FrameScout: Scouting Query-Relevant Frames for Long Video Understanding

Haonan Hu[1,4]* **Shuhao Chen**[1,2]* **Weisen Jiang**[3]* **Lizhao Gao**[5] **James Kwok**[2] **Yu Zhang**[1]†
[1]Southern University of Science and Technology
[2]The Hong Kong University of Science and Technology
[3]South China University of Technology
[4]Weixin, Tencent Inc.
[5]Independent Researcher

Abstract

Long video understanding requires VideoLLMs to answer queries about videos spanning tens of minutes to hours, but such long videos produce far more visual tokens than current VideoLLMs can process. To address this issue, keyframe selectors are proposed to select the most query-relevant frames by computing the similarity between frame and query embeddings. Existing selectors either encode frames independently without temporal context or process chunks in isolation with ranking objectives that are difficult to optimize. To alleviate those limitations, we propose **FrameScout** with two core designs: (i) a *streaming selector architecture* that leverages a sliding-window KV cache and successor aggregation to produce temporally-contextualized frame embeddings; (ii) a *frame-query contrastive objective* that aligns frame and query embeddings in a shared space, directly training the selector to distinguish relevant frames from irrelevant ones. Extensive experiments on three long-video benchmarks and six base VideoLLMs demonstrate that FrameScout consistently achieves state-of-the-art performance. Code is available at https://github.com/Hubert-1102/FrameScout.

1 Introduction

Long video understanding requires Video Large Language Models (VideoLLMs) [12, 1, 2, 73, 43] to answer queries about videos spanning tens of minutes to hours, a task with broad applications from question answering [64, 41, 18] to temporal reasoning [39, 7, 34]. However, this task poses a fundamental challenge due to the large number of visual tokens, which grows linearly with video length and incurs quadratic attention cost. For example, a 20-minute video sampled at one frame per second (256 visual tokens per frame under the Qwen2.5-VL encoder at 448×448 resolution) yields $\sim$307K tokens, exceeding the context window of many VideoLLMs (e.g., 32K tokens for Qwen2.5-VL-7B [2]). Even for recent models with substantially longer contexts (e.g., up to 1M tokens for Qwen3-VL [1]), an hour-scale video at 1 fps saturates the window, and densely prefilling such long contexts incurs prohibitive computation, memory, and latency.

Uniform downsampling, which reduces the number of frames to fit within the context window, is a naive strategy but inevitably discards query-relevant frames. This motivates the **keyframe selection** problem: learning a selector that identifies the most relevant frames for each query. An effective keyframe selector should satisfy two core requirements: (i) building *temporally-contextualized frame embeddings* that capture full video context, since determining whether a frame is relevant to a query

*Equal contribution. Work by Haonan Hu was done during an internship at Weixin.
†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

often requires understanding the temporal context surrounding it; and (ii) learning a *frame-query alignment* mechanism that accurately selects query-relevant frames from these embeddings.

Existing keyframe selection methods [54, 69, 66, 68, 59, 16, 75, 72] fall short of satisfying these two requirements. For example, FOCUS [75] adopts a *frozen* vision-language encoder (e.g., BLIP [33]) to score each frame *independently* against the query in CLIP space [47], so the resulting scores lack temporal context across frames, and the frozen encoder provides only surface-level frame-query alignment that is insufficient for long-video understanding. Recently, FlexSelect [72] improved upon this by partitioning the video into non-overlapping chunks and training a selector with a Spearman rank-correlation loss [5]. However, chunks are still processed in isolation, leaving frame representations unaware of cross-chunk temporal context. The rank-correlation objective supervises full relative ordering among all frames rather than directly aligning frame and query embeddings in a shared space, making it both difficult to optimize and misaligned with the keyframe selection task.

In this paper, we propose **FrameScout**, a keyframe selection framework that *scouts* query-relevant frames from long videos, addressing the above limitations with two core designs: (i) a **streaming selector architecture** that builds temporally-contextualized frame embeddings across the entire video. It combines a sliding-window KV cache that propagates past context across chunk boundaries with a successor aggregation mechanism that incorporates future context, requiring only *constant* KV-cache memory and *linear* computation. (ii) a **frame-query contrastive objective** that aligns frame and query embeddings in a shared space via contrastive learning, which pushes relevant frame embeddings closer to the query embedding and pulls irrelevant ones away, directly training the selector to distinguish relevant frames from irrelevant ones. Across three long-video benchmarks (i.e., VideoMME [18], MLVU [74], LongVB [62]) and six base VideoLLMs (both closed- and open-source), FrameScout consistently establishes state-of-the-art performance, e.g., $+6.5\%$ on VideoMME-Long with Qwen3-VL-8B [1], while achieving a $6\times$ speedup in multi-query scenarios.

The main contributions of this paper are as follows:

- We propose FrameScout, a keyframe selection framework built on a streaming selector architecture that produces temporally-contextualized frame representations via a sliding-window KV cache and successor aggregation, under constant KV-cache memory and linear computation in video length.
- We design a frame-query contrastive objective that aligns frame and query embeddings in a shared space, directly training the selector to distinguish query-relevant frames from irrelevant ones.
- Extensive experiments on three long-video understanding benchmarks and six base VideoLLMs demonstrate that FrameScout consistently outperforms the current state-of-the-art selectors FOCUS [75] and FlexSelect [72] with a $6\times$ speedup in multi-query scenarios.

2 Related Works

Keyframe Selection for Long Video Understanding. Early Video Large Language Models (VideoLLMs) [73, 12, 1, 2] extend image-based multimodal LLMs by densely encoding a small number of sampled frames, yielding strong performance on short video benchmarks but scaling poorly with video duration. Since long videos produce far more visual tokens than current VideoLLMs can process, selecting a compact set of query-relevant keyframes has emerged as a practical strategy. Existing keyframe selection methods can be grouped into three families. (i) *Heuristic methods*, such as uniform sampling and visual-similarity clustering, are training-free but often miss important moments and cannot condition on the query [54]. (ii) *Similarity-based methods* leverage frozen pre-trained encoders to score frames by cross-modal matching. For example, FOCUS [75] uses CLIP [47]/BLIP [33] for text-image similarity, and Adaptive Greedy [54] combines SigLIP [70] and DINOv2 [44] under a submodular objective to balance relevance and coverage. BOLT [38] applies inverse transform sampling to CLIP query-frame similarities, and F2C [53] extends frame selection to key clips with adaptive resolution under a fixed token budget. Although efficient, these methods score each frame *independently* in a *frozen* embedding space: frame representations carry no temporal context, and the pretrained CLIP-style alignment captures only surface-level concept overlap, insufficient for long-video QA. (iii) *Learning-based methods* [69, 72, 22, 66] like FlexSelect [72] train a lightweight selector to rank frames according to their relevance to the query. However, they partition the video into non-overlapping chunks processed in isolation, leaving frame representations blind to cross-chunk temporal context. Their ranking objectives (e.g., the Spearman rank-correlation loss

used by FlexSelect [72]) also entangle frame encoding with the query, requiring the entire selector to be re-run for every new query.

The proposed FrameScout differs from the above methods in three respects: (i) frame representations capture cross-chunk temporal context rather than encoding frames or chunks in isolation; (ii) the training objective targets relevant-vs-irrelevant separation rather than full ranking; (iii) frame embeddings are query-agnostic rather than entangled with query encoding.

Efficient Long-Context Modeling. Processing long sequences under bounded resources has been extensively studied. Sliding-window attention [4, 23] restricts each token to attend only to a fixed-size local neighborhood. StreamingLLM [63] further identifies the *attention sink* phenomenon and retains a small set of initial tokens alongside the sliding window to stabilize the attention distribution, enabling unbounded generation under a bounded KV cache; subsequent work [19, 8] extends this idea with hierarchical caches and dynamic eviction. These techniques have been developed primarily for text-only LLMs; FrameScout is the first to adopt sliding-window KV caching for keyframe selection, enabling constant-memory streaming over arbitrarily long videos.

However, sliding-window caching is inherently *causal*: each position only sees past context, but keyframe selection requires temporally-contextualized frame representations that also capture future context. Existing methods such as bidirectional encoders [15, 49] provide full-context awareness but require KV-cache memory that grows linearly with sequence length, making them impractical for long videos. To incorporate future context without breaking the streaming property, FrameScout draws on the idea of successor representations in reinforcement learning [14, 3], which express a state's value as an exponentially-discounted aggregation of future rewards. Unlike bidirectional encoders that require memory linear in sequence length, this *successor aggregation* accumulates an exponentially-decayed summary of all subsequent frame embeddings via a single backward pass, achieving bi-directional contextualization under constant KV-cache memory and linear time.

3 Methodology

Current VideoLLMs are constrained by fixed context windows, making it essential to select a compact set of query-relevant keyframes from long videos. This requires both temporally-contextualized frame representations that capture the full video context, and a learned alignment between frame and query semantics. FrameScout achieves this with two core components (Figure 1): a **streaming selector architecture** (Section 3.2) that builds temporally-contextualized frame representations, and a **frame-query contrastive objective** (Section 3.3) that learns accurate frame-query alignment for keyframe selection. We begin by formalizing the problem in Section 3.1.

3.1 Problem Formulation

Given a long video $\mathbf{F} = (F_1, \ldots, F_N)$ of N frames and a text query Q, our goal is to learn a lightweight selector $\mathcal{S}_{\boldsymbol{\theta}}(\mathbf{F}, Q) \to \hat{\mathbf{F}}$ that chooses a K-frame subsequence $\hat{\mathbf{F}} = (F_{i_1}, \ldots, F_{i_K})$ from $\mathbf{F}$ maximizing the answer quality of a downstream base VideoLLM $\mathcal{M}$:

$$\hat{\mathbf{F}} = \arg\max_{|\hat{\mathbf{F}}|=K} \; \mathcal{P}\big(\mathcal{M}(\hat{\mathbf{F}}, Q), A\big), \tag{1}$$

where A is the ground-truth answer and $\mathcal{P}$ is the answer quality metric.

