liu, Chen Zhao, Tianqi Xu, and Bernard Ghanem. BOLT: Boost large vision-language model without training for long-form video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3318–3327, 2025.

Ziyu Ma, Chenhui Gou, Hengcan Shi, Bin Sun, Shutao Li, Hamid Rezatofighi, and Jianfei Cai. DrVideo: Document retrieval based long video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18936–18946, 2025.

Junwen Pan, Qizhe Zhang, Rui Zhang, Ming Lu, Xin Wan, Yuan Zhang, Chang Liu, and Qi She. TimeSearch-R: Adaptive temporal search for long-form video understanding via self-verification reinforcement learning. In *International Conference on Learning Representations (ICLR)*, volume 2026, pp. 8376–8405, 2026.

Rui Qian, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Shuangrui Ding, Dahua Lin, and Jiaqi Wang. Streaming long video understanding with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 119336–119360. Curran Associates, Inc., 2024.

Jihao Qiu, Lingxi Xie, Xinyue Huo, Qi Tian, and Qixiang Ye. LongVideo-R1: Smart navigation for low-cost long video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 40505–40515, 2026.

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 *International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 2021.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Xiaoqian Shen, Yunyang Xiong, Changsheng Zhao, Lemeng Wu, Jun Chen, Chenchen Zhu, Zechun Liu, Fanyi Xiao, Balakrishnan Varadarajan, Florian Bordes, Zhuang Liu, Hu Xu, Hyunwoo J. Kim, Bilge Soran, Raghuraman Krishnamoorthi, Mohamed Elhoseiny, and Vikas Chandra. LongVU: Spatiotemporal adaptive compression for long video-language understanding. In *International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 54582–54599. PMLR, 2025.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26160–26169, 2025.

Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Haozhe Chi, Xun Guo, Tian Ye, Yanting Zhang, Yan Lu, Jenq-Neng Hwang, and Gaoang Wang. MovieChat: From dense token to sparse memory for long video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18221–18232, 2024.

Hui Sun, Shiyin Lu, Huanyu Wang, Qing-Guo Chen, Zhao Xu, Weihua Luo, Kaifu Zhang, and Ming Li. MDP^3: 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, 2025.

Wenhui Tan, Ruihua Song, Jiaze Li, Jianzhong Ju, and Zhenbo Luo. Think-Clip-Sample: Slow-fast frame selection for video understanding. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 12157–12161. IEEE, 2026.

Xi Tang, Jihao Qiu, Lingxi Xie, Yunjie Tian, Jianbin Jiao, and Qixiang Ye. Adaptive keyframe sampling for long video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29118–29128, 2025.

Weihan Wang, Zehai He, Wenyi Hong, Yean Cheng, Xiaohan Zhang, Ji Qi, Ming Ding, Xiaotao Gu, Shiyu Huang, Bin Xu, Yuxiao Dong, and Jie Tang. LVBench: An extreme long video understanding benchmark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22958–22967, 2025a.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, Guanzhou Chen, Zichen Ding, Changyao Tian, Zhenyu Wu, Jingjing Xie, Zehao Li, Bowen Yang, Yuchen Duan, Xuehui Wang, Zhi Hou, Haoran Hao, Tianyi Zhang, Songze Li, Xiangyu Zhao, Haodong Duan, Nianchen Deng, Bin Fu, Yinan He, Yi Wang, Conghui He, Botian Shi, Junjun He, Yingtong Xiong, Han Lv, Lijun Wu, Wenqi Shao, Kaipeng Zhang, Huipeng Deng, Biqing Qi, Jiaye Ge, Qipeng Guo, Wenwei Zhang, Songyang Zhang, Maosong Cao, Junyao Lin, Kexian Tang, Jianfei Gao, Haian Huang, Yuzhe Gu, Chengqi Lyu, Huanze Tang, Rui Wang, Haijun Lv, Wanli Ouyang, Limin Wang, Min Dou, Xizhou Zhu, Tong Lu, Dahua Lin, Jifeng Dai, Weijie Su, Bowen Zhou, Kai Chen, Yu Qiao, Wenhai Wang, and Gen Luo. InternVL3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025b.

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 (ECCV)*, pp. 58–76. Springer, 2024.

Ziyang Wang, Shoubin Yu, Elias Stengel-Eskin, Jaehong Yoon, Feng Cheng, Gedas Bertasius, and Mohit Bansal. VideoTree: Adaptive tree-based video representation for LLM reasoning on long

videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3272–3283, 2025c.

Yuetian Weng, Mingfei Han, Haoyu He, Xiaojun Chang, and Bohan Zhuang. LongVLM: Efficient long video understanding via large language models. In *European Conference on Computer Vision (ECCV)*, pp. 453–470. Springer, 2024.

Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. LongVideoBench: A benchmark for long-context interleaved video-language understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 28828–28857. Curran Associates, Inc., 2024.

Shoubin Yu, Jaemin Cho, Prateek Yadav, and Mohit Bansal. Self-chained image-language model for video localization and question answering. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 76749–76771. Curran Associates, Inc., 2023.

Sicheng Yu, Chengkai Jin, Huanyu Wang, Zhenghao Chen, Sheng Jin, Zhongrong Zuo, Xiaolei Xu, Zhenbang Sun, Bingni Zhang, Jiawei Wu, Hao Zhang, and Qianru Sun. Frame-Voyager: Learning to query frames for video large language models. In *International Conference on Learning Representations (ICLR)*, volume 2025, pp. 84154–84179, 2025.

Junyang Zhang, Puhan Luo, Chen Tang, Yuxi Shi, and Xiang-Yang Li. Coverage-driven adaptive keyframe selection for video understanding. *arXiv preprint arXiv:2608.00714*, 2026a.

Kun Zhang, Chenxin Fang, Tao Chen, Baiyang Song, Yunhang Shen, Yiyi Zhou, and Rongrong Ji. Towards fast and effective long video understanding of multimodal large language models via adaptive quasi-Gaussian sampling. *arXiv preprint arXiv:2606.24187*, 2026b.

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, 2025a.

Xian Zhang, Zexi Wu, Zinuo Li, Hongming Xu, Luqi Gong, Farid Boussaid, Naoufel Werghi, and Mohammed Bennamoun. AdaRD-Key: Adaptive relevance-diversity keyframe sampling for long-form video understanding. *arXiv preprint arXiv:2510.02778*, 2025b.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. LLaVA-Video: Video instruction tuning with synthetic data. *Transactions on Machine Learning Research*, 2025c.

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, 2025.

Zirui Zhu, Hailun Xu, Yang Luo, Yong Liu, Kanchan Sarkar, Zhenheng Yang, and Yang You. FOCUS: Efficient keyframe selection for long video understanding. In *International Conference on Learning Representations (ICLR)*, volume 2026, pp. 98463–98486, 2026.

Yuanhao Zou, Shengji Jin, Andong Deng, Youpeng Zhao, Jun Wang, and Chen Chen. A.I.R.: Enabling adaptive, iterative, and reasoning-based frame selection for video question answering. In *International Conference on Learning Representations (ICLR)*, volume 2026, pp. 11302–11329, 2026.

This appendix complements the main manuscript with theoretical proofs, implementation details, and additional quantitative and qualitative analyses. It is organized as follows:

1. Appendix A proves the temporal coverage properties and the optimality and complexity guarantees for the exact algorithm used in global keyframe optimization.
2. Appendix B describes the efficient GP approximation and the batched implementation of adaptive relevance scoring.
3. Appendix C specifies FORTE's implementation settings, the common evaluation protocol, and the baseline implementations.
4. Appendix D examines the contributions of FORTE's components and design choices, quantifies uncertainty in accuracy differences, and evaluates numerical stability, computational cost, and performance across video durations and scoring-budget regimes.
5. Appendix E extends the discussion of long-video modeling and keyframe selection, clarifying how FORTE relates to existing approaches.
6. Appendix F compares scored candidates, selected frames, and model answers through qualitative examples of evidence acquisition, final retention, and a shared failure case.

A Coverage Objective and Selection Guarantees

The coverage objective in Section 3 guides both adaptive relevance scoring and global keyframe optimization. We first prove its two key properties: it favors balanced temporal spacing at a fixed frame count, and the gain from adding a frame decreases as nearby frames accumulate. We then derive the exact algorithm and prove that it maximizes the final relevance–coverage objective in $\mathcal{O}(KN)$ time. Throughout, we use the main manuscript's interval $[0, T]$ and represent frame sets by their timestamps. The coverage scale $\tau > 0$ remains fixed as the selected set changes.

A.1 Temporal Gaps and Uniform-Spacing Optimality

Let $\mathcal{S} = \{s_1 < \cdots < s_m\} \subseteq [0, T]$ contain the selected timestamps. Following Section 3.2, set $s_0 = 0$ and $s_{m+1} = T$. The temporal gaps are

$$g_i(\mathcal{S}) = s_{i+1} - s_i, \qquad i = 0, \ldots, m. \tag{8}$$

These gaps include the two boundary gaps, so they are nonnegative and sum to T. For brevity, write $f_\tau(g) = \log(1 + g/\tau)$. The coverage objective is then

$$C_\tau(\mathcal{S}) = \sum_{i=0}^{m} f_\tau(g_i(\mathcal{S}))\,. \tag{9}$$

For a fixed number of selected frames, the total gap length is always T; only its distribution among the gaps changes. The concavity of f_τ makes balanced gaps preferable to a mixture of very short and very long gaps. For the empty set, there is one gap of length T, giving $C_\tau(\varnothing) = f_\tau(T)$.

Proof of Theorem 1. The function f_τ is strictly concave for $g \geq 0$, since $f''_\tau(g) = -(\tau + g)^{-2} < 0$. Applying Jensen's inequality to the $m + 1$ gaps gives

$$\frac{C_\tau(\mathcal{S})}{m+1} = \frac{1}{m+1} \sum_{i=0}^{m} f_\tau(g_i(\mathcal{S})) \leq f_\tau\left(\frac{T}{m+1}\right). \tag{10}$$

The bound depends only on T, m, and τ, not on the selected timestamps. By strict concavity, equality holds if and only if all gaps equal $T/(m + 1)$. The timestamps $s_i = iT/(m + 1)$ for $i = 1, \ldots, m$ attain this bound, proving Theorem 1. □

The theorem characterizes coverage alone when any timestamp can be selected. In practice, selection is restricted to the scored candidates and also accounts for relevance, so the optimal subset need not be uniformly spaced.

A.2 Marginal Coverage and Diminishing Returns

The marginal coverage gain measures the additional temporal coverage contributed by one frame. For a frame at timestamp $t \notin \mathcal{S}$, adding t replaces one temporal gap by two smaller gaps; all other gaps remain unchanged. Let L and R be the nearest selected timestamps to the left and right of t, using 0 or T when the corresponding neighbor is absent. The marginal gain is therefore

$$\Delta C_\tau(t \mid \mathcal{S}) = f_\tau(t - L) + f_\tau(R - t) - f_\tau(R - L). \tag{11}$$

Proof of Theorem 2. The original gap has length $R - L$. Inserting t replaces its reward $f_\tau(R - L)$ by $f_\tau(t - L) + f_\tau(R - t)$, establishing Equation 11 and its dependence only on the containing gap. Let $a = t - L \geq 0$ and $b = R - t \geq 0$ be the distances to the two neighbors. Writing the marginal gain as $h_\tau(a, b)$ gives

$$h_\tau(a, b) = \log\left(\frac{(\tau + a)(\tau + b)}{\tau(\tau + a + b)}\right) = \log\left(1 + \frac{ab}{\tau(\tau + a + b)}\right) \geq 0. \tag{12}$$

Thus, adding a frame cannot decrease coverage, establishing monotonicity.

To prove diminishing returns, we examine how this gain changes as the neighboring selected timestamps move closer to t. Its partial derivatives satisfy

$$\frac{\partial h_\tau}{\partial a} = \frac{1}{\tau + a} - \frac{1}{\tau + a + b} \geq 0, \qquad \frac{\partial h_\tau}{\partial b} = \frac{1}{\tau + b} - \frac{1}{\tau + a + b} \geq 0. \tag{13}$$

Hence, the gain is nondecreasing in each distance. Now take $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{U}$ and $t \in \mathcal{U} \setminus \mathcal{B}$. Because $\mathcal{B}$ contains every timestamp in $\mathcal{A}$, its nearest neighbors of t can only be closer, or remain the same. Both a and b therefore weakly decrease when passing from $\mathcal{A}$ to $\mathcal{B}$, so $\Delta C_\tau(t \mid \mathcal{A}) \geq \Delta C_\tau(t \mid \mathcal{B})$. This is precisely the diminishing-returns condition for submodularity on the finite timestamp set $\mathcal{U}$. □

Relation to submodular maximization. Theorem 2 places our temporal coverage function within the class of monotone submodular set functions. Since the relevance term is modular, the final objective in Eq. 5 is also monotone submodular for nonnegative relevance scores and $\beta_{\text{sel}} \geq 0$. Consequently, generic algorithms for cardinality-constrained submodular maximization, such as marginal-gain greedy selection, could be applied to this objective.

Our setting, however, has additional structure that is not used by a generic submodular optimizer: candidate frames are totally ordered in time, and the coverage term decomposes over consecutive temporal gaps. For the logarithmic gap reward used here, this structure permits exact optimization rather than approximate greedy maximization. Appendix A.3 exploits this property to derive the $O(KN)$ algorithm used by FORTE.

