Re-thinking Temporal Search for Long-Form Video Understanding

CVPR 2025

Jinhui Ye[1*†], Zihan Wang[2*], Haosen Sun[2], Keshigeyan Chandrasegaran[1],
Zane Durante[1], Cristobal Eyzaguirre[1], Yonatan Bisk[3], Juan Carlos Niebles[1], Ehsan Adeli[1],
Li Fei-Fei[1], Jiajun Wu[1], Manling Li[1,2]

[1]Stanford University [2]Northwestern University [3]Carnegie Mellon University
https://longvideohaystack.github.io

2025.06.17
채진영

Contributions

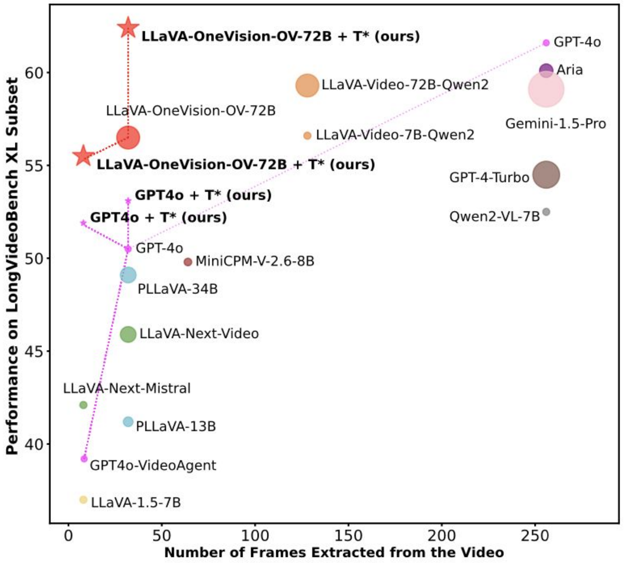
- **Long Video Haystack 문제**: 시간 검색을 특정 쿼리를 기반으로 **최소한의 관련 프레임 집합을 찾은 문제로 정의**.
- **LV-HAYSTACK 데이터셋**: search utility and efficiency를 위해 **480시간 분량의 비디오 15,092 인스턴스**로 구성된 데이터셋 소개.
- **T* framework**: **시간검색을 공간검색으로 재구성**하는 경량 시간 프레임워크인 T*제안.
- T* frame-by-frame 검색보다 3배 높은 계산 효율성 가지며, LLM 모델에 적용되는 T*는 4배 적은 프레임 사용하면서도 유사한 성능 달성함.

Introduction

- **Traditional video sampling**

Introduction

- **Problems**
  - To overcome the challenge, temporal search has emerged as a fundamental paradigm, which is framed as a Long Video Needle-in-a-Haystack.



Temporal Search in Video Understanding

- **Task formulation**
  - Query **Q**
  - Video **V** = {f1, f2, …, fN} with N frames
  - A minimal subset of k keyframes **V^k** = {f1^k, f2^k, .., fk^K} ⊆ V
  - Completeness:
    - **V^K** should be a complete frame set to answer questions
    - If the answer to **Q** based on **V** is **A**, then the answer derived from **V^K** should also be **A**.
  - Minimality
    - **V^K** should contain only essential frames.
    - No redundant or irrelevant frames while maintaining completeness.