The selector must be much smaller than the base VideoLLM $\mathcal{M}$; otherwise, the cost of selection would exceed that of processing the full video directly with $\mathcal{M}$. Designing such a selector poses two challenges: **(i)** a smaller model cannot process all frames at once, so how to build **temporally-contextualized frame representations** that capture full video context (Section 3.2), and **(ii)** how to train the selector so that **query-relevant frames** are scored above irrelevant ones (Section 3.3).

3.2 Streaming Selector Architecture

Keyframe selection requires temporally-contextualized frame representations that capture the full video context. Although an autoregressive VideoLLM can contextualize every frame with all preceding frames via causal attention, processing all frames at once far exceeds its context window for a long video. Existing methods either encode each frame independently with vision-language

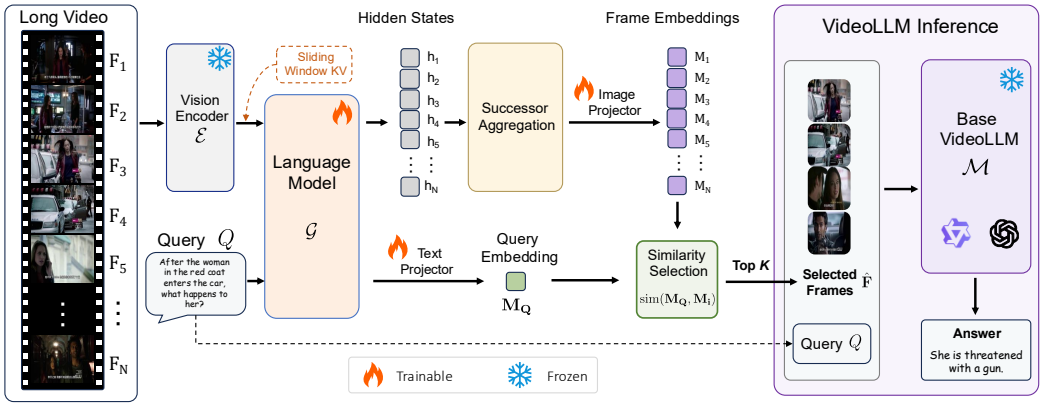

Figure 1: Illustration of FrameScout.

encoders [75, 54], or partition the video into chunks and encode frames jointly within each chunk [69, 72]; in both cases, frames or chunks are processed in isolation, so the resulting representations lack cross-boundary temporal context and future context.

We address this limitation by designing the selector $\mathcal{S}_{\boldsymbol{\theta}}$ as a streaming architecture, using a sliding-window KV cache to incorporate past context and successor aggregation to incorporate future context. Specifically, $\mathcal{S}_{\boldsymbol{\theta}}$ is a lightweight VideoLLM comprising a vision encoder $\mathcal{E}$, a language model $\mathcal{G}$, and a two-layer MLP projector ϕ. Given the input video $\mathbf{F} = (F_1, \ldots, F_N)$, $\mathcal{E}$ first encodes each frame into visual tokens: $\mathbf{v}_i = \mathcal{E}(F_i) \in \mathbb{R}^{L \times d}$, where L is the number of tokens per frame and d is the hidden dimension. Like existing methods, the selector partitions the video into non-overlapping chunks of B frames to fit within the context window. Let $\mathcal{B}_t = (F_{(t-1)B+1}, \ldots, F_{tB})$ denote the t-th chunk ($t = 1, \ldots, \lceil N/B \rceil$) and $\mathbf{V}_t = (\mathbf{v}_{(t-1)B+1}, \ldots, \mathbf{v}_{tB})$ the corresponding visual tokens. Unlike existing methods, however, our selector can maintain cross-boundary temporal context by using a sliding-window KV cache and successor aggregation.

History context via sliding-window KV cache. As illustrated in Figure 2, the sliding-window cache consists of two parts: (i) a fixed system prompt $\mathbf{s}$, whose KV cache $\mathrm{KV}(\mathbf{s})$ is computed once and always retained as *attention sinks* [63] to stabilize the attention distribution; and (ii) a sliding window of size W storing the KV cache of the W most recently processed chunks, denoted $\mathrm{KV}_{[t-W,\,t-1]}$. The full cache when processing chunk t is $\mathbf{C}_t = (\mathrm{KV}(\mathbf{s}),\ \mathrm{KV}_{[t-W,\,t-1]})$. The backbone $\mathcal{G}$ takes $\mathbf{V}_t$ and $\mathbf{C}_t$, producing a hidden state $\mathbf{h}_i \in \mathbb{R}^d$ for each frame F_i in the chunk:

$$(\mathbf{h}_{(t-1)B+1}, \ldots, \mathbf{h}_{tB}) = \mathcal{G}\big(\mathbf{C}_t,\ \mathbf{V}_t\big). \tag{2}$$

After processing chunk $\mathcal{B}_t$, the window slides forward: $\mathbf{V}_t$'s KV entries (i.e., KV_t) are appended and the oldest chunk's entries KV_{t-W} are evicted, while $\mathrm{KV}(\mathbf{s})$ is always kept. The remaining entries are re-indexed via rotary position embedding (RoPE) to preserve correct positional information. The total cache size stays constant at $|\mathbf{s}| + WBL$ tokens regardless of video length N.

Through this cache chain, each $\mathbf{h}_i$ is contextualized by more than the WB frames in the current window: each earlier chunk's hidden states were computed while attending to their own preceding context, and this information propagates forward through successive KV states. By induction, $\mathbf{h}_i$ is informed by all preceding frames $F_1, \ldots, F_i$.

Future context via successor aggregation. The KV-cache chain contains rich past context, but $\mathbf{h}_i$ cannot see future context. To incorporate future information, we propose successor aggregation to encode future context into frame representations with exponential temporal discounting, so that nearby frames contribute more than distant ones:

$$\mathbf{M}_i = \phi\left(\sum_{k=0}^{N-i} \beta^k\, \mathbf{h}_{i+k}\right) \in \mathbb{R}^d, \tag{3}$$

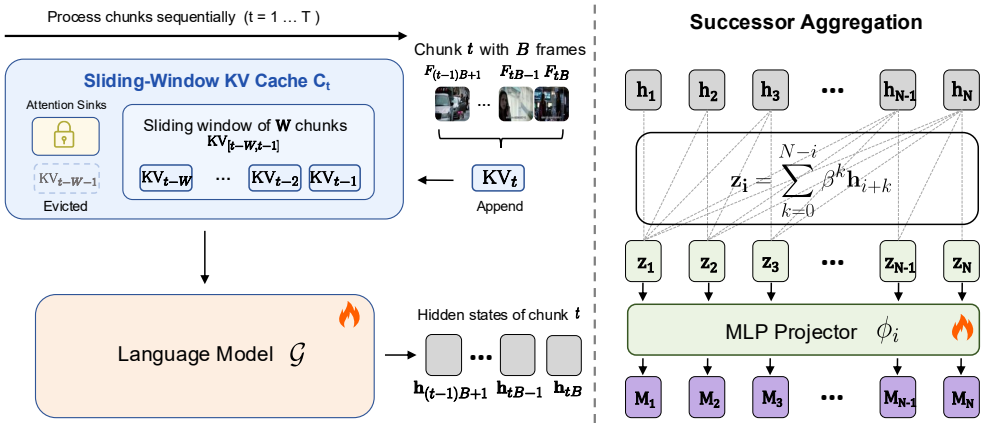

Figure 2: Illustration of the sliding-window KV Cache and successor aggregation mechanisms.

where $\beta \in [0, 1]$ is a discount factor. Letting $\mathbf{z}_i = \sum_{k=0}^{N-i} \beta^k \mathbf{h}_{i+k}$, we have the recurrence $\mathbf{z}_i = \mathbf{h}_i + \beta \mathbf{z}_{i+1}$ with $\mathbf{z}_N = \mathbf{h}_N$, so the entire sequence $\mathbf{z}_N, \ldots, \mathbf{z}_1$ can be computed by a single backward pass. This recurrence shares a similar form with the successor representation [14, 29] in reinforcement learning, but is applied here to aggregate future context rather than to decompose a value function. The resulting frame embedding $\mathbf{M}_i = \phi(\mathbf{z}_i)$ captures the full temporal context: past frames through the KV-cache chain, and future frames through the successor aggregation. After processing all N frames, we obtain their embeddings $(\mathbf{M}_1, \ldots, \mathbf{M}_N)$.

Complexity analysis. Let $L_s = |\mathbf{s}|$ denote the system prompt length and $T = \lceil N/B \rceil$ the total number of chunks. In a standard full-context VideoLLM, processing all N frames simultaneously yields a sequence of NL tokens, incurring $\mathcal{O}(N^2L^2d)$ computation and $\mathcal{O}(NLd)$ memory for the KV cache. In contrast, FrameScout processes N frames via streaming, with the attention context at each step bounded by $L_s + WBL$ tokens:

$$\text{Computation} = \mathcal{O}\big(T(L_s + WBL)^2 d\big), \quad \text{Memory} = \mathcal{O}\big((L_s + WBL)\, d\big). \tag{4}$$

Since W and B are fixed constants with $WB \ll N$, the computational cost is *linear* in video length and the KV-cache memory is *constant*. The successor aggregation recurrence adds only $O(Nd)$ computation and $O(d)$ memory, which is dominated by the attention cost above. Together, these properties enable FrameScout to process videos of arbitrary duration.

3.3 Frame-Query Contrastive Learning

The streaming architecture produces query-agnostic frame embeddings $(\mathbf{M}_1, \ldots, \mathbf{M}_N)$. To enable query-relevant frame selection, we propose a frame-query contrastive objective that aligns frame and query embeddings in a shared space, so that their similarity reflects frame-query relevance.