A.3 Exact Algorithm

We optimize F over a fixed scored pool $\mathcal{O} = \{t_1 < \cdots < t_N\} \subseteq [0, T]$, with $T > 0$ and $1 \leq K \leq N$. For brevity, write $\beta = \beta_{\text{sel}}$ and $\tau = \tau_K = T/(K+1)$, keeping both fixed throughout the optimization. We first establish the recurrence, then show how the logarithmic gap reward reduces its evaluation to $\mathcal{O}(N)$ work per layer.

Dynamic-programming recurrence. Let $D[m, i]$ be the best partial objective for m selected frames ending at t_i, excluding the final gap to T. Appending t_i to a selection ending at t_j adds only $r_i + \beta f_\tau(t_i - t_j)$, giving

$$D[1, i] = r_i + \beta f_\tau(t_i), \qquad 1 \leq i \leq N, \tag{14}$$

$$D[m, i] = r_i + \max_{m-1 \leq j < i} \left[D[m - 1, j] + \beta f_\tau(t_i - t_j)\right], \qquad \begin{matrix} 2 \leq m \leq K, \\ m \leq i \leq N. \end{matrix} \tag{15}$$

After the last layer, we restore the right-boundary gap:

$$F^\star = \max_{K \leq i \leq N} \left[D[K, i] + \beta f_\tau(T - t_i)\right]. \tag{16}$$

The recurrence is exact by induction. The initialization is the partial objective of the only one-frame selection ending at t_i. If the previous layer is optimal, every m-frame selection ending at t_i consists

of a feasible $(m-1)$-frame selection ending at some t_j, followed by t_i. Maximizing over these predecessors therefore gives the optimal partial objective. Equation 16 accounts for the only omitted gap, and stored predecessor indices recover an optimal subset by backtracking.

Transforming predecessor searches into line queries. Assume $\beta > 0$. Within a fixed layer, write $d_j = D[m-1, j]$ and define the transition contribution

$$V_j(x) = d_j + \beta \log\left(1 + \frac{x - t_j}{\tau}\right), \qquad L_j(x) = a_j(x + \tau - t_j), \quad a_j = \exp(d_j/\beta). \tag{17}$$

For eligible predecessors $j < i$ at $x = t_i$, every $L_j(x)$ is positive and

$$\max_j V_j(x) = \beta \log\left(\frac{\max_j L_j(x)}{\tau}\right). \tag{18}$$

Thus, the same predecessor maximizes the transition contribution and the corresponding line value.

Discarding dominated predecessors. Suppose $p < j$ and $d_p \geq d_j$. For every future query $x \geq t_j$,

$$V_p(x) - V_j(x) = d_p - d_j + \beta \log\left(\frac{\tau + x - t_p}{\tau + x - t_j}\right) > 0. \tag{19}$$

The earlier predecessor always contributes more, so j can be discarded. We therefore insert only predecessors whose d_j is a strict new maximum among all feasible predecessors processed so far. Their slopes a_j increase strictly, while query timestamps increase with i. Importantly, the running maximum includes predecessors already removed from the queue.

Maintaining the upper envelope. The upper envelope is the pointwise maximum of the retained lines. We maintain it using a monotone upper-envelope construction, storing its potentially optimal lines in a double-ended queue ordered by increasing slope. For retained predecessors $p < q$ with $d_p < d_q$, their intersection is

$$X(p, q) = t_p - \tau + \frac{t_q - t_p}{1 - \exp(-(d_q - d_p)/\beta)}. \tag{20}$$

When inserting a new line j, the middle line q of the last three lines p, q, j can be removed whenever $X(p, q) \geq X(q, j)$: it has no interval on which it is strictly best. At a query, if the second line is at least as good as the first, we remove the first. Because the second has larger slope, the first cannot become better at a later query. We insert the newly eligible predecessor $j = i - 1$ immediately before querying t_i, ensuring that only $j < i$ participates. Algorithm 1 gives the full procedure.

Proof of Theorem 3. For $\beta > 0$, Equation 18 preserves the maximizing predecessor. Equation 19 discards only predecessors dominated at every future query. Back removal preserves the upper envelope, and front removal discards only lines permanently overtaken as query timestamps increase. Consequently, the queue returns a maximizing eligible predecessor for every DP state. The recurrence and terminal boundary reward established above then recover a global maximizer of F.

Each predecessor enters the queue at most once and is removed at most once per layer. Insertion and querying therefore require $\mathcal{O}(N)$ total operations per layer. Initialization and terminal maximization take $\mathcal{O}(N)$ time, and backtracking takes $\mathcal{O}(K)$, giving $\mathcal{O}(KN)$ time overall. Two rolling DP rows and the queue use $\mathcal{O}(N)$ working memory; storing backtracking indices uses $\mathcal{O}(KN)$ additional memory. For $\beta = 0$, selecting the K largest relevance scores maximizes F directly. Linear-time order-statistic selection followed by a scan of the ordered pool also satisfies the claimed bounds. □

Numerical evaluation and scope. The lines in Equation 17 establish the equivalence but need not be formed explicitly. We evaluate $V_j(x)$ using $\mathrm{log1p}((x - t_j)/\tau)$ and the denominator in Equation 20 using $-\mathrm{expm1}(-(d_q - d_p)/\beta)$. This avoids exponentiating large DP values and reduces cancellation for nearby values. The optimality and operation-count statements use real arithmetic, with elementary functions counted as constant-cost operations. Floating-point comparisons near ties can still be sensitive to rounding; avoiding large exponentials is not a guarantee of numerically exact decisions. For $K = 1$, initialization and terminal maximization suffice. No pairwise gap-reward matrix is needed, and retaining only the optimal value reduces storage to $\mathcal{O}(N)$.

Algorithm 1 Exact keyframe optimization with a monotone upper envelope

Require: Sorted timestamps $t_1 < \cdots < t_N$, scores r_i, $1 \le K \le N$, $T > 0$, $\beta \ge 0$
Ensure: Optimal indices $J_1 < \cdots < J_K$ and objective value $F^\star$

if $\beta = 0$ **then**
  return the K highest-scoring indices in temporal order and their score sum
end if
$\tau \leftarrow T/(K+1)$; $\mathrm{prev}_i \leftarrow r_i + \beta \log(1 + t_i/\tau)$ for $i = 1, \ldots, N$
Allocate predecessor array $P[1{:}K, 1{:}N]$
for $m = 2, \ldots, K$ **do**
  curr $\leftarrow (-\infty, \ldots, -\infty)$; $H \leftarrow$ empty deque; $d_{\max} \leftarrow -\infty$
  Define $V_j(x)$ and $X(p, q)$ by Equations 17 and 20, with $d_j = \mathrm{prev}_j$
  for $i = m, \ldots, N$ **do**
    $j \leftarrow i - 1$
    if $\mathrm{prev}_j > d_{\max}$ **then**
      $d_{\max} \leftarrow \mathrm{prev}_j$
      while $|H| \ge 2$ and $X(H_{-2}, H_{-1}) \ge X(H_{-1}, j)$ **do**
        Remove the last element of H
      end while
      Append j to H
    end if
    while $|H| \ge 2$ and $V_{H_2}(t_i) \ge V_{H_1}(t_i)$ **do**
      Remove the first element of H
    end while
    $j^\star \leftarrow H_1$; $\mathrm{curr}_i \leftarrow r_i + V_{j^\star}(t_i)$; $P[m, i] \leftarrow j^\star$
  end for
  prev $\leftarrow$ curr
end for
$J_K \leftarrow \arg\max_{K \le i \le N} [\mathrm{prev}_i + \beta \log(1 + (T - t_i)/\tau)]$
$F^\star \leftarrow \mathrm{prev}_{J_K} + \beta \log(1 + (T - t_{J_K})/\tau)$
for $m = K, K-1, \ldots, 2$ **do**
  $J_{m-1} \leftarrow P[m, J_m]$
end for
return $(J_1, \ldots, J_K), F^\star$

H_1, H_2 denote the first two queue elements; H_{-2}, H_{-1} denote the last two. Equal DP-value records are discarded, and ties at a query favor the later line.

B Details of Adaptive Relevance Scoring

Starting from sparse uniform initialization, FORTE predicts relevance for unscored frames and selects additional frames using both predicted relevance and temporal coverage. It then evaluates these frames with the relevance scorer ψ and uses their measured scores to update the GP. Repeating this process allows new observations to inform subsequent scoring decisions until the scoring budget B is exhausted. This section explains how we make GP prediction efficient and organize the select–score–update process into batches.

B.1 Gaussian-Process Relevance Predictor

The GP relevance predictor uses only the timestamps and measured relevance scores of frames already evaluated. Its predictions guide which frames to score next; global keyframe optimization instead uses measured relevance scores. Following Section 3.2, $\mathcal{O}$ denotes the timestamps of the frames scored so far, and $\mathbf{r}$ contains their normalized relevance scores. The predicted relevance of an unscored frame at timestamp t is $\mu(t)$ in Equation 2, with μ_{obs} set to the mean of $\mathbf{r}$. For frames with no nearby scored frames, predictions approach this observed average rather than zero. Score normalization is described in Appendix C.

The main computational challenge is updating the GP as the observed set grows. A dense fit accounts for every pair of observed timestamps, even though the RBF kernel assigns small values to temporally distant pairs. We reduce this cost by retaining a band of entries around the kernel matrix's diagonal and evaluating predictions from a local neighborhood. All reported experiments involving GP prediction use the approximation described below.

Fitting the GP with a banded kernel matrix. Each frame retains its position in the original video's temporal order. At the start of a scoring round, we scan this ordered universe $\mathcal{U}$ and use a Boolean observation mask to collect the $n_{\text{obs}} = |\mathcal{O}|$ observed timestamps and their scores. This produces the required chronological order without comparison sorting. In this order, entries near the kernel matrix's diagonal relate neighboring observations. We keep $[\mathbf{K}]_{ij}$ when $|i - j| \leq w$ and set the other entries to zero, where w is the half-bandwidth, initialized to $\min\{128, n_{\text{obs}} - 1\}$ in our experiments. The band is defined by observation count rather than a fixed temporal duration. The resulting matrix, $\mathbf{K}^{(w)}$, defines the approximate fit:

$$(\mathbf{K}^{(w)} + \epsilon\mathbf{I})\boldsymbol{\alpha} = \mathbf{r} - \mu_{\text{obs}}\mathbf{1}, \tag{21}$$

where $\mathbf{r}$ contains the normalized observed scores. The diagonal adjustment $\epsilon\mathbf{I}$ is the same numerical-stability term as in Equation 2. We obtain the coefficients $\boldsymbol{\alpha}$ using banded Cholesky factorization and triangular solves, without forming a matrix inverse. These coefficients are then reused to predict relevance for unscored frames.

Numerical safeguards. Zeroing kernel entries outside the band can destroy positive definiteness and cause Cholesky factorization to fail. We therefore allow two forms of recovery: increasing the diagonal adjustment and, if needed, widening the band. All computations used to construct and solve the GP system use double precision (FP64). For a fixed half-bandwidth w, we begin with $\epsilon = 10^{-12}$ and increase it tenfold after each failed factorization, up to 10^{-3}. If none of these attempts succeeds and a wider band is allowed, we set $w \leftarrow \min\{2w, w_{\max}, n_{\text{obs}} - 1\}$, rebuild the matrix, and restart the adjustment schedule from 10^{-12}. We use a common recovery cap $w_{\max} = 1024$ across benchmarks and scoring budgets. We stop at the first successful factorization and use its fitted coefficients and half-bandwidth for prediction. If all attempts fail at the maximum permitted bandwidth, the implementation raises an error rather than returning predictions.

Predicting relevance for unscored frames. We traverse unscored frames in chronological order while maintaining a pointer into the ordered observed timestamps. As prediction timestamps increase, the pointer advances to identify each frame's insertion position and never moves backward. This locates all prediction neighborhoods in a single linear sweep. For a frame at timestamp t, we take at most $2w + 1$ adjacent observation indices around its insertion position, denoted $\mathcal{J}_w(t)$. We approximate its predicted relevance $\mu(t)$ by restricting the posterior-mean sum to this neighborhood:

$$\mu(t) \approx \mu_{\text{obs}} + \sum_{i \in \mathcal{J}_w(t)} k(t, t_i)\alpha_i. \tag{22}$$

Thus, predicting relevance for another unscored frame requires only a short weighted sum, not another GP fit. This is the local approximation used to compute the relevance predictions in Section 3.2.

Computational complexity. We separate the cost of the GP predictor from the remaining computations used to construct each acquisition batch.

GP preparation, fitting, and prediction. Extracting observations in temporal order takes $\mathcal{O}(|\mathcal{U}|)$ time. Dense GP factorization would require $\mathcal{O}(n_{\text{obs}}^3)$ time and $\mathcal{O}(n_{\text{obs}}^2)$ memory. With half-bandwidth w, banded factorization instead requires $\mathcal{O}(n_{\text{obs}}w^2)$ time and $\mathcal{O}(n_{\text{obs}}w)$ memory; constructing the banded matrix and performing the triangular solves each take $\mathcal{O}(n_{\text{obs}}w)$ time. For n_{pred} prediction frames, the chronological sweep locates all neighborhoods in $\mathcal{O}(n_{\text{obs}} + n_{\text{pred}})$ time, since each observed timestamp is passed at most once. Evaluating the local posterior-mean sums then takes $\mathcal{O}(n_{\text{pred}}w)$ time. The complete GP preparation, fitting, and prediction cost per round is therefore

$$\mathcal{O}\big(|\mathcal{U}| + n_{\text{obs}}w^2 + n_{\text{pred}}w\big)\,. \tag{23}$$

Since $n_{\text{obs}}, n_{\text{pred}} \leq |\mathcal{U}|$, this is $\mathcal{O}(|\mathcal{U}|)$ for fixed w. Including the observation mask and prediction arrays, memory use is $\mathcal{O}(|\mathcal{U}| + n_{\text{obs}}w)$, also linear for fixed w.