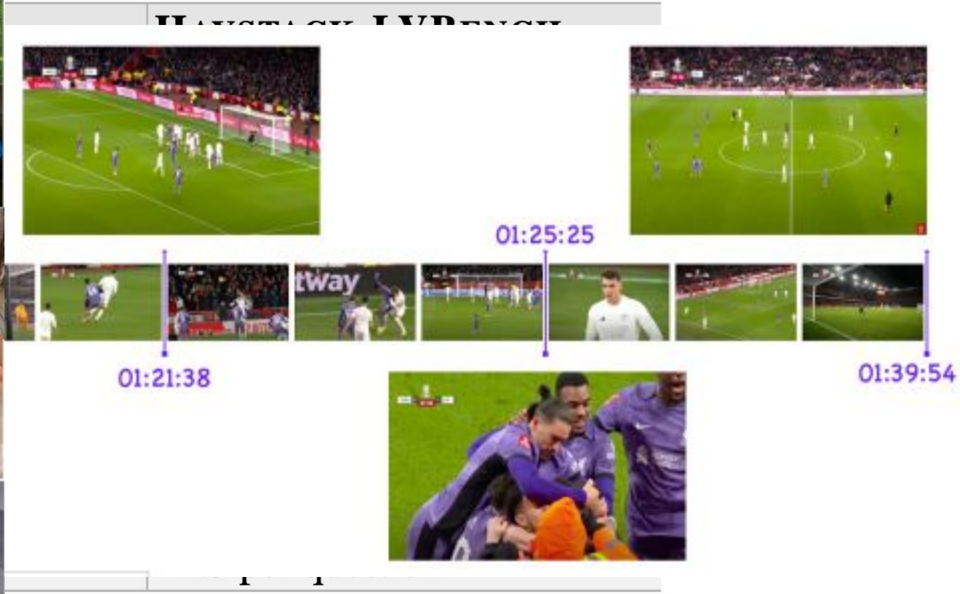
Temporal Search in Video Understanding

- **LV-HAYSTACK Benchmark**
  - The benchmark consists of both ego-centric and allocentric videos, sourced from Ego4D and LongVideoBench.

Subset	HAYSTACK-EGO4D	HAYSTACK-LVBENCH
Video Type	Egocentric	Allocentric
# video	988	114
# length	423 h - 25.7 min per video	26.7 h - 14.1 min per video
# frame	45,700,000 - 46,300 per video	2,200,000 - 19,100 per video
# QA pair	15,092 - 15.3 per video	342 - 3.0 per video
# keyframe	28,300 - 1.9 per question	496 - 1.5 per question

Temporal Search in Video Understanding

- **LV-HAYSTACK Benchmark**
  - The benchmark consists of both ego-centric and allocentric videos, sourced from Ego4D and LongVideoBench.



Temporal Search in Video Understanding

- **Evaluation metrics for search utility**
  - Frame-to-Frame Metrics
    - 1) Temporal similarity 2) Visual similarity - SSIM
  - Set-to-Set metrics

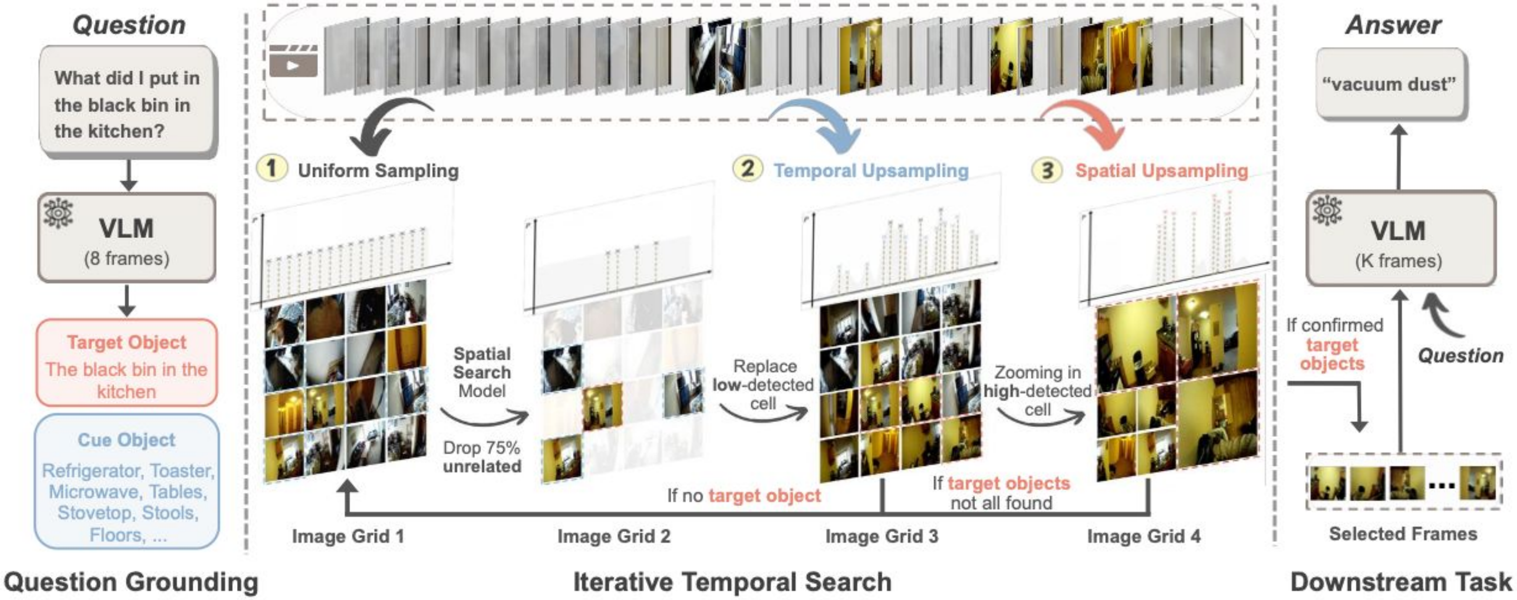
$$\text{Precision}(F_{\text{pt}}, F_{\text{gt}}) = \frac{1}{|F_{\text{pt}}|} \sum_{f_{\text{pt}}^i \in F_{\text{pt}}} \text{sim}(f_{\text{pt}}^i, F_{\text{gt}}),$$