3.3.1 Training

Existing video QA datasets only provide video-level question-answer pairs, not frame-level relevance annotations. Following [72], we generate pseudo-labels by feeding a sampled subset of N' frames (N'=100) into a teacher VideoLLM (e.g., the base VideoLLM) and extracting the attention weights from query tokens to visual tokens at a reference layer. These weights are averaged across heads and aggregated per frame to obtain a query-frame relevance score. Note that N' is chosen to fit within the teacher's context window, enabling a single forward pass. Based on these scores, we select the top-k frames as the positive set $\mathcal{T}_+$ and treat all remaining frames as the negative set $\mathcal{T}_-$.

The query Q is encoded through the same backbone $\mathcal{G}$ and projector ϕ used for frame embeddings, yielding a query embedding $\mathbf{M}_Q = \phi(\mathcal{G}(Q)) \in \mathbb{R}^d$. Let $\text{sim}(\mathbf{M}_Q, \mathbf{M}_i)$ denote the cosine similarity between $\mathbf{M}_Q$ and frame embedding $\mathbf{M}_i$. We train the selector with a frame-query contrastive loss:

$$\mathcal{L} = -\frac{1}{|\mathcal{T}_+|} \sum_{i \in \mathcal{T}_+} \log \frac{\exp(\text{sim}(\mathbf{M}_Q, \mathbf{M}_i)/\tau)}{\exp(\text{sim}(\mathbf{M}_Q, \mathbf{M}_i)/\tau) + \sum_{j \in \mathcal{T}_-} \exp(\text{sim}(\mathbf{M}_Q, \mathbf{M}_j)/\tau)}, \tag{5}$$

where τ is a temperature hyperparameter. We choose the contrastive loss over the Spearman correlation loss used in FlexSelect [72] because keyframe selection only requires relevant frames to rank *above* irrelevant ones, not the relative ordering among frames. The contrastive loss directly optimizes this relevant-vs-irrelevant separation, matching the criterion used at inference.

Algorithm 1 in Appendix A summarizes the training procedure. During training, we process each chunk independently with only the sink KV cache $\mathrm{KV}(\mathbf{s})$, because this enables chunk-level parallelism and bounds per-sample activation memory to a single chunk, making training efficient and scalable. Cross-chunk temporal context is still captured by the successor aggregation, which operates identically to inference and keeps the learned representations consistent. The streaming sliding-window KV cache is activated at inference to further enrich frame representations with history context. Section 4.3.4 verifies that, under the same wall-clock budget, this strategy converges faster and is more accurate than training with the streaming KV context.

3.3.2 Inference

Since frame embeddings are query-agnostic, they are computed once per video and reused across all queries. As shown in Algorithm 2 of Appendix A, the inference pipeline is separated into two steps:

Step 1: Video encoding (once per video). The selector streams through the entire video as described in Section 3.2, producing frame embeddings $(\mathbf{M}_1, \ldots, \mathbf{M}_N)$.

Step 2: Frame selection (per query). For a query Q, we compute its embedding $\mathbf{M}_Q = \phi(\mathcal{G}(Q))$ and select the top-K frames:

$$\hat{\mathcal{I}} = \text{Top-K}\left(\{\text{sim}(\mathbf{M}_Q, \mathbf{M}_i)\}_{i=1}^{N}\right), \quad \hat{\mathbf{F}} = (F_i)_{i \in \hat{\mathcal{I}}}, \tag{6}$$

where Top-K returns the indices of the K highest-scoring frames. The selected keyframes $\hat{\mathbf{F}}$ are then fed into the base VideoLLM $\mathcal{M}$ with Q to produce the answer $A = \mathcal{M}(\hat{\mathbf{F}}, Q)$. Step 2 involves only one text forward pass and N dot products, so the per-query cost is negligible ($\sim$0.03 s; Section 4.2). In contrast, existing methods [75, 69, 66] must re-encode the full video for every new query. The selector is also fully decoupled from $\mathcal{M}$, so $\mathcal{M}$ can be replaced by any VideoLLM (e.g., open-source Qwen3-VL [1] or closed-source GPT-5.4 [43]).

4 Experiments

4.1 Experimental Settings

Benchmarks. Following [72], we evaluate on three long-video understanding benchmarks: (i) **VideoMME** [18] provides comprehensive evaluation across short, medium, and long videos; we report the Long and Overall accuracy without subtitles. (ii) **MLVU** [74] is a multi-task benchmark for long-form video understanding; we report the M-Avg accuracy. (iii) **LongVB** [62] targets retrieval and reasoning in long-context videos; we report the validation set accuracy.

Base VideoLLMs. The evaluation is conducted on a diverse set of closed- and open-source base VideoLLMs $\mathcal{M}$. Closed-source models include GPT-5.4-mini and GPT-5.4 [43]. Open-source models include Qwen2.5-VL-7B [2], LLaVA-Video-7B [73], InternVL2.5-8B [12], and Qwen3-VL-8B [1]. For the lightweight selector $\mathcal{S}_{\boldsymbol{\theta}}$, we adopt InternVL2.5-1B [12] across all base VideoLLMs.

Implementation Details. We use InternVL2.5-8B as the teacher model to generate attention-based pseudo-labels on the LLaVA-Video dataset [73] ($\sim$80k video–text pairs). The temperature τ is set to 0.05. The chunk size is B=12 frames for an accuracy-efficiency trade-off (as shown in Appendix B.1), the system-prompt length is $|\mathbf{s}|$=256 tokens, and the discount factor is β=0.3 (see Appendix B.1 for a sensitivity analysis).

Baselines. We compare FrameScout against three frame-selection methods: (i) **Uniform**: uniformly samples a subset of frames from the video. (ii) **FOCUS** [75]: a training-free method that encodes each frame independently and scores frame–query relevance in CLIP space, without any temporal modeling. (iii) **FlexSelect** [72]: a learned selection method that partitions the video into independent chunks and scores frames within each chunk in isolation, preventing information flow across chunk boundaries. For all settings and baselines, the base VideoLLM receives exactly $K = 60$ selected frames so that comparisons isolate frame-selection quality from input budget differences. Comparisons with the

Table 1: Accuracy (%) on long-video understanding benchmarks. Relative improvements over uniform sampling are shown in parentheses.

	VideoLLM	Method	VideoMME		MLVU	LongVB
			Long	Overall	M-Avg	Val
Closed-Source	GPT-5.4-mini [43]	Uniform	46.9	56.1	57.8	52.9
		FOCUS [75]	42.1 (-4.8)	53.2 (-2.9)	62.3 (+4.5)	54.2 (+1.3)
		FlexSelect [72]	50.4 (+3.5)	59.3 (+3.2)	61.2 (+3.4)	58.8 (+5.9)
		FrameScout	**52.3** (+5.4)	**60.8** (+4.7)	**62.7** (+4.9)	**58.9** (+6.0)
	GPT-5.4 [43]	Uniform	60.9	66.9	68.5	63.6
		FOCUS [75]	54.2 (-6.7)	64.9 (-2.0)	67.3 (-1.2)	63.2 (-0.4)
		FlexSelect [72]	64.7 (+3.8)	70.6 (+3.7)	64.7 (-3.8)	68.1 (+4.5)
		FrameScout	**65.4** (+4.5)	**72.5** (+5.6)	**72.0** (+3.5)	**68.4** (+4.8)
Open-Source	Qwen2.5-VL-7B [2]	Uniform	51.4	62.6	64.9	59.0
		FOCUS [75]	54.3 (+2.9)	63.4 (+0.8)	69.3 (+4.4)	61.1 (+2.1)
		FlexSelect [72]	55.4 (+4.0)	64.9 (+2.3)	67.2 (+2.3)	63.0 (+4.0)
		FrameScout	**57.3** (+5.9)	**65.6** (+3.0)	**70.3** (+5.4)	**63.3** (+4.3)
	LLaVA-Video-7B [73]	Uniform	53.5	63.8	71.5	60.4
		FOCUS [75]	46.9 (-6.6)	59.2 (-4.6)	69.3 (-2.2)	63.2 (+2.8)
		FlexSelect [72]	51.8 (-1.7)	62.6 (-1.2)	70.2 (-1.3)	62.5 (+2.1)
		FrameScout	**56.1** (+2.6)	**65.7** (+1.9)	**72.3** (+0.8)	**63.4** (+3.0)
	InternVL2.5-8B [12]	Uniform	51.4	62.7	68.9	61.8
		FOCUS [75]	54.2 (+2.8)	61.9 (-0.8)	69.5 (+0.6)	63.9 (+2.1)
		FlexSelect [72]	55.0 (+3.6)	63.3 (+0.6)	69.6 (+0.7)	65.0 (+3.2)
		FrameScout	**55.4** (+4.0)	**64.3** (+1.6)	**70.2** (+1.3)	**65.7** (+3.9)
	Qwen3-VL-8B [1]	Uniform	54.7	65.1	67.9	62.9
		FOCUS [75]	57.5 (+2.8)	64.8 (-0.3)	70.7 (+2.8)	64.4 (+1.5)
		FlexSelect [72]	60.3 (+5.6)	67.7 (+2.6)	71.6 (+3.7)	64.1 (+1.2)
		FrameScout	**61.2** (+6.5)	**69.3** (+4.2)	**72.6** (+4.7)	**67.8** (+4.9)

recent methods BOLT [38], F2C [53], M-LLM [22], HICom [40], and VTM [30] are provided in Appendix B.2.

4.2 Main Results

As shown in Table 1, FrameScout consistently outperforms all baselines across three benchmarks and six base VideoLLMs spanning both closed- and open-source families.