Equation 23 describes one factorization attempt followed by prediction at the successful bandwidth. Retries add matrix-construction and factorization costs at the attempted bandwidths. With the bounded diagonal-adjustment schedule and fixed recovery cap $w_{\max}$, the retry-inclusive GP cost remains linear in $|\mathcal{U}|$ per round. Across R scoring rounds, the total GP preparation, fitting, and prediction cost is therefore

$$\mathcal{O}(R|\mathcal{U}|), \tag{24}$$

or $\mathcal{O}(|\mathcal{U}|)$ for a fixed round count such as the default $R = 4$.

Acquisition and heap maintenance. We next account for the frame-by-frame acquisition decisions after the GP predictions have been computed. Let $m_0 = |\mathcal{O}_0|$ denote the number of uniformly scored initialization frames. Since $m_0 \approx B/2$ and these frames are approximately uniformly distributed over the ordered frame universe, they partition $\mathcal{U}$ into temporal gaps containing at most

$$G_0 = \mathcal{O}\left(\frac{|\mathcal{U}|}{m_0}\right) = \mathcal{O}\left(\frac{|\mathcal{U}|}{B}\right) \tag{25}$$

unscored candidates per gap. Subsequent acquisitions only split these initial gaps, so every gap encountered during adaptive scoring contains at most G_0 candidates.

At the start of each scoring round, the GP predictions $\mu_h(t)$ are fixed for that round. We compute the acquisition value

$$\mu_h(t) + \beta_{\text{acq}} \Delta C_{\tau_B}(t \mid \mathcal{A})$$

for each currently unscored candidate and construct a max-heap over these values. Computing the initial keys by a chronological sweep and heapifying them both require $\mathcal{O}(|\mathcal{U}|)$ time. Across R rounds, this contributes $\mathcal{O}(R|\mathcal{U}|)$ time.

When a frame is selected, only the temporal gap containing that frame is split. Because the marginal coverage gain depends only on the two endpoints of the containing gap, acquisition values outside this gap are unchanged. At most G_0 remaining candidates are therefore affected by one selection. Their GP relevance terms remain fixed within the round, while their marginal coverage gains can only decrease as the selected set grows. Using a position-aware max-heap, each affected key can consequently be recomputed and decreased in $\mathcal{O}(\log |\mathcal{U}|)$ time. Thus, one adaptive selection requires

$$\mathcal{O}(G_0 \log |\mathcal{U}|) \tag{26}$$

heap-maintenance time. Since the total number of adaptive selections is at most $B - m_0 = \mathcal{O}(B)$, their total cost is

$$\mathcal{O}(BG_0 \log |\mathcal{U}|)\,. \tag{27}$$

Together with per-round key construction and heapification, the non-GP acquisition cost is therefore

$$\mathcal{O}(R|\mathcal{U}| + BG_0 \log |\mathcal{U}|)\,. \tag{28}$$

The heap, acquisition-value arrays, and gap bookkeeping require $\mathcal{O}(|\mathcal{U}|)$ additional memory.

Under the scoring protocol used in our experiments, the scoring budget satisfies $B = \Theta(\rho T)$, while a video with fixed frame rate v contains $|\mathcal{U}| = \Theta(vT)$ frames. Hence

$$G_0 = \mathcal{O}\left(\frac{|\mathcal{U}|}{B}\right) = \mathcal{O}\left(\frac{v}{\rho}\right). \tag{29}$$

For fixed frame rate v and scoring allowance ρ, $G_0 = \mathcal{O}(1)$, and Equation 28 simplifies to

$$\mathcal{O}(R|\mathcal{U}| + B \log |\mathcal{U}|)\,. \tag{30}$$

For a fixed number of rounds, this is

$$\mathcal{O}(|\mathcal{U}| + B \log |\mathcal{U}|)\,. \tag{31}$$

Overall adaptive-scoring computation. Combining the GP computation in Equation 24 with acquisition and heap maintenance in Equation 28, the complete adaptive-scoring computation, excluding relevance-scorer forward passes, is

$$\mathcal{O}(R|\mathcal{U}| + BG_0 \log |\mathcal{U}|) \tag{32}$$

for fixed GP bandwidth and recovery cap. Under fixed frame rate, scoring density, and round count, this becomes

$$\boxed{\mathcal{O}(|\mathcal{U}| + B \log |\mathcal{U}|)}. \tag{33}$$

The corresponding working memory is

$$\mathcal{O}(|\mathcal{U}| + n_{\text{obs}} w)\,,$$

including the GP representation, prediction arrays, max-heap, and acquisition bookkeeping.

These bounds cover observation extraction, GP fitting and posterior-mean prediction, acquisition-value computation, and frame-selection data structures. They do not include video decoding, relevance-scorer inference, final keyframe optimization, or downstream MLLM inference. Appendix D.7 reports these pipeline costs separately.

Appendix D.6 reports a replay-based recovery audit, retry-inclusive GP timings, and a separate comparison with dense GP prediction. These checks distinguish successful numerical recovery from agreement with the untruncated predictor.

B.2 Batch Construction and Observation Updates

Section 3.2 describes choosing and scoring one frame before updating the GP. Although this uses each new score immediately, it requires a GP refit after every evaluation and prevents scoring several selected frames in the same forward pass. Our implementation instead chooses a batch of frames, scores them in parallel, and updates the GP once their scores are available. We call one repetition of these steps a scoring round.

Let $\mathcal{O}_0$ contain the timestamps of the frames scored during uniform initialization. We divide the remaining $B - |\mathcal{O}_0|$ evaluations as evenly as possible among R rounds. Round $h \in \{0, \dots, R-1\}$ selects b_h new frames, with the allowances summing to the remaining scoring budget. The default uses four rounds; each round proceeds as follows.

Predict from the available scores. At the start of round h, let $\mathcal{O}_h$ contain the timestamps of the frames scored so far and $\mathbf{r}_h$ their normalized relevance scores in timestamp order. We use the approximation above to compute $\mu_h(t)$, the predicted relevance of an unscored frame at timestamp t. The subscript h identifies the current round's observations; the prediction has the same role as $\mu(t)$ in Section 3.2. These predictions remain fixed until the batch has been scored.

Choose frames while updating coverage. We choose the batch one frame at a time. If coverage were also held fixed, several choices could concentrate around the same predicted relevance peak. We therefore update coverage after every choice, counting both previously scored frames and frames already chosen for the current batch. Let $\mathcal{A}_{h,j}$ contain the timestamps of all these frames after j choices, starting from $\mathcal{A}_{h,0} = \mathcal{O}_h$. The jth chosen frame has timestamp $t_{h,j}$, given by

$$\begin{aligned} t_{h,j} &= \arg\max_{t \in \mathcal{U} \setminus \mathcal{A}_{h,j-1}} \left[\mu_h(t) + \beta_{\text{acq}} \Delta C_{\tau_B}(t \mid \mathcal{A}_{h,j-1})\right], \\ \mathcal{A}_{h,j} &= \mathcal{A}_{h,j-1} \cup \{t_{h,j}\}, \end{aligned} \tag{34}$$

for $j = 1, \dots, b_h$, with coverage evaluated over $[0, T]$ and $\tau_B = T/(B+1)$. The relevance term uses the predictions made at the start of the round, while the coverage term reflects all choices made so far. Updating coverage requires only the timestamps of the chosen frames, not their measured relevance scores.

This frame-by-frame update is inexpensive: each choice splits one temporal gap, changing coverage gains only for candidates inside it. By Equation 11, evaluating a candidate's gain uses only its neighboring selected timestamps and takes $\mathcal{O}(1)$ arithmetic operations. Maintaining the ordered timestamps and finding the next best candidate are separate from this local gain calculation.

Score the batch and update the observations. After all b_h frames have been chosen, we evaluate them with the relevance scorer using batched forward passes. These evaluations are independent, so all selected frames can be scored in parallel given sufficient computational resources. Once all scores are available, we add them to the observations and set $\mathcal{O}_{h+1} = \mathcal{A}_{h,b_h}$. If budget remains, these observations determine the GP predictions for the next round.

Within each round, GP predictions remain fixed, while temporal coverage is updated after each frame choice. Once the selected frames have been scored, their measured relevance scores are used to update the GP for the next round.

C Implementation and Evaluation Details

C.1 FORTE Implementation

Default settings. Unless otherwise stated, we use the RBF kernel in Equation 1 with length scale $\ell = 10$ seconds. The coverage weights are $\beta_{\text{acq}} = 3.5$ for adaptive relevance scoring and $\beta_{\text{sel}} = 1$ for global keyframe optimization. We allocate approximately half of the scoring budget to uniform initialization and divide the remainder among four scoring rounds. We use the $\mathcal{O}(KN)$ exact algorithm for final frame selection. Both stages of FORTE are deterministic given fixed scorer outputs and hyperparameters.

Hyperparameter choices and numerical safeguards. We use heuristic defaults shared across benchmarks and scoring budgets, without development-set tuning. With half-budget uniform initialization, a typical interior gap is approximately $2\tau_B$, and scoring its midpoint yields a coverage gain of $\log(4/3)$. We set $\beta_{\text{acq}} \approx 1/\log(4/3) \approx 3.5$ so that this reference exploration bonus is comparable to the unit relevance range. For final optimization, $\beta_{\text{sel}} = 1$ provides a simple unit-weight coverage regularizer alongside measured relevance. The GP length scale of 10 seconds, initialization fraction of 0.5, and four scoring rounds are simple fixed defaults. The sensitivity experiments in Appendix D.5 support these choices and indicate that performance is robust to moderate parameter changes. We also use a common numerical-recovery rule: FP64 arithmetic, an initial half-bandwidth of $\min\{128, n_{\text{obs}} - 1\}$, the same diagonal-adjustment schedule, and a recovery cap of $w_{\max} = 1024$ (Appendix B.1). This cap limits expansion after failed Cholesky attempts; it does not prescribe the bandwidth used in every fit.

Setting the scoring budget. The scoring allowance ρ determines how many frame–query evaluations are available per second of video, not which frames are scored. We obtain the integer budget B by counting distinct frames on a uniform reference grid. For a video with $n > 1$ original frames at frame rate v, the timestamp span is $T = (n-1)/v$. We compute the reference stride $d_\rho = \max(1, \lfloor v/\rho \rfloor)$ and construct $\max(1, \lfloor n/d_\rho \rfloor)$ reference points. Mapping these points to original frames and removing duplicates gives B.

Baselines that score uniformly sampled candidates use the frames on this reference grid. FORTE instead scores an initial grid at $\rho/2$ and allocates the remaining evaluations adaptively. Thus, budget-matched methods evaluate the same number of frames but need not evaluate the same frames. Integer rounding can make the initialization count $|\mathcal{O}_0|$ differ slightly from $B/2$. We divide the remaining $B - |\mathcal{O}_0|$ evaluations as evenly as possible among the scoring rounds, so their batch sizes differ by at most one frame.

Normalizing relevance scores. We min–max normalize the observed relevance scores to $[0, 1]$ for each question, placing relevance on a common scale before combining it with coverage. If all observed scores are identical, their normalized values are set to zero. GP predictions are clipped to the range of the normalized observations. The final scores r_i remain fixed throughout global keyframe optimization.

Implementing the exact algorithm. We maintain two rolling DP rows and a double-ended queue for the current layer, using $\mathcal{O}(N)$ working memory, and store $\mathcal{O}(KN)$ predecessor indices for backtracking. Transition values and line intersections are evaluated as described in Appendix A.3; no pairwise gap-reward matrix is materialized. Backtracking returns the selected frames in chronological order.

C.2 Evaluation Protocol

Dataset splits and question counts. We evaluate the LongVideoBench validation split (1,337 questions), the Video-MME test split (2,700 questions), the public LVBench evaluation set (1,549 questions), and the multiple-choice portion of the MLVU development set (2,174 questions). For reproducibility, the LVBench annotations come from the split named `train` in `lmms-lab/LVBench`; this is the public benchmark annotation set, not data used to train our method. The MLVU annotations come from the `mlvu` configuration and `test` split of `sy1998/MLVU_dev`.

Available input modalities. We use video frames without audio or separately supplied subtitles on all four benchmarks. In particular, we do not use LongVideoBench's interleaved-subtitle input or

Video-MME's subtitle-augmented input. Neither the scorer nor the downstream MLLM receives speech transcripts, generated captions, or separately extracted OCR text. Text embedded in the video remains visible in the selected frames. The downstream MLLM receives the question and its answer options, whereas relevance scoring uses the question alone.

Scorer checkpoints and scoring heads. Our default scorer is BLIP ViT-L, loaded through LAVIS as `blip_image_text_matching` with model type `large`. This configuration uses the COCO-retrieval checkpoint `model_large_retrieval_coco.pth` and 384×384 image preprocessing.[1] We use the image–text matching (ITM) head and take the positive-class probability after the two-class softmax as the relevance score, rather than using the contrastive head.