$$\text{Recall}(F_{\text{pt}}, F_{\text{gt}}) = \frac{1}{|F_{\text{gt}}|} \sum_{f_{\text{gt}}^j \in F_{\text{gt}}} \text{sim}(f_{\text{gt}}^j, F_{\text{pt}}),$$

- Evaluation metrics for search efficiency
  - Frame cost, FLOPs, Latencys

T*: Efficient Temporal Search

- The architecture is divided to 3 parts:
  - 1) Question Grounding 2) Iterative Temporal Search 3) Downstream Task



T*: Efficient Temporal Search

- Question grounding
  - vision-language model 사용해서 텍스트 질문에서 visual object 식별.
  - what did I put in the black dustbin? -> target object: dustbin, cue objects: room corners, furniture placement

Algorithm 1: Efficient Temporal Search with Dynamic Sampling

Input : Video V, target/cue objects $\{T, C\}$, keyframe count K, search budget B, threshold θ

Output : Keyframes F with timestamps τ

Initialize: $S, N \leftarrow \mathbf{0}^L, \mathbf{1}^L$, $P \leftarrow \frac{1}{L}\mathbf{1}^L$, $R \leftarrow T$, $F, \tau \leftarrow \emptyset$; // $L = |V|$

while $R \neq \emptyset$ ***and*** $B > 0$ **do**

    $I \leftarrow \text{Sample}(P \odot N, g^2)$, $G \leftarrow \text{Grid}(V[I])$, $B \leftarrow B - g^2$; // Sample and grid

    $(C, O) \leftarrow \text{Detect}(G)$; // Get confidence maps and objects

    for $i \in [1..|I|]$ ***where*** $O_i \cap R \neq \emptyset$ **do**

        $S[I_i], N[I_i] \leftarrow C_i, 0$; // Update scores and mark visited

        if *Verify*$(V[I_i]) > \theta$ **then**

            $F, \tau \leftarrow F \cup \{V[I_i]\}, \tau \cup \{I_i/fps\}$, $R \leftarrow R \setminus (O_i \cap R)$

    $P \leftarrow \text{Normalize}(\text{Spline}(S, N))$; // Update distribution

return $\text{Sample}(V, S, K)$

T*: Efficient Temporal Search

- Iterative temporal search
  - 시간 해상도를 점진적으로 개선하면서 시간 영역을 좁혀나가는 방식으로 시간 검색 수행
  - Initialization
    - 프레임에 대한 균일한 확률 분포 P로 시작하고 객체 감지에 대한 신뢰도 threshold θ 설정