Consistent improvements over uniform sampling. FrameScout yields consistent gains over the uniform baseline on every model–benchmark combination. The improvements are large on Qwen3-VL-8B ($+6.5$ on VideoMME Long, $+4.7$ on MLVU, $+4.9$ on LongVideoBench) and Qwen2.5-VL-7B ($+5.9$ on VideoMME Long, $+5.4$ on MLVU). Crucially, FrameScout also benefits the strongest base model: when paired with GPT-5.4, it lifts VideoMME Overall by $+5.6$ and MLVU by $+3.5$, confirming that our selector provides orthogonal improvements to frontier VideoLLMs.

Surpassing state-of-the-art. FrameScout consistently outperforms both FOCUS and FlexSelect across all base models and benchmarks. On Qwen3-VL-8B, for example, FrameScout surpasses FlexSelect by $+1.6$ on VideoMME Overall, $+1.0$ on MLVU, and $+3.7$ on LongVideoBench; the margin over FOCUS is even larger. Moreover, both baselines exhibit inconsistent behavior: FOCUS frequently degrades performance (e.g., -6.7 on VideoMME Long for GPT-5.4; -6.6 for LLaVA-Video-7B), and FlexSelect still harms several configurations (e.g., -3.8 on MLVU for GPT-5.4; -1.7 on VideoMME Long for LLaVA-Video-7B). In contrast, FrameScout delivers positive gains in all settings. FrameScout also outperforms BOLT, F2C, M-LLM, HICom and VTM (Appendix B.2). We

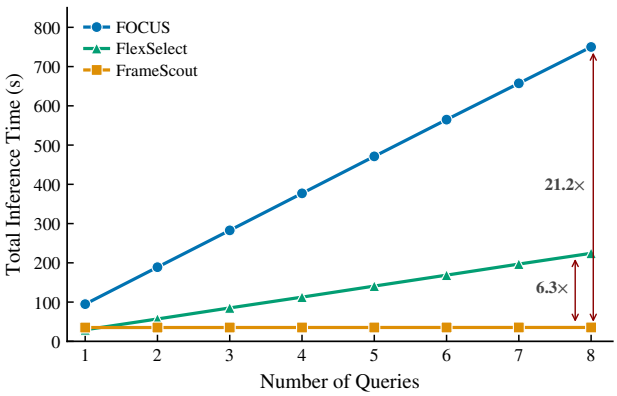

Figure 3: Total inference time vs. #Queries.

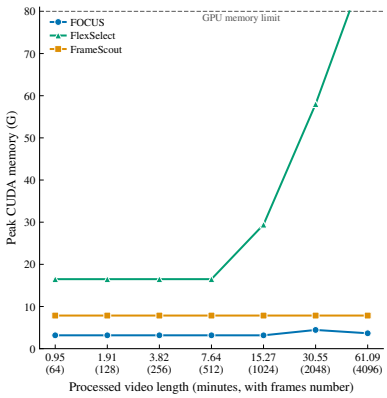

Figure 4: Memory vs. video length.

attribute this robustness to two design choices: (i) our streaming architecture contextualizes every frame with the full video via a sliding-window KV cache and successor aggregation, whereas existing selectors process frames or chunks in isolation and thus lack cross-boundary temporal context. (ii) our frame-query contrastive objective directly optimizes the separation between relevant and irrelevant frames in a shared embedding space, enabling more accurate selection than the CLIP-space heuristics of FOCUS, BOLT, and F2C, the rank-supervised per-chunk scoring of FlexSelect, or the pointwise score regression of M-LLM.

Inference Efficiency and Scalability. In practice, users often pose multiple queries about the same video (e.g., first requesting a summary, then drilling into specific scenes), making multi-turn QA a common and important use case. We therefore benchmark the end-to-end inference latency of FrameScout against FOCUS [75] and FlexSelect [72] as the number of queries on the same video grows from 1 to 8 (Figure 3). The efficiency gap widens sharply as queries accumulate. Both FOCUS and FlexSelect must re-encode the video for every new query, so their total latency grows linearly with the number of queries. In contrast, FrameScout computes frame embeddings once and reuses them across all queries, keeping total latency nearly constant. At eight queries, FrameScout achieves a $21.2\times$ speedup over FOCUS and a $6.3\times$ speedup over FlexSelect.

We further measure peak GPU memory of frame selection process as video length increases from 64 to 4096 frames (Figure 4). FlexSelect's memory grows and exceeds 80 GB at 4096 frames, making it impractical for hour-scale videos on a single GPU. In contrast, FrameScout maintains constant memory ($\sim$8 GB) regardless of video length, consistent with the $\mathcal{O}((L_s + WBL)\, d)$ bound derived in Section 3.2, and remains $10\times$ lower than FlexSelect at 4096 frames. Moreover, transferring the retained frame embeddings between GPU and disk for reuse takes only 35.75 ms for an 8192-frame video, merely 0.014% of its 252 s encoding time, which is negligible.

4.3 Analysis

4.3.1 Effectiveness of the Streaming Selector Architecture

We validate the streaming selector architecture (Section 3.2) by comparing against a *Separate Windows* baseline that partitions the video into non-overlapping chunks of B frames and processes each chunk independently, with no temporal context propagated across chunk boundaries. All other components remain identical.

As shown in Table 2, the streaming architecture consistently improves accuracy across both VideoLLMs and benchmarks. This confirms that the streaming architecture produces frame representations with richer cross-boundary temporal context, which is critical for queries requiring reasoning over events that span multiple chunks. Finer-grained ablations on successor aggregation and the contrastive objective, together with analyses on frame budgets, pseudo-label reliability, and other design choices, are provided in Appendix B.3–B.10.

Table 2: Ablation study on the streaming selector architecture.

VideoLLM $\mathcal{M}$	Method	VideoMME	LongVB
Qwen2.5-VL-7B	Separate Windows	65.0	61.3
	Streaming	**65.6**	**63.3**
LLaVA-Video-7B	Separate Windows	63.7	60.6
	Streaming	**65.7**	**63.4**

Table 3: Generalization across selector backbones on VideoMME using Qwen2.5-VL-7B.

Selector $\mathcal{S}_\theta$	Long	Overall
Uniform (no selector)	51.4	62.6
FrameScout (LLaVA-OV-0.5B)	55.8	64.6
FrameScout (InternVL2.5-1B)	**57.3**	**65.6**

4.3.2 Generalization Across Selector Backbones

The selector is designed to be architecture-agnostic: any lightweight VideoLLM can serve as its backbone. To verify this, we replace the default InternVL2.5-1B backbone with LLaVA-OneVision-0.5B [32], a model with a different architecture and less than half the parameters. As shown in Table 3, both backbones yield substantial improvements over uniform sampling, confirming that the gains stem from the streaming architecture and contrastive training framework (Sections 3.2–3.3) rather than from a specific backbone choice. The larger InternVL2.5-1B selector provides additional gains, suggesting that increased selector capacity further improves frame selection quality.

4.3.3 Representational Consistency

The streaming architecture processes chunks sequentially under a bounded KV cache, raising a natural question: do the resulting frame representations degrade compared to full-sequence inference that processes all frames at once? To answer this, we compare the two on 100 videos each containing $N = 1024$ frames, computing the mean frame-level cosine similarity between their outputs per video and visualizing the distribution in Figure 5.

As shown, the streaming architecture yields per-video similarities tightly concentrated near 1.0, closely recovering the full-sequence representations under constant KV-cache memory. In contrast, Separate Windows exhibits substantially lower similarity with a larger variance, confirming that processing chunks in isolation leads to significant representational degradation.

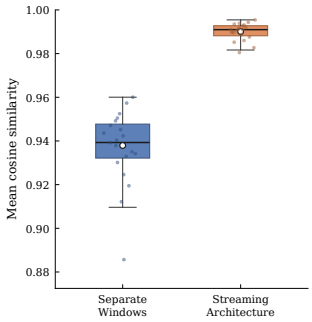

Figure 5: Distribution of similarities between two strategies.

4.3.4 Training Context Strategy

As described in Section 3.3.1, the selector is trained on independent chunks with only the sink KV cache, while the sliding-window KV cache is activated only at inference. To test whether this training-inference difference hurts performance, we train two selectors from the same initialization under the same wall-clock budget, changing only the training context: independent chunks (ours) versus the streaming KV context used at inference. As shown in Table 4, independent-chunk training converges faster (lower training loss) and achieves higher accuracy on both benchmarks, confirming that our training strategy is both more efficient and more accurate.

Table 4: Comparison of training context strategies under the same wall-clock budget using Qwen2.5-VL-7B.

Training context	Training loss	VideoMME	LongVB
Streaming KV context	0.3731	64.2	62.1
Independent chunks (ours)	**0.2057**	**65.6**	**63.3**

4.3.5 Visualization of the Shared Embedding Space

To verify that the frame-query contrastive objective (Section 3.3) learns a well-structured shared embedding space, we use t-SNE [56] to project the frame and query embeddings into a 2D plane. We select a sample from the NExT-GQA dataset [65] with ground-truth (GT) temporal annotations.

As can be seen from Figure 6, the query embedding (red star) is mapped closer to the GT keyframes than to the background frames, with an average cosine similarity of 0.33 for keyframes versus 0.12 for background frames. Moreover, the top-3 retrieved frames fall precisely within the GT span, demonstrating that the contrastive objective successfully separates query-relevant frames from irrelevant ones in the shared space, enabling accurate retrieval via simple cosine similarity.