For BLIP-2, we use `Salesforce/blip2-itm-vit-g` with its image–text retrieval model and associated processor. Its ITM head produces two-class logits for the Q-Former query tokens; these logits are averaged across the query tokens before applying softmax, and the positive-class probability is the frame's relevance score. For CLIP, we use `openai/clip-vit-large-patch14` and compute cosine similarity between the projected, ℓ_2-normalized image and text embeddings, without logit-temperature scaling. Each scorer uses its corresponding evaluation image processor and tokenizer. All three receive only the original question text: we do not concatenate answer options or rewrite the question for scoring. The resulting scalar scores are normalized as described above before they enter FORTE.

Downstream prompting and generation. We use each downstream model's chat or conversation template with the selected frames and a multiple-choice question. The text contains the question followed by one answer option per line, using `A.`, `B.`, etc. for LongVideoBench, the supplied option labels for Video-MME, and `(A)`, `(B)`, etc. for LVBench and MLVU. It ends with "Answer with the option's letter from the given choices directly." The Video-MME task additionally prepends "Select the best answer to the following multiple-choice question based on the video. Respond with only the letter (A, B, C, or D) of the correct option." This inherited instruction mentions subtitles, but no subtitle text is supplied. We provide no in-context examples or chain-of-thought instruction. Answer generation uses greedy decoding without stochastic sampling, with a maximum of 16 new tokens for Video-MME and 32 for the other benchmarks.

Answer extraction. We compare the extracted option letter with the reference answer, without an LLM judge. The LongVideoBench and Video-MME task parsers strip standard answer prefixes and extract the first uppercase option letter (`A`–`E` and `A`–`D`, respectively). LVBench and MLVU use the evaluation harness's multiple-choice label extractor, which recognizes option letters in standard answer formats. An unparseable answer is counted as incorrect.

Original timestamps and nonuniform frame inputs. All selected frames are supplied in chronological order, retaining their original video-frame indices. For original frame index i and source video frame rate v, its timestamp is i/v seconds. In Qwen3-VL, the selected frames are passed as a video tensor together with the original frame indices, source frame rate, and total number of original frames. The processor uses this metadata to represent their temporal positions; further frame sampling is disabled, so a nonuniform selection is not replaced by a uniformly sampled sequence. In the downstream-model comparisons, InternVL3.5 receives a textual prefix specifying the video duration and the selected frames' original timestamps in seconds. LLaVA-Video and LLaVA-OneVision receive the chronologically ordered frames without an additional timestamp instruction. These input conventions are held fixed across selectors for each downstream model.

Accuracy aggregation. We report the percentage of correctly answered questions for each benchmark, weighting all evaluated questions equally. The four-benchmark mean, *Avg.*, is the unweighted mean of these accuracies.

Completing short selections. We adopt a common *preserve-and-supplement* rule for all methods. If a selector returns fewer than K frames, we retain its selected frames and repeatedly add the original video frame farthest in time from the current set until K frames are obtained, breaking ties in favor of earlier timestamps. The added frames require no relevance evaluations and therefore do not consume additional scoring budget.

[1] Official checkpoint configuration: `https://github.com/salesforce/LAVIS/blob/main/lavis/configs/models/blip_itm_large.yaml`.

C.3 Baseline Implementations

We compare baseline frame-selection rules within a common evaluation pipeline, using Qwen3-VL-8B-Instruct and BLIP unless otherwise stated. In the scorer ablation, we change the relevance scorer while keeping each selection rule and its parameters fixed. In the downstream-model ablation, we evaluate the same selected frames with different MLLMs. All selected frames are supplied in chronological order. The descriptions below identify the official implementations and specify our adaptations to this common protocol.

All score-based baselines except FOCUS score a uniformly sampled candidate pool before selecting the final input frames. We refer to these as *fixed-grid baselines* and match their scoring allowance ρ and final-frame budget K to FORTE. FOCUS follows its official scoring procedure, detailed below. Appendix C.2 specifies the common rule for completing short selections.

Uniform and Top-K. Uniform samples K evenly spaced frames from the original video without relevance scoring. Top-K selects the highest-scoring frames from the observed candidate pool.

AKS.[2] (Tang et al., 2025) We follow the official AKS implementation and its default normalization and threshold settings, setting the maximum recursion depth to $\log_2 K$ for the tested frame budgets.

BOLT.[3] (Liu et al., 2025) We follow BOLT's official inverse-transform sampler and default hyperparameters.

Q-Frame.[4] (Zhang et al., 2025a) We follow Q-Frame's official implementation and default selection temperature. We keep downstream image preprocessing consistent across methods.

AdaQ.[5] (Zhang et al., 2026b) We follow AdaQ's official implementation, retaining its default variance-scaling coefficient, top-p threshold, and minimum-support parameter.

WFS-SB.[6] (Chen et al., 2026) We follow the official WFS-SB implementation and default selection hyperparameters, using BLIP relevance scores and visual features in our common evaluation pipeline.

KFC from Nar-KFC.[7] (Fang et al., 2026) We follow the official greedy KFC implementation, retaining its diversity and low-rank settings. We fix the candidate-reduction target and the local-refinement radius across benchmarks.

FOCUS.[8] (Zhu et al., 2026) We follow the official FOCUS implementation and default selection hyperparameters, using BLIP as the scorer. Its scoring schedule determines the number of evaluations rather than enforcing a prescribed ρ, yielding an effective allowance of approximately 0.44 frame–query evaluations per second of video in the full-benchmark comparison.

Randomness in stochastic selectors. Q-Frame uses Gumbel-Max sampling, and AdaQ also samples frames from a relevance-dependent distribution. To account for this randomness, we evaluate each method with five different random seeds, keeping the candidate pools, relevance scores, hyperparameters, and downstream inference settings fixed within each experimental condition. For each seed, we evaluate the selected frames and compute accuracy separately on each benchmark; the reported accuracy in Table 1 is the arithmetic mean across the five runs. Avg. is then the unweighted mean of the four seed-averaged benchmark accuracies. Downstream answer generation uses greedy decoding, so the seeds vary frame selection rather than token sampling.

D Additional Experimental Results

Unless otherwise stated, experiments use Qwen3-VL-8B-Instruct, BLIP, $\rho = 0.5$, $K = 64$, and the exact algorithm for global keyframe optimization, following Appendix C. Accuracy is question-weighted within each benchmark; Avg. weights the four benchmarks equally.

[2] https://github.com/ncTimTang/AKS.
[3] https://github.com/sming256/BOLT; MIT license.
[4] https://github.com/xiaomi-research/q-frame; Apache-2.0 license.
[5] https://github.com/Zkayovo-xmu/AdaQ.
[6] https://github.com/MAC-AutoML/WFS-SB.
[7] https://github.com/bofang98/Nar-KFC; Apache-2.0 license.
[8] https://github.com/NUS-HPC-AI-Lab/FOCUS; Apache-2.0 license.

D.1 Matched-Budget Comparison with FOCUS

Table 1 places FOCUS, whose official procedure yields an effective scoring allowance of approximately 0.44, alongside methods evaluated at $\rho = 0.5$. To separate the effect of frame selection from this budget difference, Table 6 compares FOCUS and FORTE with identical scoring counts for each question. FOCUS retains its official procedure, and FORTE receives the same number of relevance evaluations as the number of distinct frames scored by FOCUS for each video–question pair. Only the count is shared; each method chooses its own frames. This matches the integer budgets directly, including cases where FOCUS's per-video scoring rate differs from its aggregate rate of approximately 0.44.

Both methods use BLIP, Qwen3-VL-8B-Instruct, $K = 64$, and the same question sets, prompting, decoding, and preserve-and-supplement rule. FORTE retains its default hyperparameters, initialization fraction, four scoring rounds, and exact final optimizer; only its total scoring budget changes.

Table 6: Matched-budget comparison at an effective scoring allowance of approximately $\rho = 0.44$, using BLIP, Qwen3-VL-8B-Instruct, and $K = 64$. Both methods score the same number of distinct frames for each question. Values are QA accuracy (%); Avg. is the unweighted four-benchmark mean.

Method	Effective ρ	LongVB	Video-MME	LVBench	MLVU	Avg.
FOCUS	$\sim$0.44	64.1	65.3	48.7	71.3	62.4
FORTE (Ours)	$\sim$0.44	67.8	69.7	50.5	75.2	65.8

Results. FORTE achieves higher observed accuracy than FOCUS on all four benchmarks under matched scoring budgets, with a mean of 65.8% compared with 62.4%. The accuracy advantage observed in Table 1 therefore persists when both methods receive the same number of relevance evaluations, rather than arising solely from FORTE's larger scoring allowance in that comparison.

D.2 Contributions of the Two Stages

Table 7 breaks down the main-text ablation in Table 2 by benchmark.

Stage 1: Adaptive relevance scoring. With Stage 2 fixed to global keyframe optimization, combining GP-predicted relevance with temporal coverage improves mean accuracy from 65.2% with uniform scoring to 66.1%, and also outperforms either single-term variant. Coverage-only scoring matches uniform scoring at 65.2%, whereas relevance-only scoring achieves 64.2%, indicating that the benefit comes from jointly exploiting predicted relevance and coverage rather than either term alone. When Stage 2 is instead fixed to Top-K, adaptive scoring yields a smaller improvement over uniform scoring, from 64.6% to 64.9%. Thus, adaptive scoring improves the quality of the candidate pool, but its downstream benefit depends on how the final frames are selected; a relevance-only Top-K selector cannot fully exploit candidates that are useful for improving temporal coverage.

Stage 2: Global keyframe optimization. On identical scored pools, replacing Top-K with global keyframe optimization improves mean accuracy for both scoring policies: from 64.6% to 65.2% on the uniformly scored pool and from 64.9% to 66.1% on the adaptively scored pool. The larger gain on the adaptive pool indicates that the two stages are complementary: adaptive scoring makes more informative and temporally distributed candidates available, while global optimization determines which of these candidates should be retained by jointly considering measured relevance and coverage.

D.3 Effect of the Final Optimization Algorithm

Unlike the Top-K comparison, Table 8 isolates the optimizer by holding the objective in Equation 5 fixed. Exact and marginal-gain greedy optimization receive identical pools and measured scores under both uniform and adaptive scoring, with $\beta_{\text{sel}} = 1$ and the same completion rule.

The greedy algorithm, labeled "Approx.", starts from the empty set and repeatedly adds the frame with the largest gain $r_i + \beta_{\text{sel}} \Delta C_{\tau_K}(t_i \mid \mathcal{S})$ until K frames have been selected, keeping $\tau_K = T/(K+1)$ fixed.

Table 7: Contributions of the scoring policy and final selector under matched scoring budgets and completion rules. "Global opt." denotes global keyframe optimization of the joint relevance–coverage objective. Values are accuracy (%); bold marks the best result in each column.

Stage 1	Stage 2	LongVB	Video-MME	LVBench	MLVU	Avg.
Uniform	Top-K	65.7	68.1	50.6	73.8	64.6
Uniform	Global opt.	66.6	69.2	50.4	74.5	65.2
Coverage only	Global opt.	65.9	69.6	50.4	74.9	65.2
Relevance only	Global opt.	66.4	67.9	49.6	73.0	64.2
Relevance + coverage	Top-K	66.7	67.5	**51.0**	74.4	64.9
Relevance + coverage	Global opt.	68.1	**69.9**	50.7	**75.7**	**66.1**

Table 8: Exact versus approximate optimization of the same relevance–coverage objective at $\rho = 0.5$ and $K = 64$. Approx. denotes marginal-gain greedy maximization. Accuracy is reported in percent; Avg. is the unweighted four-benchmark mean. $\Delta F = F(\mathcal{S}_{\text{exact}}) - F(\mathcal{S}_{\text{solver}})$ measures the objective shortfall relative to the exact optimum, in objective units. Objective gaps and final-selection times are evaluated on pools with $N \geq K$, averaged within each benchmark, then equally across benchmarks. Timing excludes relevance scoring and downstream inference.

Scoring	Solver	QA accuracy (%)					Mean ΔF	Time (ms)
		LongVB	Video-MME	LVBench	MLVU	Avg.		
Uniform	Exact	66.6	69.2	50.4	74.5	65.2	0.0	45
Uniform	Approx.	66.1	68.9	50.5	74.1	64.9	0.4	42
Adaptive	Exact	68.1	69.9	50.7	75.7	66.1	0.0	50
Adaptive	Approx.	67.3	69.4	50.8	75.3	65.7	0.7	47

Results. Exact optimization improves mean accuracy by 0.3 points on uniform pools and 0.4 points on adaptive pools, eliminating mean objective gaps of 0.4 and 0.7, respectively, at an additional 3 ms per selection. Accuracy improves on LongVideoBench, Video-MME, and MLVU under both policies; Approx. is slightly higher on LVBench. The objective guarantee therefore accompanies a mean QA improvement, not a guarantee of higher accuracy on every benchmark.

D.4 Paired Uncertainty Analysis

FORTE is deterministic given fixed scorer outputs and hyperparameters, and answer generation uses greedy decoding, so we do not repeat FORTE inference with different random seeds. Here, we quantify video-sampling uncertainty through a paired video-cluster bootstrap. This is distinct from averaging Q-Frame and AdaQ over five selector seeds (Appendix C.3); those two methods are not included in the paired comparisons below.

We compare default FORTE with Uniform, AKS, FOCUS, and WFS-SB. All share the default model and frame budget; FOCUS retains its effective scoring allowance of approximately 0.44, and Uniform requires no relevance scoring.