Algorithm 1: Efficient Temporal Search with Dynamic Sampling

Input : Video V, target/cue objects $\{T, C\}$, keyframe count K, search budget B, threshold θ Initialization

Output : Keyframes F with timestamps τ

Initialize: $S, N \leftarrow \mathbf{0}^L, \mathbf{1}^L, P \leftarrow \frac{1}{L}\mathbf{1}^L, R \leftarrow T, F, \tau \leftarrow \emptyset$; // $L = |V|$

while $R \neq \emptyset$ ***and*** $B > 0$ **do**

  $I \leftarrow \text{Sample}(P \odot N, g^2), G \leftarrow \text{Grid}(V[I]), B \leftarrow B - g^2$; // Sample and grid

  $(C, O) \leftarrow \text{Detect}(G)$; // Get confidence maps and objects

  for $i \in [1..|I|]$ ***where*** $O_i \cap R \neq \emptyset$ **do**

    $S[I_i], N[I_i] \leftarrow C_i, 0$; // Update scores and mark visited

    if *Verify*$(V[I_i]) > \theta$ **then**

      $F, \tau \leftarrow F \cup \{V[I_i]\}, \tau \cup \{I_i/fps\}, R \leftarrow R \setminus (O_i \cap R)$

  $P \leftarrow \text{Normalize}(\text{Spline}(S, N))$; // Update distribution

return $\text{Sample}(V, S, K)$

T*: Efficient Temporal Search

- Iterative temporal search
  - Frame Sampling and Grid Construction
    - 현재 확률 분포 P에 따라 프레임을 샘플링하고, 샘플링된 프레임을 g x g 크기의 레이아웃 G로 배열. (Line 3)

Algorithm 1: Efficient Temporal Search with Dynamic Sampling

Input : Video V, target/cue objects $\{T, C\}$, keyframe count K, search budget B, threshold θ

Output : Keyframes F with timestamps τ

1 Initialize: $S, N \leftarrow \mathbf{0}^L, \mathbf{1}^L, P \leftarrow \frac{1}{L}\mathbf{1}^L, R \leftarrow T, F, \tau \leftarrow \emptyset$; // $L = |V|$

2 **while** $R \neq \emptyset$ ***and*** $B > 0$ **do**

3 $I \leftarrow \text{Sample}(P \odot N, g^2), G \leftarrow \text{Grid}(V[I]), B \leftarrow B - g^2$; // Sample and grid

4 $(C, O) \leftarrow \text{Detect}(G)$; // Get confidence maps and objects

5 **for** $i \in [1..|I|]$ ***where*** $O_i \cap R \neq \emptyset$ **do**

6 $S[I_i], N[I_i] \leftarrow C_i, 0$; // Update scores and mark visited

7 **if** *Verify*$(V[I_i]) > \theta$ **then**

8 $F, \tau \leftarrow F \cup \{V[I_i]\}, \tau \cup \{I_i/fps\}, R \leftarrow R \setminus (O_i \cap R)$

9 $P \leftarrow \text{Normalize}(\text{Spline}(S, N))$; // Update distribution

10 **return** $\text{Sample}(V, S, K)$

T*: Efficient Temporal Search

- Iterative temporal search
  - Object Detection and Scoring
    - 사전 훈련된 모델을 사용하여 각 그리드 이미지에서 target 및 cue 객체들 식별. 각 그리드 셀 (i, j)에 대한 detection confidence는 다음과 같이 계산 (Line 5-8)

$$C_{i,j} = \max_{o \in \mathcal{D}_{i,j}} (c_o \cdot w_o)$$

- D_{i, j}: detected objects in cell (i, j)
- c_o: detection confidence , w_o: object weight

Algorithm 1: Efficient Temporal Search with Dynamic Sampling

Input : Video V, target/cue objects $\{T, C\}$, keyframe count K, search budget B, threshold θ