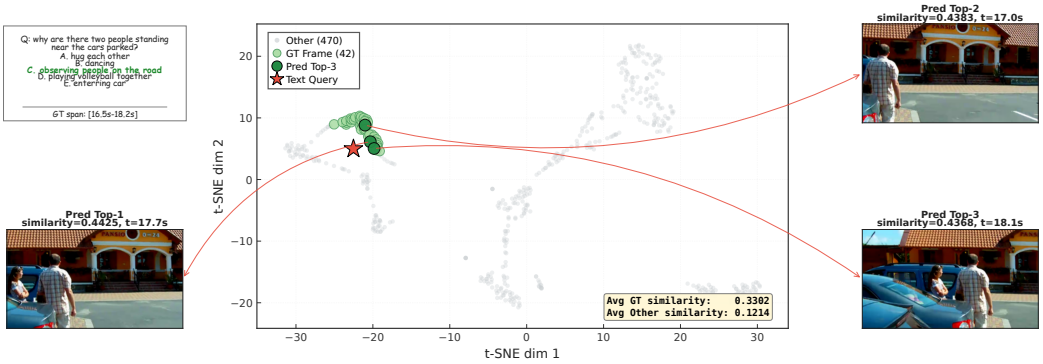

Figure 6: t-SNE visualization of query and frame embeddings.

4.3.6 Qualitative Results

Figure 7 shows a qualitative comparison with FlexSelect [72] on a temporal reasoning query. As can be seen, FlexSelect fails to propagate the anchoring event across chunk boundaries and selects temporally irrelevant frames, leading to the wrong answer. In contrast, FrameScout successfully selects the key post-event frames marked with red stars and generates the correct answer, verifying that the streaming architecture captures cross-chunk temporal dependencies that isolated per-chunk encoding misses. The higher cosine similarity of the selected keyframes further confirms that the contrastive objective learns a well-separated embedding space where query-relevant frames are clearly distinguished from irrelevant ones. Additional examples are provided in Appendix B.11.

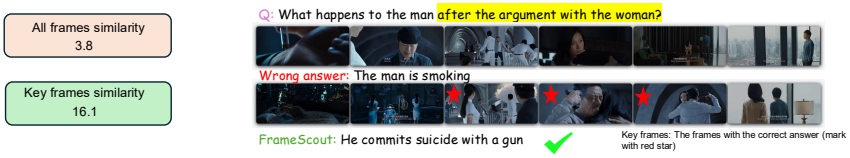

Figure 7: Qualitative comparison with FlexSelect on a temporal reasoning query.

5 Conclusion

In this paper, we proposed FrameScout, a keyframe selection framework for long video understanding with two core designs: a streaming selector architecture that produces temporally-contextualized frame embeddings under constant KV-cache memory and linear computation, and a frame-query contrastive objective that aligns frame and query embeddings in a shared space for accurate selection. Experiments on three long-video benchmarks and six base VideoLLMs demonstrate consistent state-of-the-art results, while achieving a $6\times$ speedup in multi-query scenarios.

Acknowledgement

This research was supported in part by National Key Research and Development Program of China under Grant no. 2022YFA1004102, National Natural Science Foundation of China under Grant no. 62136005, National Science Fund for Distinguished Young Scholars no. 62325204, Shenzhen fundamental research program JCYJ20250604144724032, and the Research Grants Council of the Hong Kong Special Administrative Region (Grants 16202523 and 16213126).

References

[1] S. Bai, Y. Cai, R. Chen, K. Chen, X. Chen, Z. Cheng, L. Deng, W. Ding, C. Gao, C. Ge, et al. Qwen3-VL technical report. Preprint arXiv:2511.21631, 2025.

[2] S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang, S. Wang, J. Tang, et al. Qwen2.5-VL technical report. Preprint arXiv:2502.13923, 2025.

[3] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. van Hasselt, and D. Silver. Successor features for transfer in reinforcement learning. In *Neural Information Processing Systems*, 2017.

[4] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer. Preprint arXiv:2004.05150, 2020.

[5] M. Blondel, O. Teboul, Q. Berthet, and J. Djolonga. Fast differentiable sorting and ranking. In *International Conference on Machine Learning*, 2020.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, et al. Language models are few-shot learners. In *Neural Information Processing Systems*, 2020.

[7] M. Cai, R. Tan, J. Zhang, B. Zou, K. Zhang, F. Yao, F. Zhu, J. Gu, Y. Zhong, Y. Shang, et al. Temporalbench: Benchmarking fine-grained temporal understanding for multimodal video models. Preprint arXiv:2410.10818, 2024.

[8] Z. Cai, Y. Zhang, B. Gao, Y. Liu, Y. Li, T. Liu, K. Lu, W. Xiong, Y. Dong, J. Hu, and W. Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *Conference on Language Modeling*, 2025.

[9] S. Chen, W. Jiang, B. Lin, J. T. Kwok, and Y. Zhang. RouterDC: Query-based router by dual contrastive learning for assembling large language models. In *Neural Information Processing Systems*, 2024.

[10] S. Chen, W. Jiang, Y. Gong, S. Luo, C. Zhuo, Z. Li, J. T. Kwok, and Y. Zhang. SPARD: Defending harmful fine-tuning attack via safety projection with relevance–diversity data selection. In *International Conference on Machine Learning*, 2026.

[11] Y. Chen, F. Xue, D. Li, Q. Hu, L. Zhu, X. Li, Y. Fang, H. Tang, S. Yang, Z. Liu, E. He, H. Yin, P. Molchanov, J. Kautz, J. Fan, Y. Zhu, Y. Lu, and S. Han. Longvila: Scaling long-context visual language models for long videos. In *International Conference on Learning Representations*, pages 18227–18246, 2025.

[12] Z. Chen, J. Wu, W. Wang, W. Su, G. Chen, S. Xing, M. Zhong, Q. Zhang, X. Zhu, L. Lu, et al. InternVL: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[13] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, et al. Training verifiers to solve math word problems. Preprint arXiv:2110.14168, 2021.

[14] P. Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 1993.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

[16] Y. Fan, X. Ma, R. Wu, Y. Du, J. Li, Z. Gao, and Q. Li. VideoAgent: A memory-augmented multimodal agent for video understanding. In *European Conference on Computer Vision*, 2024.

[17] W. Fedus, B. Zoph, and N. Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[18] C. Fu, Y. Dai, Y. Luo, L. Li, S. Ren, R. Zhang, Z. Wang, C. Zhou, Y. Shen, M. Zhang, et al. Video-MME: The first-ever comprehensive evaluation benchmark of multi-modal LLMs in video analysis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[19] C. Han, Q. Wang, H. Peng, W. Xiong, Y. Chen, H. Ji, and S. Wang. Lm-infinite: Zero-shot extreme length generalization for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024.

[20] B. He, H. Li, Y. K. Jang, M. Jia, X. Cao, A. Shah, A. Shrivastava, and S.-N. Lim. MA-LMM: Memory-augmented large multimodal model for long-term video understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[21] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[22] K. Hu, F. Gao, X. Nie, P. Zhou, S. Tran, T. Neiman, L. Wang, M. Shah, R. Hamid, B. Yin, and T. Chilimbi. M-LLM based video frame selection for efficient video understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13702–13712, 2025.

[23] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al. Mistral 7B. Preprint arXiv:2310.06825, 2023.

[24] W. Jiang and S. J. Pan. MetaDefense: Defending fine-tuning based jailbreak attack before and during generation. In *Neural Information Processing Systems*, 2025.

[25] W. Jiang, Y. Zhang, and J. T. Kwok. Effective structured prompting by meta-learning and representative verbalizer. In *International Conference on Machine Learning*, 2023.

[26] W. Jiang, H. Shi, L. Yu, Z. Liu, Y. Zhang, Z. Li, and J. T. Kwok. Forward-backward reasoning in large language models for mathematical verification. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024.

[27] W. Jiang, S. Chen, and S. J. Pan. MetaMoE: Diversity-aware proxy selection for privacy-preserving mixture-of-experts unification. In *International Conference on Machine Learning*, 2026.

[28] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. In *Neural Information Processing Systems*, 2022.

[29] T. D. Kulkarni, A. Saeedi, S. Gautam, and S. J. Gershman. Deep successor reinforcement learning. Preprint arXiv:1606.02396, 2016.

[30] S.-H. Lee, J. Wang, Z. Zhang, D. Fan, and X. Li. Video token merging for long video understanding. In *Advances in Neural Information Processing Systems*, volume 37, pages 13851–13871, 2024.

[31] B. Lester, R. Al-Rfou, and N. Constant. The power of scale for parameter-efficient prompt tuning. In *Conference on Empirical Methods in Natural Language Processing*, 2021.

[32] B. Li, Y. Zhang, D. Guo, R. Zhang, F. Li, H. Zhang, K. Zhang, P. Zhang, Y. Li, Z. Liu, and C. Li. LLaVA-OneVision: Easy visual task transfer. *Transactions on Machine Learning Research*, 2025.

[33] J. Li, D. Li, C. Xiong, and S. Hoi. BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*, 2022.

[34] K. Li, Y. Wang, Y. He, Y. Li, Y. Wang, Y. Liu, Z. Wang, J. Xu, G. Chen, P. Luo, et al. MVBench: A comprehensive multi-modal video understanding benchmark. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[35] Y. Li, C. Wang, and J. Jia. LLaMA-VID: An image is worth 2 tokens in large language models. In *European Conference on Computer Vision*, 2024.

[36] B. Lin, W. Jiang, Y. Xu, H. Chen, and Y.-C. Chen. PARM: Multi-objective test-time alignment via preference-aware autoregressive reward model. In *International Conference on Machine Learning*, 2025.

[37] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. In *Neural Information Processing Systems*, 2023.

[38] S. Liu, C. Zhao, T. Xu, and B. Ghanem. BOLT: Boost large vision-language model without training for long-form video understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3318–3327, 2025.

[39] Y. Liu, S. Li, Y. Liu, Y. Wang, S. Ren, L. Li, S. Chen, X. Sun, and L. Hou. TempCompass: Do video LLMs really understand videos? In *Findings of the Association for Computational Linguistics*, 2024.

