

---

# Video-RAG: Visually-aligned Retrieval-Augmented Long Video Comprehension

---

**Yongdong Luo<sup>1</sup>**   **Xiawu Zheng<sup>1\*</sup>**   **Guilin Li<sup>1</sup>**   **Shukang Yin**   **Haojia Lin<sup>1</sup>**  
**Chaoyou Fu<sup>2</sup>**   **Jinfa Huang<sup>3</sup>**   **Jiayi Ji<sup>1</sup>**   **Fei Chao<sup>1</sup>**   **Jiebo Luo<sup>3</sup>**   **Rongrong Ji<sup>1</sup>**

<sup>1</sup>Key Laboratory of Multimedia Trusted Perception and Efficient Computing,  
Ministry of Education of China, Xiamen University, 361005, P.R. China

<sup>2</sup> Nanjing University <sup>3</sup> University of Rochester

## Abstract

Existing large video-language models (LVLMs) struggle to comprehend long videos correctly due to limited context. To address this problem, fine-tuning long-context LVLMs and employing GPT-based agents have emerged as promising solutions. However, fine-tuning LVLMs would require extensive high-quality data and substantial GPU resources, while GPT-based agents would rely on proprietary models (e.g., GPT-4o). In this paper, we propose **Video Retrieval-Augmented Generation (Video-RAG)**, a training-free and cost-effective pipeline that employs visually-aligned auxiliary texts to help facilitate cross-modality alignment while providing additional information beyond the visual content. Specifically, we leverage open-source external tools to extract visually-aligned information from pure video data (e.g., audio, optical character, and object detection), and incorporate the extracted information into an existing LVLM as auxiliary texts, alongside video frames and queries, in a plug-and-play manner. Our **Video-RAG** offers several key advantages: (i) lightweight with low computing overhead due to single-turn retrieval; (ii) easy implementation and compatibility with any LVLM; and (iii) significant, consistent performance gains across long video understanding benchmarks, including Video-MME, MLVU, and LongVideoBench. Notably, our model demonstrates superior performance over proprietary models like Gemini-1.5-Pro and GPT-4o when utilized with a 72B model. Codes are available at <https://github.com/Leon1207/Video-RAG-master>.

## 1 Introduction

With the advancements in Large Language Models (LLMs), numerous studies have been conducted to enhance their ability to comprehend and process videos [12, 16, 18, 51, 24, 47, 19, 3, 2, 50, 23, 25, 17], collectively termed Large Video-Language Models (LVLMs). Although current LVLMs have demonstrated promising performance in understanding short videos, effective comprehension of extremely long videos continues to be a major challenge.

To address this challenge, recent studies [49, 45, 35, 42, 55] have sought to extend the reasoning context length of LVLMs, essentially finetuning long-context LVLMs for long video understanding. LongVA [49] first introduces increasing the token capacity of an LLM and transferring its long-context comprehension capabilities to video data. However, training such a model requires pre-training on an extended corpus, and often there are distribution shifts between deployment videos and finetuning videos. As demonstrated in Video-MME [6], LongVA declines when increasing the video frame sampling rate from 128 to 384 (52.6% → 51.8%). This outcome suggests that simply increasing the

---

\*Corresponding author: zhengxiawu@xmu.edu.cn

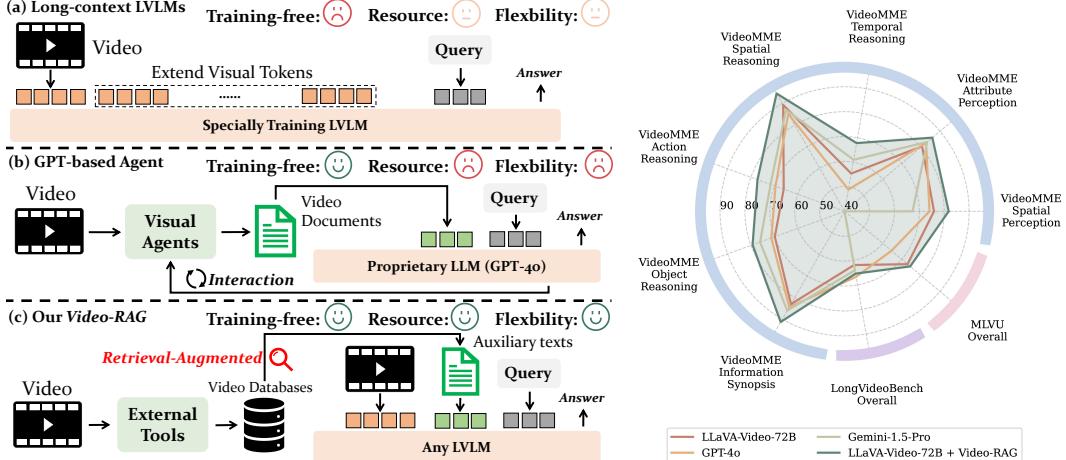


Figure 1: (Left) Two common approaches for understanding long videos, alongside our Video-RAG. Video-RAG provides a resource-efficient, training-free pipeline compatible with any LVLM. By leveraging RAG, it retrieves auxiliary texts for input, leading to notable performance enhancement. (Right) Comparison of the performance of Video-RAG with LLaVA-Video-72B [52], Gemini-1.5-Pro [32], and GPT-4o [29] across various benchmarks, including the sub-tasks from Video-MME [6] (we focus only on those that outperform Gemini-1.5-Pro), LongVideoBench [43], and MLVU [54].