Output : Keyframes F with timestamps τ

1 Initialize: $S, N \leftarrow \mathbf{0}^L, \mathbf{1}^L, P \leftarrow \frac{1}{L}\mathbf{1}^L, R \leftarrow T, F, \tau \leftarrow \emptyset$; // $L = |V|$

2 **while** $R \neq \emptyset$ ***and*** $B > 0$ **do**

3 $\quad I \leftarrow \text{Sample}(P \odot N, g^2), G \leftarrow \text{Grid}(V[I]), B \leftarrow B - g^2$; // Sample and grid

4 $\quad (C, O) \leftarrow \text{Detect}(G)$; // Get confidence maps and objects

5 $\quad$ **for** $i \in [1..|I|]$ ***where*** $O_i \cap R \neq \emptyset$ **do**

6 $\quad\quad S[I_i], N[I_i] \leftarrow C_i, 0$; // Update scores and mark visited

7 $\quad\quad$ **if** *Verify*$(V[I_i]) > \theta$ **then**

8 $\quad\quad\quad F, \tau \leftarrow F \cup \{V[I_i]\}, \tau \cup \{I_i/fps\}, R \leftarrow R \setminus (O_i \cap R)$

9 $\quad P \leftarrow \text{Normalize}(\text{Spline}(S, N))$; // Update distribution

10 **return** $\text{Sample}(V, S, K)$

T*: Efficient Temporal Search

- Iterative temporal search
  - Distribution update
    - spline-based interpolation 사용해서 점수 분포를 업데이트함. 각 샘플링된 f 에 대해 점수를 업데이트 하고 방문한 것으로 표시 (Line 6)
    - temporal locality를 포착하기 위해 높은 confidence 프레임에 대해 window-based

$$S_{f\pm\delta} = \max(S_{f\pm\delta}, \frac{S_f}{|\delta|+1}), \quad \delta \in [-w, w]$$

– w: window

Algorithm 1: Efficient Temporal Search with Dynamic Sampling

Input : Video V, target/cue objects $\{T, C\}$, keyframe count K, search budget B, threshold θ

Output : Keyframes F with timestamps τ

1 Initialize: $S, N \leftarrow \mathbf{0}^L, \mathbf{1}^L$, $P \leftarrow \frac{1}{L}\mathbf{1}^L$, $R \leftarrow T$, $F, \tau \leftarrow \emptyset$; // $L = |V|$

2 **while** $R \neq \emptyset$ ***and*** $B > 0$ **do** target/cue object 모두 찾거나 search budget B 소진될때까지 검색 프로세스 반복

3 $I \leftarrow \text{Sample}(P \odot N, g^2)$, $G \leftarrow \text{Grid}(V[I])$, $B \leftarrow B - g^2$; // Sample and grid

4 $(C, O) \leftarrow \text{Detect}(G)$; // Get confidence maps and objects

5 **for** $i \in [1..|I|]$ ***where*** $O_i \cap R \neq \emptyset$ **do**

6 $S[I_i], N[I_i] \leftarrow C_i, 0$; // Update scores and mark visited

7 **if** *Verify*$(V[I_i]) > \theta$ **then**

8 $F, \tau \leftarrow F \cup \{V[I_i]\}, \tau \cup \{I_i/fps\}$, $R \leftarrow R \setminus (O_i \cap R)$