Measuring paired differences. Let $c^{\mathrm{F}}_{di}, c^{\mathrm{b}}_{di} \in \{0, 1\}$ indicate whether FORTE and a baseline correctly answer question i in benchmark d. We compute their accuracy difference in percentage points as

$$\widehat{\Delta}_d = \frac{100}{n_d} \sum_{i=1}^{n_d} \left(c^{\mathrm{F}}_{di} - c^{\mathrm{b}}_{di} \right), \qquad \widehat{\Delta}_{\text{avg}} = \frac{1}{4} \sum_{d=1}^{4} \widehat{\Delta}_d, \tag{35}$$

where n_d is the number of evaluated questions in benchmark d. Pairing keeps each question's outcomes together rather than treating the two methods' accuracies as independent measurements.

Video-cluster bootstrap. Questions about the same video may have correlated outcomes, so we resample whole videos rather than individual questions. Within each benchmark, a replicate samples the original number of videos with replacement and retains all questions from each sampled video, including repeated occurrences. The same sampled videos are used for all five methods. We

recompute question-weighted accuracy differences within each benchmark, then average the four differences with equal weight. Resampling is independent across benchmarks.

We use $20{,}000$ replicates with random seed 42. The 2.5th and 97.5th percentiles define each 95% confidence interval. These intervals describe video-sampling uncertainty conditional on the evaluated models and settings, not variation across inference runs.

Results. At the default setting, the 95% confidence intervals for the four-benchmark mean differences against Uniform, AKS, FOCUS, and WFS-SB all lie above zero (Table 9), supporting a mean accuracy advantage over each of these baselines. FORTE also has higher observed accuracy in every benchmark-level comparison, with intervals excluding zero against Uniform on all four benchmarks.

Table 9: Paired comparisons of FORTE with four baselines. Each baseline has two columns: the accuracy difference Δ (FORTE minus baseline, in percentage points) and its 95% video-cluster bootstrap confidence interval. Positive differences favor FORTE. Avg. is the unweighted four-benchmark mean. Its 95% confidence interval is obtained by averaging the four benchmark differences within each bootstrap replicate, then taking the 2.5th and 97.5th percentiles of these averages.

Benchmark	Uniform		AKS		FOCUS		WFS-SB	
	Δ	95% CI	Δ	95% CI	Δ	95% CI	Δ	95% CI
LongVideoBench	$+5.7$	[3.7, 7.7]	$+3.0$	[1.2, 4.8]	$+4.0$	[1.9, 6.0]	$+2.6$	[0.7, 4.5]
Video-MME	$+2.2$	[0.9, 3.6]	$+0.9$	$[-0.3, 2.2]$	$+4.6$	[3.3, 6.0]	$+1.5$	[0.3, 2.8]
LVBench	$+9.7$	[7.0, 12.3]	$+3.0$	[0.8, 5.3]	$+2.0$	$[-0.4, 4.4]$	$+0.1$	$[-1.8, 2.0]$
MLVU	$+5.4$	[3.7, 7.2]	$+1.2$	$[-0.3, 2.7]$	$+4.4$	[2.8, 5.9]	$+1.0$	$[-0.4, 2.4]$
Avg.	$+5.8$	[4.8, 6.8]	$+2.0$	[1.1, 2.9]	$+3.8$	[2.9, 4.8]	$+1.3$	[0.5, 2.2]

D.5 Parameter Sensitivity

Figure 5 varies one parameter at a time around the heuristic defaults in Appendix C, reporting four-benchmark mean accuracy. These sweeps assess sensitivity rather than select defaults. The initialization and round-count sweeps keep the total scoring budget fixed.

Scoring coverage weight. In Figure 5(a), every positive weight improves on relevance-only scoring. Accuracy rises up to $\beta_{\text{acq}} = 3$, remains near its peak from 3 to 5, and declines slightly at larger weights. The default of 3.5 lies within this broad high-performing region.

Selection coverage weight. In Figure 5(b), all positive weights outperform relevance-only optimization. Accuracy peaks at $\beta_{\text{sel}} = 0.5$ but remains similar from 0.25 to 1.5, including the default of 1, before declining at the two largest weights. We retain the unit-weight default rather than the sweep's best-performing value, so the main results reflect a shared heuristic setting rather than a weight selected to maximize evaluation accuracy.

Initialization fraction. Accuracy varies by only 0.2 points across fractions from 0.25 to 0.75, with 0.5 giving the highest result (Figure 5(c)). All three outperform spending the entire budget on uniform scoring: reserving evaluations for adaptive scoring matters more than the precise split within this range.

Number of scoring rounds. More frequent GP updates improve accuracy at a fixed scoring budget (Figure 5(d)), but doubling the rounds from four to eight adds only 0.1 points. Four rounds retain most of the observed benefit with fewer refits.

GP length scale. At $\rho = 0.1$, accuracy increases from $\ell = 1$ to 10 seconds and decreases slightly at 20 seconds (Figure 5(e)–(f)). At $\rho = 0.5$, all scales lie within 0.2 points, with 1 and 10 seconds tied for the highest accuracy. A broader neighborhood thus appears more useful under sparse observations; the shared 10-second default is highest or joint-highest at both budgets.

D.6 Numerical Validation of the GP Predictor

We assess numerical recovery, its cost, and agreement with dense GP prediction separately: successful factorization does not by itself establish approximation accuracy.

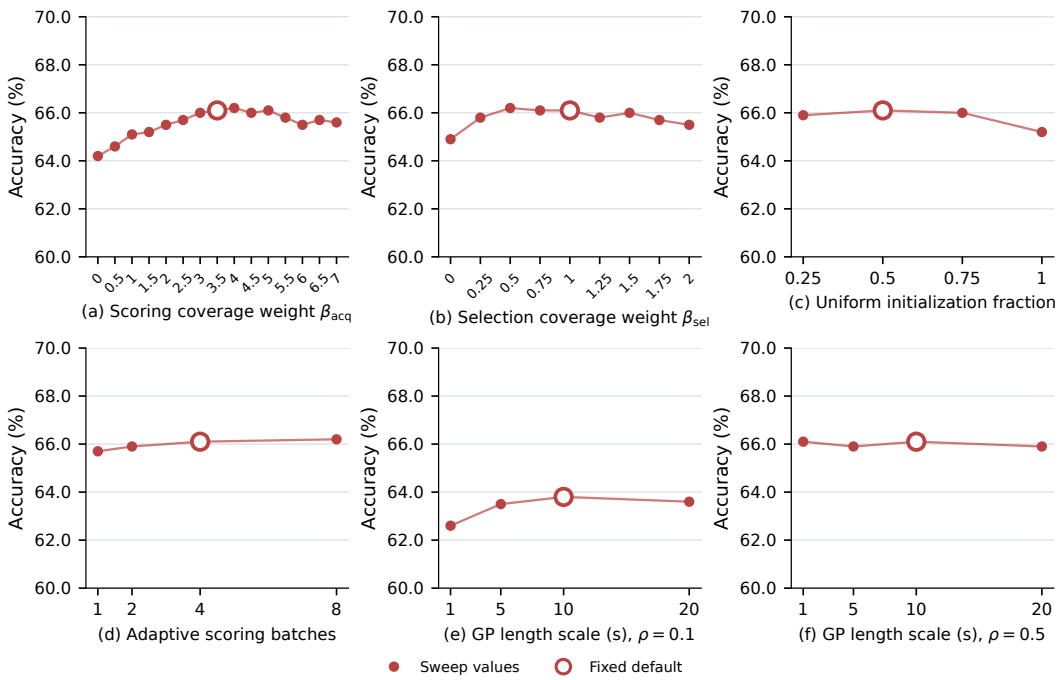


Figure 5: Parameter sensitivity measured by the unweighted four-benchmark mean accuracy (%). We vary one parameter at a time around the heuristic defaults, marked by hollow circles. Panels (a)–(d) use $\rho = 0.5$; panels (e)–(f) use the indicated budgets. All panels use $K = 64$.

Numerical recovery. We replay adaptive scoring on all four benchmarks at $\rho \in \{0.1, 0.5, 1.0\}$ using the original numerical settings, revealing saved scores only when the policy selects their frames. All 23,280 question–budget pairs exactly reproduce their saved scored pools.

All 91,247 fits succeed (Table 10); 1.95% require an increased diagonal adjustment and 0.55% a wider band. No retries occur at $\rho = 0.1$, and they are more frequent at the largest budget. The successful half-bandwidth is at most 128 for 99.45% of fits; the diagonal adjustment remains 10^{-12} for 98.48% and never exceeds 10^{-3}.

Table 10: GP recovery in the scoring replay. Counts and percentages indicate fits requiring any increase in the diagonal adjustment or initial half-bandwidth. All fits succeed without terminal errors. The common recovery cap is $w_{\max} = 1024$; the largest successful half-bandwidth w is reported separately.

Benchmark	ρ	Fits	Adjustment increase		Bandwidth expansion		$w_{\max}$	Largest successful w
			Count	Rate (%)	Count	Rate (%)		
LongVideoBench	0.1	4,195	0	0.00	0	0.00	1024	128
Video-MME	0.1	10,302	0	0.00	0	0.00	1024	128
LVBench	0.1	6,196	0	0.00	0	0.00	1024	128
MLVU	0.1	8,696	0	0.00	0	0.00	1024	128
LongVideoBench	0.5	5,132	38	0.74	9	0.18	1024	512
Video-MME	0.5	10,794	19	0.18	4	0.04	1024	256
LVBench	0.5	6,196	49	0.79	13	0.21	1024	512
MLVU	0.5	8,696	17	0.20	7	0.08	1024	512
LongVideoBench	1.0	5,348	336	6.28	117	2.19	1024	1024
Video-MME	1.0	10,800	284	2.63	55	0.51	1024	512
LVBench	1.0	6,196	629	10.15	151	2.44	1024	512
MLVU	1.0	8,696	406	4.67	145	1.67	1024	1024

Cost of numerical recovery. We measure GP fitting and prediction time on a duration-stratified subset of 100 videos, with 25 per benchmark and one question per video. Across the three budgets and all scoring rounds, this gives $1{,}148$ observed sets on which to fit the GP. We time each set three times on the same CPU host with one math-library thread, then average the repeats before computing the summary statistics.

Table 11 includes every factorization attempt and its matrix construction. Failed attempts account for 0.097% of GP time; prediction for unscored frames dominates. End-to-end costs are reported in Appendix D.7.

Table 11: GP time per observed set on the 100-video subset. Fitting includes all retries. The last row reports the portion spent on unsuccessful attempts, already included above; its 95th percentile is zero because fewer than 5% of the observed sets require retries.

Component	Mean (ms)	95th percentile (ms)
Matrix construction	1.229	2.708
Factorization	0.653	2.373
Triangular solve	0.041	0.152
Total fitting wall time	1.999	5.380
Relevance prediction	223.272	948.733
Total GP wall time	225.271	959.180
Unsuccessful attempts (included above)	0.218	0.000

Agreement with a dense GP. On a separate, duration-stratified subset of 40 videos chosen before inspecting results, we compare predictors with identical observations, prior mean, length scale, diagonal adjustment, and FP64 arithmetic. Both predict relevance for the same unscored frames and use the same coverage term to choose the next frame.

After clipping predictions to the normalized score range, the mean absolute difference, averaged over observed sets, is 0.00706 (95th percentile: 0.02375). Both predictors choose the same next frame in 424 of 451 sets (94.0%), indicating close agreement on the tested pools.

Downstream QA with dense and banded GP prediction. Agreement on a fixed observed set does not establish whether the approximation affects the final answer: small prediction differences can change subsequent scoring decisions and the frames supplied to the MLLM. To test this downstream effect, we compare dense-GP and banded-GP FORTE on the full evaluation sets of LongVideoBench, Video-MME, LVBench, and MLVU, using the same splits and questions as Table 1. The comparison uses $\rho = 0.5$ and $K = 64$, with BLIP and Qwen3-VL-8B-Instruct. Both variants start from identical uniform observations, then independently perform adaptive scoring, global keyframe optimization, and answer generation. Their scored pools can therefore diverge as relevance predictions change.

The dense reference uses the full kernel matrix and all observed frames in each posterior-mean prediction; the banded variant uses the approximation in Appendix B.1. Both retain the same RBF kernel, prior-mean rule, length scale, FP64 arithmetic, and diagonal-adjustment schedule. Scoring and final-frame budgets, normalization and clipping, coverage weights, batching, final optimization, completion, and downstream inference settings remain unchanged. Table 12 reports full-benchmark QA accuracy and GP fitting-and-prediction time, measured on the same CPU host and averaged over three timing repeats.

The reported four-benchmark mean accuracies differ by 0.1 percentage point in favor of dense GP, with absolute differences of at most 0.2 points on individual benchmarks. Banded GP reduces fitting-and-prediction time from 15.07 to 0.81 seconds per question, an approximately $18.6\times$ speedup. This substantial reduction in GP computation, accompanied by a small observed accuracy difference, supports using the banded approximation as FORTE's default predictor.

Table 12: Dense-GP and banded-GP FORTE on the full evaluation sets of all four benchmarks at $\rho = 0.5$ and $K = 64$, using the same questions as Table 1. Accuracy is reported in percent; Avg. is the unweighted four-benchmark mean. GP time includes fitting and prediction across all scoring rounds per question, excluding scorer and MLLM inference. Times are averaged within each benchmark and then equally across benchmarks.

GP variant	LongVB	Video-MME	LVBench	MLVU	Avg.	GP time (s/question)
Banded (default)	68.1	69.9	50.7	75.7	66.1	0.81
Dense	68.3	69.8	50.9	75.8	66.2	15.07

D.7 Computational Cost

Table 13 gives the runtime breakdown underlying Figure 4 and peak GPU memory. Fixed-budget methods use $\rho = 0.5$; FOCUS retains its official scoring schedule.