[40] Z. Liu, C.-W. Xie, P. Li, L. Zhao, L. Tang, Y. Zheng, C. Liu, and H. Xie. Hybrid-level instruction injection for video token compression in multi-modal large language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8568–8578, 2025.

[41] K. Mangalam, R. Akshulakov, and J. Malik. EgoSchema: A diagnostic benchmark for very long-form video language understanding. In *Neural Information Processing Systems*, 2023.

[42] OpenAI. GPT-4 technical report. Preprint arXiv:2303.08774, 2023.

[43] OpenAI. OpenAI GPT-5 system card. Preprint arXiv:2601.03267, 2025.

[44] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jégou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

[45] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, et al. Training language models to follow instructions with human feedback. In *Neural Information Processing Systems*, 2022.

[46] X. Qi, Y. Zeng, T. Xie, P.-Y. Chen, R. Jia, P. Mittal, and P. Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *International Conference on Learning Representations*, 2024.

[47] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

[48] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. In *Neural Information Processing Systems*, 2023.

[49] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 1997.

[50] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

[51] Y. Shu, Z. Liu, P. Zhang, M. Qin, J. Zhou, Z. Liang, T. Huang, and B. Zhao. Video-XL: Extra-long vision language model for hour-scale video understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[52] E. Song, W. Chai, G. Wang, Y. Zhang, H. Zhou, F. Wu, H. Chi, X. Guo, T. Ye, Y. Zhang, Y. Lu, J.-N. Hwang, and G. Wang. MovieChat: From dense token to sparse memory for long video understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[53] G. Sun, A. Singhal, B. Uzkent, M. Shah, C. Chen, and G. Kessler. From frames to clips: Training-free adaptive key clip selection for long-form video understanding. *arXiv preprint arXiv:2510.02262*, 2025.

[54] X. Tang, J. Qiu, L. Xie, Y. Tian, J. Jiao, and Q. Ye. Adaptive keyframe sampling for long video understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[55] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, et al. LLaMA: Open and efficient foundation language models. Preprint arXiv:2302.13971, 2023.

[56] L. Van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 2008.

[57] P. Wang, S. Bai, S. Tan, S. Wang, Z. Fan, J. Bai, K. Chen, X. Liu, J. Wang, W. Ge, et al. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[58] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

[59] X. Wang, Y. Zhang, O. Zohar, and S. Yeung-Levy. VideoAgent: Long-form video understanding with large language model as agent. In *European Conference on Computer Vision*, 2024.

[60] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Neural Information Processing Systems*, 2022.

[61] Y. Wei, S. Fu, W. Jiang, Z. Zhang, Z. Zeng, Q. Wu, J. T. Kwok, and Y. Zhang. GITA: Graph to visual and textual integration for vision-language graph reasoning. In *Neural Information Processing Systems*, 2024.

[62] H. Wu, D. Li, B. Chen, and J. Li. LongVideoBench: A benchmark for long-context interleaved video-language understanding. In *Neural Information Processing Systems*, 2024.

[63] G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024.

[64] J. Xiao, X. Shang, A. Yao, and T.-S. Chua. NExT-QA: Next phase of question-answering to explaining temporal actions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[65] J. Xiao, A. Yao, Y. Li, and T.-S. Chua. Can i trust your answer? visually grounded video question answering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[66] Y. Yao, Y. Yun, J. Wang, H. Zhang, D. Zhao, K. Tian, Z. Wang, M. Qiu, and T. Wang. K-Frames: Scene-driven any-k keyframe selection for long video understanding. Preprint arXiv:2510.13891, 2025.

[67] L. Yu, W. Jiang, H. Shi, J. Yu, Z. Liu, Y. Zhang, J. T. Kwok, Z. Li, A. Weller, and W. Liu. MetaMath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations*, 2024.

[68] S. Yu, J. Cho, P. Yadav, and M. Bansal. Self-chained image-language model for video localization and question answering. In *Neural Information Processing Systems*, 2023.

[69] S. Yu, C. Jin, H. Wang, Z. Chen, S. Jin, Z. Zuo, X. Xu, Z. Sun, B. Zhang, J. Wu, H. Zhang, and Q. Sun. Frame-Voyager: Learning to query frames for video large language models. In *International Conference on Learning Representations*, 2025.

[70] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. Sigmoid loss for language image pre-training. In *IEEE/CVF International Conference on Computer Vision*, 2023.

[71] P. Zhang, K. Zhang, B. Li, G. Zeng, J. Yang, Y. Zhang, Z. Wang, H. Tan, C. Li, and Z. Liu. Long context transfer from language to vision. Preprint arXiv:2406.16852, 2024.

[72] Y. Zhang, Y. Lu, T. Wang, F. Rao, Y. Yang, and L. Zhu. FlexSelect: Flexible token selection for efficient long video understanding. In *Neural Information Processing Systems*, 2025.

[73] Y. Zhang, J. Wu, W. Li, B. Li, Z. Ma, Z. Liu, and C. Li. LLaVA-video: Video instruction tuning with synthetic data. *Transactions on Machine Learning Research*, 2025.

[74] J. Zhou, Y. Shu, B. Zhao, B. Wu, Z. Liang, S. Xiao, M. Qin, X. Yang, Y. Xiong, B. Zhang, et al. MLVU: Benchmarking multi-task long video understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[75] Z. Zhu, H. Xu, Y. Luo, Y. Liu, K. Sarkar, Z. Yang, and Y. You. FOCUS: Efficient keyframe selection for long video understanding. In *International Conference on Learning Representations*, 2026.

A Training and Inference Algorithms

We provide the complete training and inference procedures of FrameScout in Algorithm 1 and Algorithm 2, respectively.

Algorithm 1 FrameScout: Training via frame-query contrastive learning.

Require: Dataset $\mathcal{D} = \{(\mathbf{F}, Q, A)\}$; teacher VideoLLM $\mathcal{M}_{\text{teacher}}$; trainable $\boldsymbol{\theta} = \{\mathcal{G}, \phi\}$ ($\mathcal{E}$ frozen); training frame count N'; chunk size B; pseudo-label top-k; contrastive temperature τ; successor discount β; learning rate η; sink KV cache $\text{KV}(\mathbf{s})$.

repeat
  Sample a mini-batch $\mathcal{B}$ from $\mathcal{D}$;
  for each $(\mathbf{F}, Q, A) \in \mathcal{B}$ **do**
  Pseudo-label generation:
    $\mathbf{F}' = (F_1, \ldots, F_{N'}) \leftarrow$ uniformly subsample N' frames from $\mathbf{F}$;
    $(\mathcal{T}_+, \mathcal{T}_-) \leftarrow \textsc{TeacherPseudoLabel}(\mathcal{M}_{\text{teacher}}, \mathbf{F}', Q, k)$;
  Forward encoding:
    for $i = 1, \ldots, N'$ **do**
      $\mathbf{v}_i \leftarrow \mathcal{E}(F_i) \in \mathbb{R}^{L \times d}$;
    end for
    for $t = 1, \ldots, N'/B$ **do**
      $\mathbf{V}_t \leftarrow (\mathbf{v}_{(t-1)B+1}, \ldots, \mathbf{v}_{tB})$;
      $\left(\mathbf{h}_{(t-1)B+1}, \ldots, \mathbf{h}_{tB}\right) \leftarrow \mathcal{G}(\text{KV}(\mathbf{s}),\, \mathbf{V}_t)$;
    end for
  Successor aggregation:
    $\mathbf{z}_{N'} \leftarrow \mathbf{h}_{N'}$;
    for $i = N' - 1, \ldots, 1$ **do**
      $\mathbf{z}_i \leftarrow \mathbf{h}_i + \beta\, \mathbf{z}_{i+1}$;
    end for
    $\mathbf{M}_i \leftarrow \phi(\mathbf{z}_i)$ for all $i = 1, \ldots, N'$;
  Query encoding and contrastive loss:
    $\mathbf{M}_Q \leftarrow \phi(\mathcal{G}(Q))$;
    Compute $\mathcal{L}$ via Eq. (5);
  end for
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\, \nabla_{\boldsymbol{\theta}} \mathcal{L}$;
until convergence
return $\mathcal{S}_{\boldsymbol{\theta}} = \{\mathcal{E}, \mathcal{G}, \phi\}$.

B Additional Experiments

B.1 Sensitivity to Hyperparameters

We examine two key hyperparameters: the chunk size B that controls how many frames the selector processes per step, and the discount factor β in successor aggregation (Section 3.2).

Chunk size B. As shown in Table 5, varying B from 8 to 16 on Qwen3-VL-8B causes VideoMME overall to fluctuate by only 0.5, indicating that FrameScout is robust to this choice. We use B=12 as the default in all experiments.

Discount factor β. Table 6 shows the effect of β on LLaVA-Video-7B. Setting β=0 disables successor aggregation entirely, reducing frame representations to purely causal (past-only) context; performance drops by 0.4 on VideoMME Overall compared to β=0.3, confirming that incorporating future context is beneficial. However, a large β (e.g., 0.8) over-weights distant future frames and degrades accuracy. β=0.3 is used in all experiments as it balances the future and past context.

B.2 Comparison with More Baseline Methods

We further compare FrameScout with the recent training-free selectors BOLT [38] and F2C [53], the training-based selector M-LLM [22], and the token compression methods HICom [40] and VTM [30],

Algorithm 2 FrameScout: Inference via streaming encoding and per-query retrieval.

Require: Video $\mathbf{F} = (F_1, \ldots, F_N)$ (N a multiple of B); queries $\{Q_j\}_{j=1}^{m}$; trained selector $\mathcal{S}_{\boldsymbol{\theta}} = \{\mathcal{E}, \mathcal{G}, \phi\}$; base VideoLLM $\mathcal{M}$; chunk size B; sliding-window size W; successor discount β; frame budget K; sink KV cache $\mathrm{KV}(\mathbf{s})$.