9 $P \leftarrow \text{Normalize}(\text{Spline}(S, N))$; // Update distribution

10 **return** $\text{Sample}(V, S, K)$ final score 기반해서 Top k frame을 출력

Experimental Results

- **LV-HAYSTACK search performance- search utility**

Method	Frames↓	HAYSTACK-EGO4D						HAYSTACK-LVBENCH					
		Temporal			Visual			Temporal			Visual		
		Precision ↑	Recall ↑	F_1 ↑	Precision ↑	Recall ↑	F_1↑	Precision ↑	Recall ↑	F_1 ↑	Precision ↑	Recall ↑	F_1↑
Baselines: Static Frame Sampling													
Uniform [72]	8	1.0	3.4	1.6	58.0	63.0	60.2	1.4	6.3	2.2	56.0	72.0	62.7
Uniform [72]	32	1.1	14.8	2.0	58.5	65.6	61.5	1.4	24.9	2.7	58.7	81.6	67.3
Baselines: Adaptive Temporal Search													
VideoAgent [68]	10.1	1.7	5.8	2.7	58.0	62.4	59.9	1.2	8.5	2.1	58.8	73.2	64.7
Retrieval-based	8	1.2	4.2	1.9	58.5	61.7	59.9	1.5	6.3	2.3	63.1	65.5	64.1
Retrieval-based	32	1.0	13.8	1.9	58.5	65.4	61.4	1.3	21.8	2.4	**59.9**	80.8	**67.8**
Ours: T^* for Zooming In Temporal Search													
Attention-based	8	2.2	7.5	3.3	58.4	62.5	60.2	1.5	6.6	2.4	63.6	68.6	65.7
Training-based	8	1.4	4.9	2.1	58.0	61.5	59.6	1.5	6.6	2.3	59.8	71.1	64.5
Detector-based	8	1.7	5.8	2.7	63.8	70.1	66.8	1.6	7.1	2.5	58.4	72.7	64.3
Detector-based	32	**1.8**	**26.3**	**3.4**	**62.9**	**76.2**	**68.9**	**1.7**	**28.2**	**3.1**	58.3	**83.2**	**67.8**

Attention-based:VLM attention matrix
Detector-based: object detector like YOLO-world
Training-based: custom trained model

Experimental Results

- **LV-HAYSTACK search performance- search efficiency**

Method	Search Efficiency				Overall Task Efficiency		
	Grounding	Matching	TFLOPs ↓	Latency (sec) ↓	TFLOPs ↓	Latency (sec) ↓	Acc ↑
Baselines: Static Frame Sampling							
Uniform-8 [72]	N/A	N/A	N/A	0.2	139.3	3.8	45.9
Baselines: Adaptive Temporal Search							
VideoAgent [68]	GPT4×4	CLIP-1B×840	536.5[†]	30.2	690.7[†]	34.9	49.2
Retrieval-based	N/A	YOLO-world-110M×840	216.1	28.6	355.4	32.2	50.3
Ours: T^* for Efficient Temporal Search							
Attention-based	LLaVA-72B×3	N/A	88.9	13.7	228.2	17.3	49.6
Detector-based	LLaVA-7B×1	YOLO-world-110M×49	33.3	7.5	172.6	11.1	50.8
Training-based	LLaVA-7B×1	YOLO-world-110M×38	**30.3**	**6.8**	**169.6**	**10.4**	**51.0**

Experimental Results

- **Evaluation on downstream tasks: Ego4D LongVideo QA**

Model	Frames	Tiny		Test	
		Clip	Video	Clip	Video
Baselines using Static Uniform Sampling					
GPT4o	8	45.5	41.5	45.9	45.3
GPT4o + T*	8	49.5	45.0	49.4	46.7
GPT4o	32	49.0	45.5	52.3	51.0
GPT4o + T*	32	**51.0**	46.5	54.9	52.5
QWen2.5VL 7B	8	37.0	32.0	40.0	38.8
QWen2.5VL 7B + T*	8	38.5	37.0	42.7	40.3
QWen2.5VL 7B	16	37.5	35.0	40.9	38.8
QWen2.5VL 7B + T*	16	39.5	38.5	43.8	42.8
QWen2.5VL 72B	8	45.0	45.0	51.0	50.1
QWen2.5VL 72B + T*	8	45.5	46.0	53.5	52.8
QWen2.5VL 72B	16	49.0	49.5	53.6	50.6
QWen2.5VL 72B + T*	16	50.0	**50.0**	**55.1**	**52.8**