Videos and repetitions. We select 25 videos per benchmark, five from each of five equally sized duration groups, and pair each with its first annotated question. Every method runs three times on these 100 pairs using the same CPU host and A100-SXM4-80GB GPU. We average over repeats, then over pairs.

Measurement protocol. Each run has eight logical CPUs: decoding uses four workers with one codec thread each, preprocessing uses four threads, and math libraries use one. Frames are decoded in chunks of 32 and scored in FP32 mini-batches of 16. Selection and answering run sequentially on the same GPU; their separately measured wall times are summed, excluding loading and warmup. Inference uses CUDA-synchronized timing, and decoding records exposed wait rather than summed worker CPU time.

Timing components. Video decoding includes decoding for both scoring and answering. Scorer forward measures BLIP inference, while MLLM forward includes visual encoding, prompt processing, and autoregressive answer generation. FORTE's adaptive-scoring component includes GP fitting, relevance prediction, and coverage-based acquisition; its final optimization is reported separately. Other operations include preprocessing, data transfers, input construction, and remaining pipeline work. A dash denotes a component that is inapplicable or not separately reported, rather than a measured zero.

Cost breakdown. Decoding and scorer inference dominate runtime. FORTE spends 0.83 seconds on adaptive scoring and 0.05 seconds on final optimization, together 3.4% of its 25.56-second total. Its latency is within 0.7 seconds of all compared fixed-grid baselines. Uniform is faster because it requires no relevance scoring.

GPU memory. Peak allocated memory is the maximum over videos and repeats, reported separately for selection and answering. All scored methods reach 2.52 GiB and 23.46 GiB, respectively; FORTE does not increase either peak. Uniform's selection-stage memory is unreported.

Table 13: Mean runtime and peak allocated GPU memory on 100 videos with three repeats per method. Fixed-budget scored methods use $\rho = 0.5$, while FOCUS follows its official schedule; all use $K = 64$. Dashes denote inapplicable or separately unreported components. Rounding can cause displayed components to differ slightly from the reported totals.

Method	Mean time (s / question)							Peak GPU (GiB)	
	Video decoding	Scorer forward	Adaptive scoring	Final selection	Other operations	MLLM forward	Total	Selection	Answering
Uniform	10.58	–	–	–	2.11	2.51	15.20	–	23.46
Top-K	10.67	7.87	–	–	3.85	2.58	24.97	2.52	23.46
AKS	10.61	7.87	–	–	3.85	2.54	24.87	2.52	23.46
FOCUS	10.70	6.93	–	–	3.67	2.49	23.79	2.52	23.46
WFS-SB	10.56	7.87	–	–	3.87	2.57	24.87	2.52	23.46
KFC	10.65	7.87	–	–	4.47	2.53	25.52	2.52	23.46
Ours	10.60	7.87	0.83	0.05	3.65	2.56	25.56	2.52	23.46

D.8 Duration Breakdowns

Table 14 uses the annotated duration groups of LongVideoBench and Video-MME; overall accuracy weights groups by question count.

FORTE leads overall on both benchmarks. On LongVideoBench, it leads the 1 h group, where its gain over Uniform is largest, and ties for best in the 15 s group; other baselines lead the 1 min and 10 min groups. On Video-MME, FORTE leads the short and long groups, while BOLT leads the medium group. The strongest benefits thus occur on long videos, with gains also present in some shorter-duration groups.

Table 14: QA accuracy (%) by video duration on LongVideoBench and Video-MME, using Qwen3-VL-8B, BLIP, and $K = 64$ input frames. Scored methods use $\rho = 0.5$, except FOCUS with an effective budget of approximately 0.44; Uniform requires no relevance scoring. Overall accuracy is the question-weighted average across duration groups. Bold marks the best results in each accuracy column, including ties.

Method	ρ	LongVideoBench					Video-MME			
		15 s	1 min	10 min	1 h	Overall	Short	Medium	Long	Overall
Uniform	0	78.3	76.2	62.1	53.0	62.4	79.8	67.8	55.7	67.7
Top-K	0.5	**79.9**	**79.7**	64.1	58.0	65.7	78.7	67.0	58.8	68.1
AKS	0.5	**79.9**	**79.7**	64.8	56.0	65.1	78.7	70.1	58.8	69.0
BOLT	0.5	64.6	70.9	65.5	60.3	63.9	78.7	**70.4**	59.1	69.4
Q-Frame	0.5	**79.9**	**79.7**	65.3	53.9	64.4	78.7	66.7	57.4	67.6
AdaQ	0.5	**79.9**	**79.7**	64.1	59.0	66.2	78.7	67.3	58.9	68.3
WFS-SB	0.5	**79.9**	**79.7**	66.5	55.7	65.5	78.7	67.6	58.9	68.4
FOCUS	$\sim$0.44	76.2	72.1	65.0	56.9	64.1	75.2	64.1	56.7	65.3
KFC	0.5	**79.9**	**79.7**	**68.4**	54.3	65.5	78.7	65.9	58.9	67.8
Ours	0.5	**79.9**	76.2	67.2	**62.4**	**68.1**	**81.2**	69.1	**59.4**	**69.9**

D.9 Scoring-Budget Regimes

When $B < K$, scored-pool selectors require supplementation to supply K frames (Appendix C.2); when $B \geq K$, selection can remain within the scored pool if enough usable candidates are available. We define these groups using the integer budget in Appendix C and apply them to every method, including Uniform and FOCUS despite their different scoring protocols.

Prevalence of $B < K$. The fraction of questions in this regime decreases from 52.5% at $\rho = 0.1$ to 16.2% at 0.5 and 8.0% at 1.0 (Table 15). At the default budget, only LongVideoBench and Video-MME contain such questions. These are budget-defined counts, not supplementation rates: completion may still be needed when $B \geq K$ if fewer than K usable candidates remain.

Table 15: Questions with a scoring budget smaller than the final frame budget, $B < K$, for $K = 64$. Each entry gives the question count and its percentage within the corresponding benchmark. Total percentages use all 7,760 evaluated questions as the denominator.

Benchmark	$\rho = 0.1$	$\rho = 0.5$	$\rho = 1.0$
LongVideoBench	773 (57.8%)	361 (27.0%)	361 (27.0%)
Video-MME	1,554 (57.6%)	897 (33.2%)	258 (9.6%)
LVBench	0	0	0
MLVU	1,746 (80.3%)	0	0
Total	**4,073 (52.5%)**	**1,258 (16.2%)**	**619 (8.0%)**

Supplemented frame inputs. A separate audit at $\rho = 0.5$ and $K = 64$ uses the approximation algorithm for final selection. For evaluated questions $\mathcal{Q}$, let $n_{\text{sup}}(q)$ count supplemental frames for question q. Their fraction of all input frames is

$$f_{\text{sup}} = \frac{\sum_{q \in \mathcal{Q}} n_{\text{sup}}(q)}{K|\mathcal{Q}|}. \tag{36}$$

The denominator includes every evaluated question. A frame reused across questions is counted once per input; supplemental frames receive no additional relevance scoring.

Supplementation occurs only on LongVideoBench and Video-MME (Table 16). Pooled across questions, 8.13% of input frames are supplemental and 91.87% are scored. Equal benchmark weighting instead gives a supplemented fraction of 8.64%.

Table 16: Composition of FORTE's final frame inputs at $\rho = 0.5$ and $K = 64$, using the approximation algorithm for final selection. Counts include all evaluated questions. Supplemented and scored percentages partition the final frame inputs; Pooled sums the counts across benchmarks before computing the percentages.

Benchmark	Supplemental / total frame inputs	Supplemented (%)	Scored (%)
LongVideoBench	18,943 / 85,568	22.14	77.86
Video-MME	21,435 / 172,800	12.40	87.60
LVBench	0 / 99,136	0.00	100.00
MLVU	0 / 139,136	0.00	100.00
Pooled	40,378 / 496,640	8.13	91.87

Accuracy within each regime. Within $B \geq K$, FORTE achieves the highest observed four-benchmark mean accuracy at both scoring budgets (Table 17). At $\rho = 0.1$, it leads LongVideoBench, Video-MME, and LVBench, while KFC leads MLVU; at $\rho = 0.5$, it leads all four benchmarks. Thus, the observed mean advantage is not confined to questions whose scoring budget is too small to supply K input frames.

Results within $B < K$ are more mixed. FORTE has the highest observed accuracy on MLVU at $\rho = 0.1$ and Video-MME at $\rho = 0.5$, while Top-K, AKS, WFS-SB, and KFC tie for the highest accuracy on LongVideoBench at both budgets and Video-MME at $\rho = 0.1$. For the budget-matched score-based methods, final inputs necessarily include unscored supplemental frames, so accuracy reflects scored-pool construction together with the completion rule. Because group membership differs across regimes and scoring budgets, differences in their absolute accuracies do not isolate the effect of supplementation or additional scoring.

D.10 Further Mechanism Controls

Each variant in Table 18 changes only the named component of default FORTE, with variants grouped by stage.

Table 17: QA accuracy (%) by scoring-budget regime at $\rho \in \{0.1, 0.5\}$ and $K = 64$, using Qwen3-VL-8B and BLIP. FORTE uses the exact algorithm for final optimization. Within each budget block, all methods share identical question groups defined by the corresponding integer scoring budget. Uniform requires no scoring. †FOCUS is reported only in the $\rho = 0.5$ block at its effective allowance of approximately 0.44 and is not budget-matched. Avg. is the unweighted four-benchmark mean within $B \geq K$. The $B < K$ group contains no LVBench questions at either budget and no MLVU questions at $\rho = 0.5$, so no four-benchmark mean is reported for it. Bold marks the highest observed accuracy in each row, including ties.

Regime	Benchmark	Questions	FORTE	Uniform	Top-K	AKS	WFS-SB	KFC	FOCUS†
Scoring budget $\rho = 0.1$									
$B < K$	LongVB	773	70.2	69.2	**70.4**	**70.4**	**70.4**	**70.4**	–
	Video-MME	1,554	74.5	74.7	**75.2**	**75.2**	**75.2**	**75.2**	–
	MLVU	1,746	**73.3**	71.9	71.8	71.8	71.8	71.8	–
$B \geq K$	LongVB	564	**57.6**	53.0	53.9	53.4	53.4	51.6	–
	Video-MME	1,146	**62.0**	58.3	60.0	60.9	59.0	58.8	–
	LVBench	1,549	**49.0**	41.0	46.6	45.6	46.9	47.6	–
	MLVU	428	68.7	63.8	66.9	67.7	68.8	**69.4**	–
	Avg.	–	**59.3**	54.0	56.9	56.9	57.0	56.9	–
Scoring budget $\rho = 0.5$									
$B < K$	LongVB	361	78.1	77.3	**79.8**	**79.8**	**79.8**	**79.8**	74.2
	Video-MME	897	**81.3**	79.8	78.7	78.7	78.7	78.7	75.4
$B \geq K$	LongVB	976	**64.4**	56.9	60.5	59.7	60.2	60.2	60.3
	Video-MME	1,803	**64.3**	61.7	62.8	64.2	63.2	62.4	60.3
	LVBench	1,549	**50.7**	41.0	50.6	47.8	50.6	46.4	48.7
	MLVU	2,174	**75.7**	70.3	73.8	74.5	74.7	74.9	71.3
	Avg.	–	**63.8**	57.5	61.9	61.5	62.2	61.0	60.2

Stage 1: Adaptive relevance scoring. Replacing GP prediction with linear interpolation or the nearest observed score reduces accuracy on all four benchmarks.

The zero-mean GP sets $\mu_{\text{obs}} = 0$ in Equation 2 instead of the observed average. Centering improves every benchmark and raises mean accuracy from 65.7% to 66.1%, supporting a nonzero baseline in regions without nearby observations.

The round-frozen variant holds coverage gains fixed throughout each batch rather than reducing them around newly chosen frames. Its lower accuracy supports frame-by-frame coverage updates even when relevance predictions remain fixed.

Stage 2: Global keyframe optimization. Both controls keep the scored pool and measured relevance fixed. The square-root variant replaces the logarithmic gap reward with a scaled square-root reward, matching the marginal gain for midpoint insertion into a gap of length τ_K without changing the coverage weight. The other variant removes boundary-gap rewards, retaining only gaps between selected frames. Both reduce accuracy on all four benchmarks, supporting the chosen reward and boundary treatment.

Table 18: Controls for the two stages of FORTE at $\rho = 0.5$ and $K = 64$. Ours uses the centered GP with the observed average as its prior mean. Variants are grouped by the stage they modify, with all other settings fixed. Values are accuracy (%).

Variant	LongVB	Video-MME	LVBench	MLVU	Avg.
Ours (centered GP)	**68.1**	**69.9**	**50.7**	**75.7**	**66.1**
Stage 1: Adaptive relevance scoring					
Linear interpolation	66.3	68.8	50.2	74.8	65.0
Nearest observed score	66.3	68.5	49.6	74.7	64.8
Zero-mean GP	67.9	69.8	50.0	75.2	65.7
Round-frozen coverage	65.1	69.0	49.8	74.6	64.6
Stage 2: Global keyframe optimization					
Matched square-root potential	66.9	68.5	49.5	74.9	64.9
No boundary-gap rewards	67.4	68.6	49.9	74.2	65.0

D.11 Comparison with Smooth Relevance Predictors

Table 19 replaces the GP mean predictor at $\rho \in \{0.1, 0.5\}$ while keeping initialization, batching, coverage updates and weights, normalization, completion, and final optimization fixed. Variants share initial observations but subsequently acquire frames according to their own predictions.