Stage 1: Streaming video encoding (once per video):

for $i = 1, \ldots, N$ **do**
  $\mathbf{v}_i \leftarrow \mathcal{E}(F_i) \in \mathbb{R}^{L \times d}$;
end for
Initialize sliding buffer $\mathrm{KV}_{\text{win}} \leftarrow \emptyset$;
for $t = 1, \ldots, N/B$ **do**
  $\mathbf{V}_t \leftarrow (\mathbf{v}_{(t-1)B+1}, \ldots, \mathbf{v}_{tB})$;
  $\big(\mathbf{h}_{(t-1)B+1}, \ldots, \mathbf{h}_{tB}\big) \leftarrow \mathcal{G}(\mathrm{KV}(\mathbf{s}),\, \mathrm{KV}_{\text{win}},\, \mathbf{V}_t)$;
  Append $\mathrm{KV}(\mathbf{V}_t)$ to KV_{win}; evict the oldest chunk if $|\mathrm{KV}_{\text{win}}| > W$;
end for

Successor aggregation:

$\mathbf{z}_N \leftarrow \mathbf{h}_N$;
for $i = N-1, \ldots, 1$ **do**
  $\mathbf{z}_i \leftarrow \mathbf{h}_i + \beta\, \mathbf{z}_{i+1}$;
end for
$\mathbf{M}_i \leftarrow \phi(\mathbf{z}_i)$ for all $i = 1, \ldots, N$;

Stage 2: Per-query retrieval and answering:

for $j = 1, \ldots, m$ **do**
  $\mathbf{M}_{Q_j} \leftarrow \phi(\mathcal{G}(Q_j))$;
  $\hat{\mathcal{I}}_j \leftarrow \text{Top-K}\big(\{\mathrm{sim}(\mathbf{M}_{Q_j}, \mathbf{M}_i)\}_{i=1}^{N}\big)$;
  $\hat{\mathbf{F}}_j \leftarrow (F_i)_{i \in \hat{\mathcal{I}}_j}$;
  $A_j \leftarrow \mathcal{M}(\hat{\mathbf{F}}_j, Q_j)$;
end for
return $\{A_j\}_{j=1}^{m}$.

Table 5: Sensitivity to chunk size B on VideoMME using Qwen3-VL-8B.

Chunk size B	**Long**	**Overall**
8	60.3	68.8
12	**61.2**	**69.3**
16	61.1	68.9

Table 6: Sensitivity to discount factor β on VideoMME using LLaVA-Video-7B.

β	**Long**	**Overall**
0.0	55.5	65.3
0.2	55.1	65.5
0.3	**56.1**	**65.7**
0.5	54.6	64.8
0.8	51.4	63.2

under the same protocol as Table 1; M-LLM is compared under its Qwen2-VL-7B [57] setting. As shown in Table 7, FrameScout outperforms the training-free selectors BOLT and F2C (e.g., $+2.1$ on VideoMME and $+1.7$ on LongVB over BOLT), the training-based selector M-LLM ($+4.1$ on VideoMME and $+3.6$ on LongVB), and the token compression methods HICom and VTM ($+7.8$ and $+4.5$ on VideoMME).

Table 7: Comparison with recent training-free selectors (F2C, BOLT), token compression methods (HICom, VTM), and the training-based selector M-LLM. VideoMME reports Overall accuracy. "–": not evaluated.

$\mathcal{M}$	Method	VideoMME	LongVB
Qwen2.5-VL	F2C [53]	62.3	60.6
	BOLT [38]	63.5	61.6
	HICom [40]	57.8	–
	VTM [30]	61.1	–
	FrameScout	**65.6**	**63.3**
Qwen2	M-LLM [22]	58.7	57.0
	FrameScout	**62.8**	**60.6**

B.3 Contribution of Each Component

FrameScout consists of three components: the sliding-window KV cache and successor aggregation in the streaming selector architecture (Section 3.2), and the frame-query contrastive objective (Section 3.3). To disentangle their contributions, we compare FrameScout with three variants, each changing exactly one component: removing the sliding-window KV cache (i.e., the Separate Windows baseline in Section 4.3.1), removing successor aggregation, or replacing the contrastive objective with the Spearman rank-correlation loss used in FlexSelect [72]. As shown in Table 8, the contrastive objective is the dominant contributor: replacing it with the Spearman loss causes the largest drop (-4.5 on VideoMME and -2.6 on LongVB). Removing either architectural component yields smaller but consistent degradation on both benchmarks.

Table 8: Effect of components in FrameScout using Qwen2.5-VL-7B.

Variant	VideoMME	LongVB
w/o sliding-window KV cache (Separate Windows)	65.0	61.3
w/o successor aggregation	65.3	62.7
w/o contrastive objective (Spearman)	61.1	60.7
Full FrameScout	**65.6**	**63.3**

B.4 Robustness to Frame Budget

Low budgets. We evaluate all selectors with $K \in \{4, 8, 16, 60\}$ (Table 9). FrameScout achieves the best accuracy in every setting, and its advantage is most pronounced under low budgets ($+5.9$ over Uniform at $K{=}4$ versus $+3.0$ at $K{=}60$ on VideoMME Overall). A small K demands high precision at the top of the ranking, which our contrastive objective directly optimizes by separating relevant from irrelevant frames, matching the top-K criterion at inference.

Table 9: Accuracy under different frame budgets K on VideoMME using Qwen2.5-VL-7B.

	Overall				Long			
Method	$K{=}4$	8	16	60	$K{=}4$	8	16	60
Uniform	49.2	53.9	57.3	62.6	44.4	47.8	48.3	51.4
FOCUS	54.4	57.0	57.7	63.4	47.4	47.3	49.0	54.3
FlexSelect	52.7	55.0	59.0	64.9	47.6	48.3	51.8	55.4
FrameScout	**55.1**	**57.4**	**60.3**	**65.6**	**48.1**	**51.4**	**52.7**	**57.3**

High budget (1 fps). We further evaluate a high-budget regime at 1 fps on VideoMME videos shorter than 10 minutes, so that all selected frames fit in the context capacity of the base VideoLLM. All methods rank the same candidate frames, with K set by the video duration to yield one selected frame per second. As shown in Table 10, FrameScout still achieves the best accuracy among all methods.

Table 10: Accuracy (%) under a 1 fps frame budget on the VideoMME subset with videos shorter than 10 minutes, using Qwen2.5-VL-7B.

	Uniform	**FOCUS**	**FlexSelect**	**FrameScout**
Accuracy	64.9	61.4	65.3	**66.2**

B.5 Reliability of Teacher-Attention Pseudo-Labels

To validate the supervision signal, we measure how often the teacher-attention pseudo-labels hit the human-annotated ground-truth (GT) temporal spans of NExT-GQA [65]. As shown in Table 11, more than 99% of GT spans are recalled by the top-40 pseudo-labeled frames, confirming that teacher-attention scores reliably correspond to answer-relevant frames identified by human annotators. We further show in Appendix B.8 that FrameScout is insensitive to the choice of teacher.

Table 11: Recall (%) of human-annotated GT temporal spans on NExT-GQA by the top-K teacher-attention pseudo-labeled frames.

Top-K	10	20	30	40
GT span recall	73.6	86.0	91.7	**99.2**

B.6 Design of the Projection Head

We compare three projection heads ϕ: *Identity* (directly using the backbone hidden states), *Linear* (a single trainable linear layer), and our default *two-layer MLP*. As shown in Table 12, the MLP performs best on all benchmarks, e.g., $+3.9$ on VideoMME Overall over Identity, showing that hidden states pretrained for next-token prediction are not directly suited for similarity-based retrieval. The MLP also consistently outperforms Linear, indicating that a nonlinear projection is beneficial for reshaping frame and query representations into a shared retrieval space.

Table 12: Ablation on the projection head ϕ using Qwen2.5-VL-7B.

Projector	**VideoMME Overall**	**VideoMME Long**	**LongVB**
Identity	61.7	56.4	60.3
Linear	63.0	56.0	59.9
Two-layer MLP (ours)	**65.6**	**57.3**	**63.3**

B.7 Choice of Future Context Aggregation

We compare successor aggregation with two simpler future-context alternatives: *backward temporal smoothing*, which averages the hidden states over a fixed-size future window, and *non-causal temporal aggregation*, which uniformly averages all future hidden states. Note that exponential future pooling is mathematically identical to successor aggregation (Eq. 3) and is thus not a separate baseline. As shown in Table 13, successor aggregation outperforms both alternatives on both benchmarks, confirming that the exponential decay structure, which weights nearby frames more than distant ones, is essential.

Table 13: Comparison of future context aggregation strategies using Qwen2.5-VL-7B.

Future aggregation	**VideoMME Overall**	**LongVB**
Backward temporal smoothing	65.0	61.3
Non-causal temporal aggregation	65.2	62.4
Successor aggregation (ours)	**65.6**	**63.3**

B.8 Sensitivity to the Teacher Model

We re-generate the pseudo-labels with a different teacher, Qwen2.5-VL-7B, and retrain the selector under the same setting. As shown in Table 14, the two teachers yield close downstream accuracy (average $\Delta = 0.3$), and both outperform FlexSelect, showing that FrameScout is insensitive to the choice of teacher.

Table 14: Sensitivity to the teacher model used for pseudo-label generation using Qwen2.5-VL-7B.

Method	VideoMME Long	VideoMME Overall	LongVB
FlexSelect [72]	55.4	64.9	63.0
FrameScout (teacher: InternVL2.5-8B)	**57.3**	**65.6**	**63.3**
FrameScout (teacher: Qwen2.5-VL-7B)	57.0	65.2	63.1

B.9 Adaptive Frame Budget

Instead of selecting a fixed number of frames, we select every frame whose cosine similarity to the query embedding exceeds a threshold $\delta = 0.7$, without any retraining. As shown in Table 15, the adaptive variant achieves higher accuracy on LongVB while using fewer frames on average, suggesting that the number of relevant frames varies across queries and that adapting K is a promising direction.