Experimental Results

- **Evaluation on downstream tasks: Long Video QA**

LongVideoBench

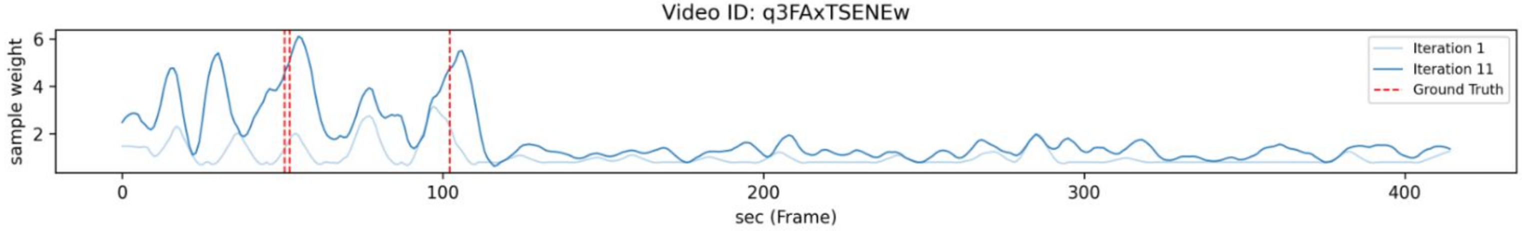
Model and Size	#Frame	XLong 15-60min	Long 2-10min	Medium 15-60s	Short 8-15s
GPT4o	8	47.1	49.4	67.3	69.7
GPT4o + *T**	8	**51.9**	**52.4**	**72.7**	**70.0**
LLaVA-OneVision-72B	8	53.7	57.4	74.1	73.0
LLaVA-OneVision-72B + *T**	8	**55.5**	**63.7**	**76.3**	**73.5**
GPT4o	32	50.5	57.3	73.5	71.4
GPT4o + *T**	32	**53.1**	**59.4**	**74.3**	**71.4**
LLaVA-OneVision-72B	32	56.5	61.6	77.4	74.3
LLaVA-OneVision-72B + *T**	32	**62.4**	**64.1**	**79.3**	**74.6**
GPT-4o (0513)	256	61.6	66.7	76.8	71.6
Aria-8x3.5B	256	60.1	64.6	76.6	69.4
LLaVA-Video-72B-Qwen2	128	59.3	63.9	77.4	72.4
Gemini-1.5-Pro (0514)	256	59.1	65.0	75.3	70.2
Qwen2-VL-7B	256	52.5	56.7	67.6	60.1