The primary control is normalized RBF kernel regression, which predicts relevance by averaging observed scores with temporal kernel weights:

$$\mu_{\mathrm{RBF}}(t) = \frac{\sum_{t_i \in \mathcal{O}} k(t, t_i) r_i}{\sum_{t_i \in \mathcal{O}} k(t, t_i)}. \tag{37}$$

Here, k is the kernel in Equation 1 and r_i the normalized observed score. RBF regression requires no kernel-system solve and shares the GP's fixed $\ell = 10$ seconds, without predictor-specific tuning. We also evaluate shape-preserving piecewise cubic Hermite interpolation (PCHIP), complementing the simpler controls in Table 18.

Accuracy and cost. The GP leads every benchmark at both budgets, improving mean accuracy over RBF regression by 0.7 points at $\rho = 0.5$ and 1.1 points at $\rho = 0.1$. The larger gain with fewer observations suggests that prediction quality matters more under sparse scoring. RBF regression and PCHIP are faster: the GP adds 0.28–0.41 seconds over RBF regression while remaining below one second of fitting-and-prediction time per question.

Table 19: Comparison of relevance predictors at two scoring budgets, with $K = 64$ and global keyframe optimization. Only the predictor used during adaptive scoring changes. GP prediction and RBF regression share the same RBF kernel and fixed length scale, $\ell = 10$ seconds. Accuracy is reported in percent; Avg. is the unweighted four-benchmark mean. Time is the mean total fitting-and-prediction time across all scoring rounds per video–question pair, excluding scorer and MLLM inference.

Predictor	LongVB	Video-MME	LVBench	MLVU	Avg.	Time (s)
Scoring budget $\rho = 0.5$						
FORTE (GP mean)	68.1	69.9	50.7	75.7	66.1	0.81
RBF regression	67.0	69.3	50.1	75.1	65.4	0.40
PCHIP interpolation	66.8	69.0	50.3	75.0	65.3	0.21
Scoring budget $\rho = 0.1$						
FORTE (GP mean)	64.9	69.2	49.0	72.0	63.8	0.59
RBF regression	63.7	68.8	47.4	71.0	62.7	0.31
PCHIP interpolation	63.8	68.9	47.5	71.4	62.9	0.18

D.12 Posterior Uncertainty versus Temporal Coverage

Relation to GP-based sequential acquisition. The adaptive-scoring stage is related to Gaussian-process-based Bayesian optimization and sequential experimental design, which use posterior predictions to choose subsequent evaluations. A common strategy, including GP-UCB-style acquisition, promotes exploration through posterior uncertainty. FORTE instead uses the GP posterior mean to estimate frame relevance and introduces exploration explicitly through marginal temporal coverage. The comparison below isolates these choices under the same predictor and scoring budget.

Table 20 tests posterior uncertainty as an alternative or addition to coverage at $\rho \in \{0.1, 0.5\}$. The four rules use GP mean only, mean plus posterior standard deviation (GP-UCB-style acquisition), mean plus coverage (FORTE), or all three terms. All share $\mathcal{U}$, B, initial observations, the GP mean predictor, four-round schedule, normalization, completion, and exact final optimizer; only acquisition changes.

Uncertainty-aware acquisition. The uncertainty bonus is $\lambda\sigma(t)$, where $\sigma(t)$ is the GP posterior standard deviation of relevance at an unscored frame. Before batching or numerical approximation, its reference definition is

$$\sigma^2(t) = k(t,t) - \mathbf{k}(t)^\top (\mathbf{K} + \epsilon \mathbf{I})^{-1} \mathbf{k}(t). \tag{38}$$

We retain FORTE's unit-amplitude RBF kernel and $\ell = 10$ seconds, set $\lambda = 1$, and use $\beta_{\text{acq}} = 3.5$ and $\tau_B = T/(B+1)$ when coverage is included. Within each batch, the mean remains fixed while uncertainty is conditioned on each newly chosen frame's location, without requiring its unobserved score. Coverage also updates after every choice. Both exploration mechanisms can therefore discourage redundant within-batch choices.

Table 20: Exploration mechanisms for adaptive relevance scoring, with $K = 64$ and the same exact final optimizer. Uncertainty denotes $\lambda\sigma(t)$ with $\lambda = 1$; coverage denotes $\beta_{\text{acq}}\Delta C_{\tau_B}$ with $\beta_{\text{acq}} = 3.5$. Accuracy is reported in percent; Avg. weights the four benchmarks equally. Bold marks the highest accuracy in each budget block. Acquisition time includes GP fitting, prediction, and batch construction, excluding video decoding, scorer inference, final optimization, and MLLM inference. Times are per video–question pair, averaged within benchmarks and then equally across benchmarks.

Acquisition rule	LongVB	Video-MME	LVBench	MLVU	Avg.	Time (s)
Scoring budget $\rho = 0.1$						
Mean only	63.1	68.0	47.5	71.4	62.5	0.60
Mean + uncertainty	63.6	68.4	47.8	71.8	62.9	0.95
Mean + coverage (FORTE)	**64.9**	**69.2**	49.0	72.0	**63.8**	0.65
Mean + uncertainty + coverage	64.8	69.1	**49.1**	**72.1**	**63.8**	1.00
Scoring budget $\rho = 0.5$						
Mean only	66.4	67.9	49.6	73.0	64.2	0.80
Mean + uncertainty	67.0	68.4	49.8	73.6	64.7	1.30
Mean + coverage (FORTE)	**68.1**	**69.9**	**50.7**	**75.7**	**66.1**	0.83
Mean + uncertainty + coverage	68.0	69.8	**50.7**	75.6	66.0	1.33

Accuracy. Uncertainty improves on mean-only acquisition on every benchmark, but coverage yields larger gains at both budgets. Adding uncertainty to coverage leaves mean accuracy unchanged at $\rho = 0.1$ and slightly lower at 0.5. Thus, at the tested setting, uncertainty helps without coverage but adds no mean accuracy benefit once coverage is present.

Acquisition cost. Adding uncertainty to FORTE increases acquisition time from 0.65 to 1.00 seconds at $\rho = 0.1$ and from 0.83 to 1.33 seconds at 0.5, without improving mean accuracy. Coverage alone therefore offers the better observed accuracy–cost trade-off.

E Extended Analysis of Related Work

This section supplements Section 2 with a broader overview of long-video modeling and keyframe selection, focusing on their connections to adaptive scoring and global keyframe optimization.

E.1 Video LLMs for Long-Video Understanding

Video LLMs such as LLaVA-OneVision, LLaVA-Video, Qwen3-VL, and InternVL3.5 combine visual representations with language reasoning (Li et al., 2025; Zhang et al., 2025c; Bai et al., 2025; Wang et al., 2025b). To extend these capabilities to long videos, LongVILA scales the supported context (Chen et al., 2025), while MovieChat, MA-LMM, and VideoStreaming retain historical information through memory mechanisms (Song et al., 2024; He et al., 2024; Qian et al., 2024). LongVLM, LongVU, Video-XL, and VideoChat-Flash use hierarchical representations or visual compression (Weng et al., 2024; Shen et al., 2025; Shu et al., 2025; Li et al., 2026b); FLoC further explores facility-location-based token compression (Cho et al., 2026). These approaches improve how video information is represented and retained. Keyframe selection complements them by determining which frames form the model's visual input.

E.2 Keyframe Selection Methods

Training-based selection. SeViLA, Frame-Voyager, and ReFocus learn selection policies from supervision or feedback (Yu et al., 2023; 2025; Lee et al., 2026). TimeSearch-R, VideoZoomer, and LongVideo-R1 use reinforcement learning to learn temporal search policies (Pan et al., 2026; Ding et al., 2026; Qiu et al., 2026). In contrast, FORTE operates with frozen models and requires no additional selector training.

LLM-guided evidence search. VideoAgent coordinates vision–language tools for iterative retrieval (Wang et al., 2024), while VideoTree and DrVideo organize captioned or text-based evidence for question answering (Wang et al., 2025c; Ma et al., 2025). A.I.R. and DIG guide evidence selection through VLM analysis or MLLM feedback (Zou et al., 2026; Li et al., 2026a). Such workflows use semantic judgments beyond frame–query similarity, but their additional captioning, analysis, and reasoning calls can be computationally expensive. FORTE instead uses a lightweight relevance scorer without invoking the downstream answer model during selection.

E.2.1 Score-Based Frame Selection

Relevance-based selection. Pretrained vision–language models such as CLIP and BLIP provide frame–query relevance scores without task-specific training (Radford et al., 2021; Li et al., 2022). KeyVideoLLM uses text–frame matching to select relevant frames (Liang et al., 2024). Beyond Top-K selection, BOLT samples according to temporal relevance mass, Q-Frame combines relevance-based sampling with resolution adaptation, and AdaQ adjusts its sampling distribution using score statistics (Liu et al., 2025; Zhang et al., 2025a; 2026b). These methods refine the selection of input frames from scored candidates; when that candidate pool is fixed in advance, improving the final selection rule cannot recover evidence omitted before scoring.

AKS balances relevance with temporal coverage, while TCS combines relevance-focused clips with sparse global context (Tang et al., 2025; Tan et al., 2026). WFS-SB (Chen et al., 2026) detects semantic boundaries in the relevance trajectory and selects diverse frames within the resulting segments. Nar-KFC incorporates visual diversity (Fang et al., 2026). Together, these methods highlight the importance of selecting complementary evidence rather than relying on relevance alone. FOCUS allocates relevance evaluations using clip-level mean relevance and confidence bounds (Zhu et al., 2026). The concurrent preprint CSES (Zhang et al., 2026a) formulates keyframe selection as a 0–1 integer programming problem with a complex optimization objective. Rather than providing an exact polynomial-time solver, CSES employs a greedy approximation algorithm, which does not guarantee a globally optimal keyframe set with respect to its proposed objective.

FORTE jointly addresses where to spend a limited scoring budget and which scored frames to retain. Its adaptive scoring stage combines frame-level relevance predictions with marginal temporal coverage to construct the candidate pool. Its global optimization stage then balances measured relevance and temporal coverage through a logarithmic gap objective, for which we derive an exact $\mathcal{O}(KN)$ algorithm (Section 3.3). Our contribution lies in coupling adaptive candidate acquisition with efficient, exact optimization of the final input.

Scope of experimental comparisons. CSES was first submitted to arXiv on August 1, 2026 (Zhang et al., 2026a), and we could not locate a publicly available official implementation at the time of

writing. We cite it as concurrent work without including it in our experimental baselines, consistent with the ICLR 2027 reviewer guidelines.[9]

F Qualitative Analysis

We compare Uniform, Top-K, AKS, FOCUS, and FORTE using their selected frames and saved answers from Qwen3-VL-8B-Instruct. All methods supply $K = 64$ frames; Top-K, AKS, and FORTE use BLIP with a nominal scoring budget of $\rho = 0.5$, while FOCUS follows its official scoring procedure. FORTE uses the exact algorithm for global keyframe optimization. The following cases distinguish acquiring evidence in the scored pool from retaining it in the final input, and include a shared failure despite retained evidence. They are selected examples rather than an estimate of how frequently each behavior occurs.

Preserving brief evidence. Figure 6(a) asks who appears after a reporter's spoken cue. FORTE retains the subsequent courtroom shot, whereas Top-K, AKS, and FOCUS omit the specified follow-up interval. Importantly, Top-K and AKS had scored frames from this interval but did not retain them: making evidence available in the candidate pool does not ensure that it survives final selection. FOCUS instead retains several similar views of the reporter before the transition. FORTE and Uniform both preserve the follow-up and answer correctly, illustrating the value of maintaining temporal coverage when the most visually prominent subject is not the answer.

Figure 6(b) illustrates a related distinction between temporal proximity and readable evidence. The question asks which object appears at 16:46 in a roughly 42-minute video. FORTE retains a nearby view showing the keyboard and its stand, and is the only method to answer correctly. Top-K and AKS retain an earlier frame in which only an edge of the instrument is visible, while Uniform and FOCUS omit the shot. Thus, selecting a frame near the requested time is not equivalent to retaining a view from which the object can be identified.

Retaining a readable view. Figure 6(c) asks for the color of a stuffed toy in a briefly displayed photograph. FORTE and Top-K retain the full-size photograph, in which the red toy is readily visible, and both answer correctly. AKS retains a nearby view of the presenters with a smaller inset of the photograph, while FOCUS skips the full-size image; both answer incorrectly. Uniform also answers correctly despite having no frame in the narrow full-size-photo interval. The comparison illustrates a difference in the visibility of the queried detail, rather than establishing that this particular view is necessary for a correct answer.

Linking distant scenes. Figure 6(d) requires matching a man identified in a restroom to another location in the video. FORTE retains both the reference appearance and direct views of the man on the beach more than nine minutes later. Top-K, AKS, and FOCUS retain the restroom reference but omit the annotated shore sequence. AKS and FOCUS instead retain nearby suspension-bridge views and answer "suspension bridge," whereas FORTE and Uniform answer "beach." AKS and FOCUS also retain later beach-adjacent scenes, so the distinction is not the complete absence of beach imagery: FORTE preserves the direct matching scene together with the reference appearance.