Table 15: Fixed versus threshold-based adaptive frame selection on LongVB using Qwen2.5-VL-7B.

Selection rule	Accuracy	Avg. #Frames
Fixed K	61.7	30
Adaptive (sim $> \delta$)	**62.0**	**26.5**

B.10 Robustness to Evaluation Prompts

For a fair comparison, all methods in Table 1 and Table 7 are evaluated with the same prompt template. To verify that the gains of FrameScout do not depend on this template, we further compare FrameScout with BOLT [38] under the prompt from BOLT's official repository, keeping all other components (frame sampling, frame selection, and answer extraction) unchanged. As shown in Table 16, FrameScout outperforms BOLT under both prompts and all frame budgets, e.g., by $+2.07$ under our prompt and $+2.11$ under BOLT's official prompt at K=60, showing that its advantage is robust to the evaluation prompt. Under BOLT's official prompt, our reproduced BOLT achieves 64.19 at K=32, consistent with its officially reported 64.0.

Table 16: BOLT and FrameScout under our evaluation prompt and BOLT's official prompt on VideoMME Overall using Qwen2.5-VL-7B.

Prompt	Method	K=8	K=16	K=32	K=60
Ours	BOLT [38]	55.63	59.44	62.07	63.52
	FrameScout	**57.44**	**60.30**	**63.11**	**65.59**
BOLT official	BOLT [38]	57.44	61.07	64.19	64.67
	FrameScout	**59.07**	**61.26**	**64.67**	**66.78**

B.11 Additional Examples

We provide additional qualitative comparisons with FOCUS [75] (Figure 8) and FlexSelect [72] (Figure 9). In both cases, the baselines select temporally irrelevant frames due to the lack of cross-chunk temporal context, leading to incorrect answers. In contrast, FrameScout consistently selects the query-relevant frames, confirming that the streaming architecture captures temporal dependencies across chunk boundaries and the contrastive objective produces a well-separated embedding space for accurate frame selection.

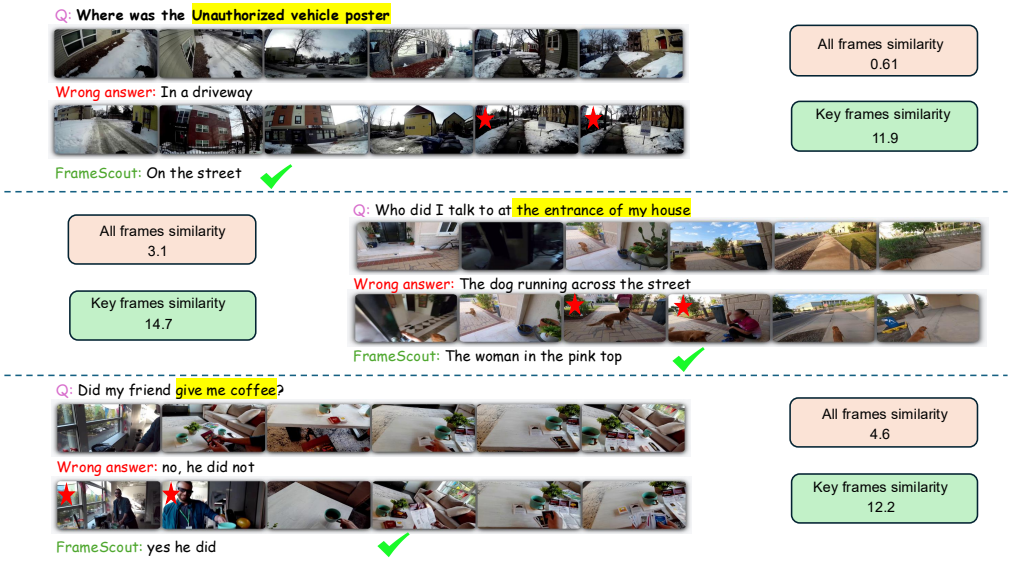

Figure 8: Comparison with the frame selection results of the FOCUS [75].



Figure 9: Comparison with the frame selection results of the FlexSelect [72].

C Additional Related Works

Video Large Language Models. Early VideoLLMs [73, 12, 1, 2] extend image-based multimodal LLMs by densely encoding a small number of sampled frames, yielding strong performance on short video benchmarks but scaling poorly with video duration. To enable long-video understanding, three main lines of work have emerged. *Context-window extension* [11, 71] scales the underlying LLM to hundreds of thousands of tokens, allowing denser frame sampling at the cost of expensive long-context training and inference. *Visual token compression* [35, 51] reduces the per-frame token count via learned aggregation or summarization tokens, but risks discarding fine-grained visual details crucial for question answering. *Memory-augmented architectures* [20, 52] maintain an online memory bank that stores compressed visual features, decoupling the model's working memory from total video length. All these methods modify, retrain, or wrap the base VideoLLM. In contrast, FrameScout operates as an external, lightweight keyframe selector that is fully decoupled from the base VideoLLM and can be combined with any open- or closed-source VideoLLM without modification.

Large Language Models. Large language models (LLMs) [6, 42, 55, 23] have demonstrated remarkable capabilities across a wide range of tasks. FrameScout builds on them in two roles: a lightweight LLM serves as the backbone of the selector, and a downstream (Video)LLM produces the final answer. Much work improves LLMs themselves, e.g., by enhancing reasoning via chain-of-thought prompting [60, 28], self-consistency [58], verifiers [13, 26], and data bootstrapping [67], or by extending them to visual inputs [37, 61]. Two lines of work are most related to ours. *(i) Lightweight modules serving a frozen LLM.* Parameter-efficient methods adapt a frozen model with soft prompts [31, 25] or low-rank adapters [21], and PARM [36] aligns a frozen LLM at test time with a small preference-aware reward model, instead of fine-tuning it via RLHF [45] or DPO [48]. Similarly, FrameScout leaves the base VideoLLM untouched, which also avoids the safety degradation induced by fine-tuning [46] that otherwise calls for dedicated defenses [24, 10]. *(ii) Relevance-based selection.* Mixture-of-experts models route each token to a few experts [50, 17], MetaMoE [27] selects proxies by relevance and diversity, and RouterDC [9] learns query and LLM embeddings via dual contrastive learning to route each query to the most suitable LLM. FrameScout adopts an analogous query–candidate contrastive alignment, but its candidates are video frames whose embeddings are computed from content with temporal context, rather than a fixed set of learnable model embeddings. While these works improve the LLM itself, FrameScout improves what it sees, selecting compact query-relevant evidence from long videos for any (Video)LLM.

D Limitations

FrameScout is trained with teacher-attention pseudo-labels, which are only a proxy for true frame-level relevance; higher-quality supervision (e.g., human-annotated evidence frames) could further improve performance. An adaptive frame budget based on query-specific evidence could further improve efficiency and accuracy, as suggested by our preliminary threshold-based variant (Appendix B.9). Finally, evicted chunks are retained only implicitly through hidden states propagated along the KV-cache chain; consolidating them into explicit memory tokens kept after the attention sinks is a natural constant-memory extension that we leave to future work.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 clearly state our core contributions.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in a dedicated Limitations section (Appendix D), including the reliance on teacher-attention pseudo-labels, the fixed frame budget, and the implicit retention of evicted chunks.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not contain formal theoretical results. The complexity analysis in Section 3.2 follows directly from the architecture design and does not require formal proof.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 4.1 specifies the benchmarks, base VideoLLMs, selector backbone, training data, teacher model, and key hyperparameters. Full training and inference procedures are detailed in Algorithms 1 and 2 in the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All benchmarks (VideoMME, MLVU, LongVideoBench) and training data (LLaVA-Video) are publicly available. The code, training scripts, model checkpoints, and reproduction instructions are available at https://github.com/Hubert-1102/FrameScout.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4.1 describes the training setup, including the teacher model (InternVL2.5-8B), training dataset (LLaVA-Video, 80k pairs), chunk size, system-prompt length, discount factor, and contrastive temperature. Each base VideoLLM receives exactly 60 selected frames for fair comparison.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We report single-run results and do not include error bars. This is because evaluation involves closed-source VideoLLMs (e.g., GPT-5.4, GPT-5.4-mini) where repeated API calls incur substantial cost. The consistent improvements across three benchmarks and six base VideoLLMs (Table 1) nonetheless provide strong evidence for the robustness of FrameScout.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Our experiments do not depend on a specific GPU model or hardware vendor. To reproduce the results, it is sufficient to use GPUs with more than 40GB of memory; no particular card type is required.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: Our research conforms to the NeurIPS Code of Ethics. We use only publicly available benchmarks and pretrained models and do not involve human subjects or sensitive personal data.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: This work proposes a general-purpose keyframe selection technique for video understanding and does not introduce new risks beyond those inherent to existing VideoLLMs. We do not foresee direct negative societal impacts from improving frame selection efficiency.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: FrameScout is a lightweight keyframe selector that does not generate new content and does not release scraped datasets or generative models with high misuse potential; therefore safeguards for dual-use risks are not applicable.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The authors have properly credited all referenced assets, and the associated licenses and terms of use have been respected.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The new assets are the FrameScout code, which are released at `https://github.com/Hubert-1102/FrameScout` together with documentation on installation, training, inference, and evaluation.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This paper does not involve crowdsourcing or research with human subjects. All evaluation is conducted on publicly available benchmarks.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve human-subject research; therefore no IRB approval is required.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were used solely for minor revisions to the writing of this paper. All core contributions, including the proposed method and experimental evaluation, are entirely our original work.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.