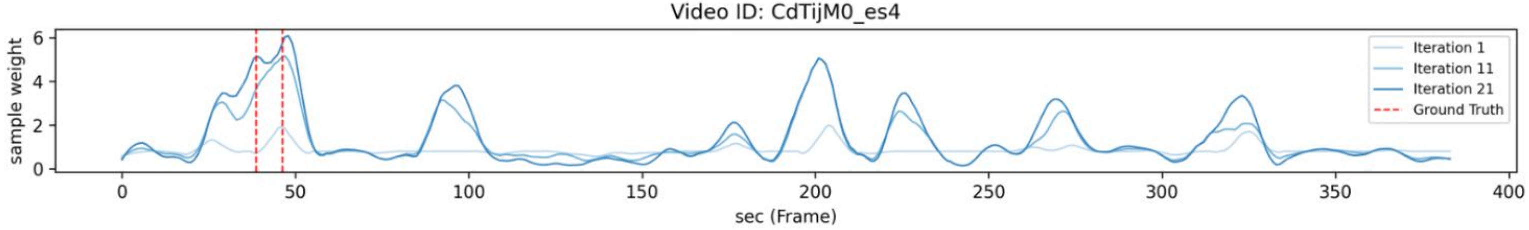
Video-MME

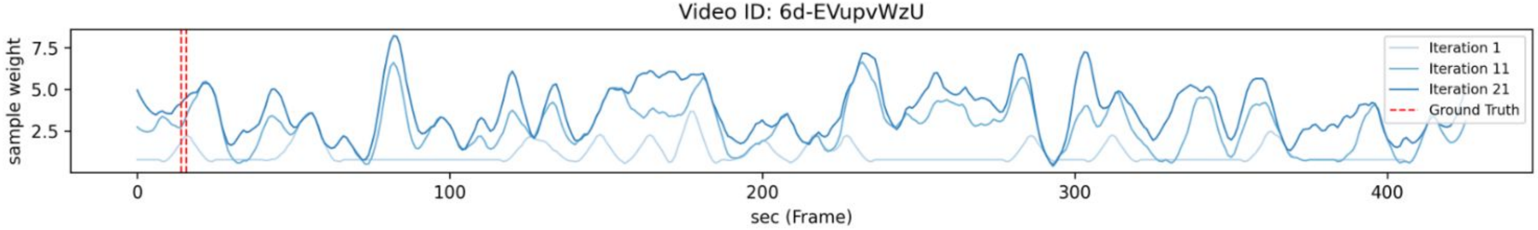
Model and Size	#Frame	Long 41min	Medium 9min	Short 1.3min	Total 17min
GPT4o	8	51.4	54.3	55.7	53.8
GPT4o + *T**	8	**55.9**	**57.3**	**56.4**	**56.5**
LLaVA-OneVision-72B	8	52.6	55.5	59.6	55.9
LLaVA-OneVision-72B + *T**	8	**57.7**	**57.5**	**61.7**	**59.0**
GPT4o	32	56.3	60.7	68.3	61.8
GPT4o + *T**	32	**59.3**	**63.5**	**69.5**	**64.1**
LLaVA-OneVision-72B	32	60.0	62.2	76.7	66.3
LLaVA-OneVision-72B + *T**	32	**61.0**	**66.6**	**77.5**	**68.3**
Gemini-1.5-Pro (0615)	1/0.5 fps[1*]	67.4	74.3	81.7	75.0
Qwen2-VL-72B	768[3*]	62.2	71.3	80.1	71.2
GPT-4o (0615)	384[2*]	65.3	70.3	80.0	71.9
LLaVA-Video-72B	64	61.5	68.9	81.4	70.6
Aria-8x3.5B	256	58.8	67.0	76.9	67.6

Analysis

- **Sampling iteration dynamics**

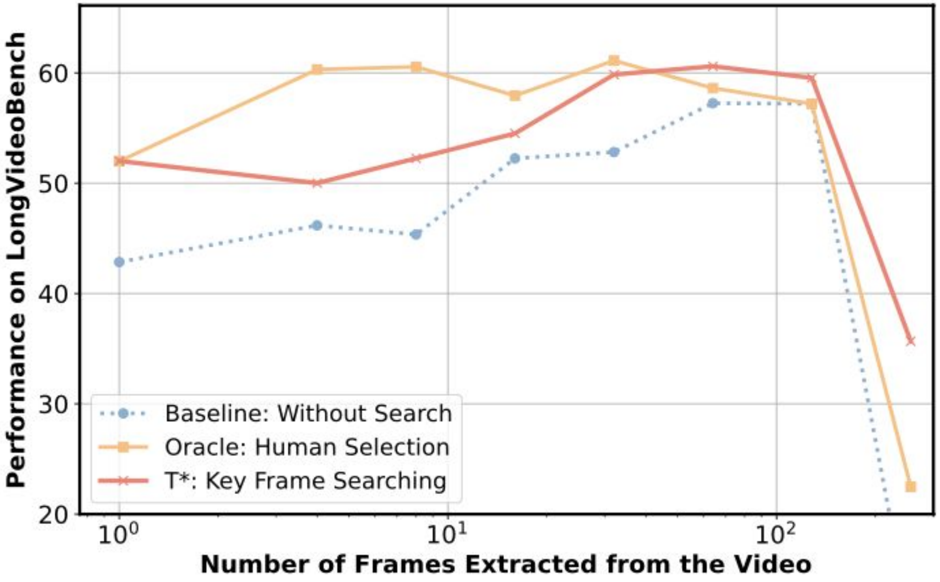






Analysis

- **Effect of search frame count on accuracy**



Conclusion

- Limitation
  - the assumption that most problems can be addressed with a few keyframes
    - not extended to more complex tasks requiring context or dense reasoning
  - focus primarily on visual cues, without leveraging other modalities