Acquiring evidence absent from the scored pool. The preceding comparisons primarily concern which evidence survives final selection. Figure 6(e–f) instead examines evidence that uniform scoring never observes. We inspect the scored pools as well as the final inputs, checking whether a frame shows the queried gesture or object detail rather than merely falling within the annotated interval. Top-K, AKS, and FORTE have matched scoring counts in each case, with $B > K$; all five methods return 64 frames without supplementation.

[9] The guidelines encourage citation of relevant contemporaneous work but do not require experimental comparisons; see `https://iclr.cc/Conferences/2027/ReviewerGuidelines`.

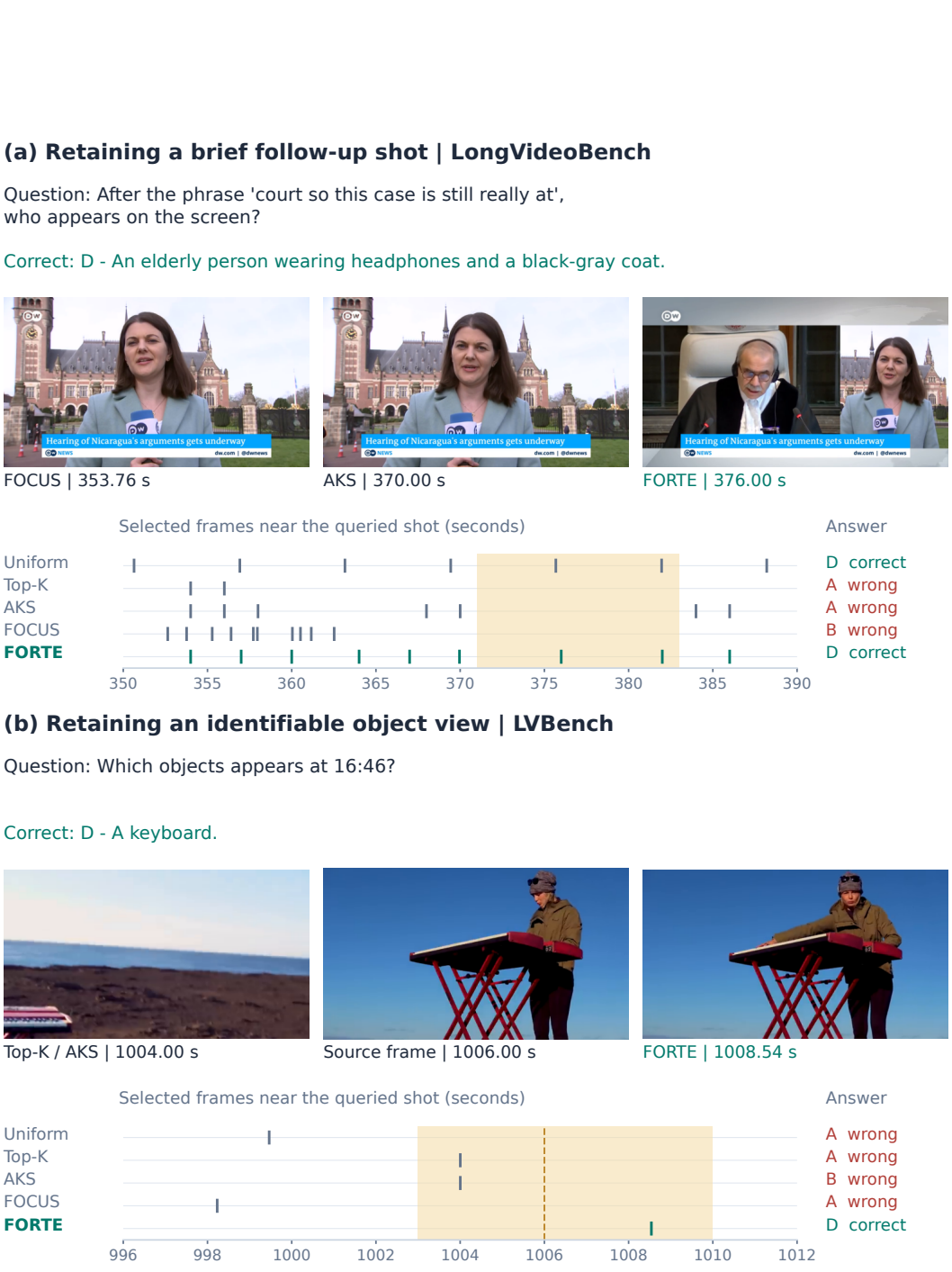


Figure 6: Brief evidence may be omitted from the final input or shown only partially. Each tick marks a selected frame within the displayed time range; shading highlights the queried interval or its surrounding shot. (a) FORTE and Uniform retain the courtroom follow-up at 371–383 seconds. (b) FORTE retains a clear keyboard view near the requested timestamp (dashed line). The middle image in (b) is a source-video reference, not a selected input; all other images are selected-frame excerpts.

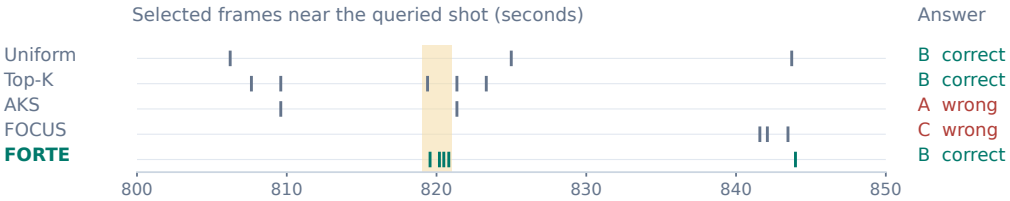
(c) Retaining a briefly displayed photograph | LongVideoBench

Question: When the subtitle mentions 'those and the word balikbayan means,' what color is the stuffed toy held by the woman in the blue jacket?

Correct: B - Red.



AKS | 821.36 s FOCUS | 841.59 s FORTE | 820.19 s



(d) Connecting a person across distant scenes | LongVideoBench

Question: Where else does the gray-shirted man with a backpack, shown giving a thumbs-up in the restroom, appear?

Correct: E - On the beach.



FORTE | 159.12 s AKS | 721.39 s FORTE | 727.06 s

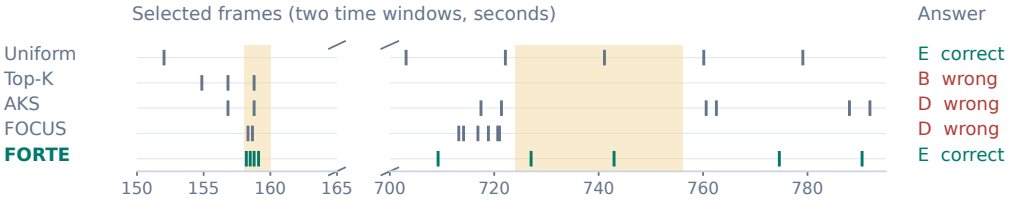
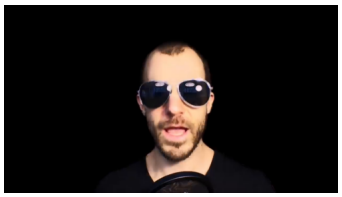


Figure 6: Additional baseline comparisons (continued). (c) The full-size photograph makes the toy's color readily visible; Top-K also retains this view and answers correctly. (d) FORTE retains the restroom reference and the later shore sequence. The broken timeline shows two separate time windows with different scales; shading marks the annotated intervals. All images are selected-frame excerpts, and all ticks represent selected frames within the displayed windows.

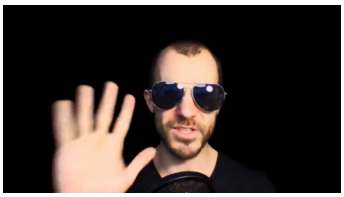
(e) Acquiring a gesture between uniform samples | LongVideoBench

Question: In front of a pure black background, there is a man with cornrows. He is wearing a black coat and sunglasses. What did he do when he first appeared?

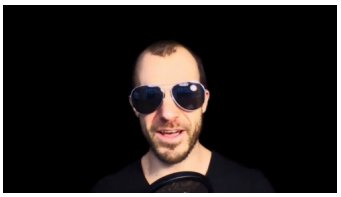
Correct: C - He raised his right hand. B = 1,567; K = 64



Uniform-scored
102.00 s



FORTE: acquired + retained
102.37 s



Uniform-scored
104.00 s

Answer-relevant evidence in:	Scored pool	Final input	Answer
Uniform	Not scored	No	A wrong
Top-K	No	No	B wrong
AKS	No	No	B wrong
FOCUS	Yes	No	B wrong
FORTE	Yes	Yes	C correct

Evidence absent from initialization; acquired during adaptive scoring and retained in the final input.

(f) Acquiring an object detail between uniform samples | LongVideoBench

Question: On a white shelf, there is a pair of mismatched shoes, and next to the shoes, there's a wooden-colored bag. What kind of fastener does the bag in the video have?

Correct: B - A silver square fastener. B = 604; K = 64



Uniform-scored
452.83 s



FORTE: acquired + retained
454.20 s



Uniform-scored
454.79 s

Answer-relevant evidence in:	Scored pool	Final input	Answer
Uniform	Not scored	No	C wrong
Top-K	No	No	C wrong
AKS	No	No	C wrong
FOCUS	No	No	C wrong
FORTE	Yes	Yes	B correct

Evidence absent from initialization; acquired during adaptive scoring and retained in the final input.

Figure 6: Evidence acquisition and final retention (continued). FORTE captures (e) a brief gesture and (f) a bag's fastener between uniform-scored frames, and is the only method to answer correctly in both cases. The tables distinguish visually usable evidence in the scored pool from evidence retained in the final input, alongside each method's answer. Uniform selects its final input without scoring. Top-K, AKS, and FORTE each score 1,567 frames in (e) and 604 in (f); FOCUS scores 1,369 and 509, respectively. All methods retain 64 frames without supplementation. Uniform-scored images are source-video frames; FORTE images are reconstructed selected inputs.

In Figure 6(e), the question concerns the man's first appearance in a 52-minute video. Uniform scoring captures frames at 102 and 104 seconds, but neither shows his brief right-hand raise between them. FORTE's initialization also misses the gesture; adaptive scoring captures it at 102.37 seconds, and final optimization retains this frame. FOCUS scores a view of the same gesture but discards it, instead retaining a later clenched fist. FORTE is the only method to answer correctly: Uniform answers that the man takes off his sunglasses, while Top-K, AKS, and FOCUS answer that he clenches his right hand into a fist. The comparison thus separates two mechanisms: FORTE acquires an event absent from the uniformly scored pool and preserves evidence that FOCUS loses during final selection.

Figure 6(f) concerns the fastener on a bag displayed beside a pair of shoes. Uniform scoring captures the presenter at 452.83 seconds and a different, zippered backpack at 454.79 seconds, missing the intervening view of the queried fastener. Although the latter frame falls within the annotated interval, it does not show the required detail. FORTE's initialization also misses this evidence; adaptive scoring acquires a clear view at 454.20 seconds, which final optimization retains. FORTE correctly identifies a silver square fastener, whereas Uniform, Top-K, AKS, and FOCUS all answer "silver round fastener." The usable detail is absent from the uniformly scored pool and FOCUS's scored pool, as well as all four baselines' final inputs. Together, these cases illustrate evidence acquisition between uniform samples within the video: final optimization cannot recover visual evidence that was never included in the scored pool.

A shared failure despite retained evidence. Figure 7 asks about the woman's wedding dress. All five methods retain views showing her black dress, yet every answer is white. Several attendants wear white, making it important to distinguish the queried woman from nearby people. The failure therefore cannot be explained solely by missing views of the relevant clothing. It highlights a remaining challenge beyond evidence retention: the downstream model must associate the question with the correct visual subject and use that evidence to answer. The saved responses alone do not establish whether subject confusion or another aspect of the model's reasoning caused the error. Together with the successful cases, this example distinguishes the quality of the selected evidence from the model's ability to interpret it.

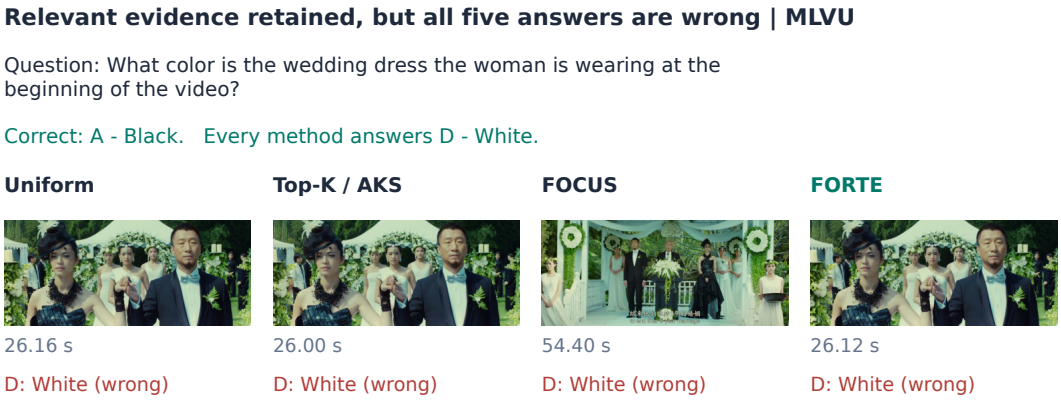


Figure 7: A shared failure on MLVU. The selected inputs visibly contain the black dress, but every method produces the incorrect answer "white." Top-K and AKS select the same displayed frame. Retaining relevant evidence does not by itself guarantee that the downstream MLLM uses it correctly.