number of sampled frames not only leads to information redundancy but also imposes additional challenges for the model to handle complex reasoning. Retrieval-Augmented Generation [14] (RAG) is a technique that enhances generative tasks by retrieving relevant documents from an external corpus, thus improving response quality in LLMs. Recent studies have begun exploring the integration of RAG with video-based tasks [1, 27, 48, 33], employing tools to process videos in long contexts and sending them to a proprietary model for generation, which is known as the GPT-based Agent method. However, they come with several limitations. First, most of them process long video content as plain text, subsequently utilizing the RAG mechanisms to retrieve relevant documents for LLMs. Therefore, they lack alignment with the visual context of the video, resulting in a loss of critical visual information. Second, they are often resource-intensive in multi-turn interactions and typically require powerful LLMs to function as the driving force, thus limiting their flexibility and generative capabilities. Executing the whole Video-MME [6] using VideoAgent [4] requires approximately 20 days and incurs a substantial consumption of GPT-4o API tokens.

In this study, we propose Video-RAG, an effective RAG pipeline that can be seamlessly integrated with any LVLM. Specifically, instead of simply increasing the number of sampled video frames, we propose to replace the corresponding extended visual tokens with auxiliary texts extracted from pure video data by invoking open-source foundation models, such as optical character recognition (OCR), automatic speech recognition (ASR), and object detection. These auxiliary texts are more aligned with the visual context while providing additional information beyond the visual data, as demonstrated in [20, 4]. Besides dealing with the context windows limit of LVLMs, we employ RAG in Video-RAG to filter auxiliary texts, ensuring their relevance to the user’s query in the text embedding space. As sampled visual context often lacks explicit alignment with the instructions, the auxiliary texts can facilitate cross-modality alignment while reducing the modality divide. As illustrated in Figure 5, with Video-RAG, the retrieved auxiliary texts help guide the LVLM to pay more attention to the query-relevant keyframes, while simultaneously facilitating cross-modality alignment between query and keyframes. In this framework, an LVLM serves as the central component of Video-RAG, processing visual tokens to preserve detailed visual context and minimize potential information loss. Moreover, the retrieval process is parallelly executed in a single operation, ensuring efficiency.

We evaluate Video-RAG across several long video benchmarks, including Video-MME [6], MLVU [54], and LongVideoBench [43]. By applying the Video-RAG to seven distinctive open-source LVLMs, we achieve an average performance improvement of 2.8% on Video-MME with only 2.0K text tokens addition (equal to 14 frames in most configuration) per case, while beating the proprietary LVLM when integrated with the 72B model, as shown in the right part of Figure 1. Applying Video-

RAG to a 7B LVLM only requires an additional 8GB of inference GPU memory and approximately 5 seconds of inference time per case (details in the Supplemental Material).

In summary, our contributions are as follows:

- **We integrate RAG into open-source LVLMs:** Video-RAG incorporates three types of visually-aligned auxiliary texts (OCR, ASR, and object detection) processed by external tools and retrieved via RAG, enhancing the LVLM. It's implemented using completely open-source tools, without the need for any commercial APIs.
- **We design a versatile plug-and-play RAG-based pipeline for any LVLM:** Video-RAG offers a training-free solution for a wide range of LVLMs in a plug-and-play manner, delivering performance improvements with minimal additional resource requirements.
- **We achieve proprietary-level performance with open-source models:** Applying Video-RAG to a 72B open-source model yields proprietary-level performance, surpassing models such as GPT-4o and Gemini-1.5-Pro.

## 2 Related Work

### 2.1 Large Video-Language Models

With the rapid advancement of large language models (LLMs), there has been increasing interest in developing generalist video models capable of handling video-related tasks. Video-ChatGPT [28] extracts features from individual frames and aggregates them through both spatial and temporal pooling operations. VideoChat [16] encodes videos by generating both textual descriptions and video appearance embeddings. Video-LLaVA [18] aligning image and video encoders during a pre-processing phase, using a shared projector to map the encoded representations into a common language space. LLaVA-NeXT-Video [51] extends LLaVA-NeXT [22] by fine-tuning the model specifically on video data. Despite their contributions, these approaches face challenges when processing long videos, primarily due to the limited number of frames sampled for analysis.

### 2.2 Long-context Large Video-Language Models

Recent approaches have sought to expand the context window size to enhance long video understanding. LongVA [49] and Long-LLaVA [45] address this by continuously training LLMs on extended textual data, to transfer their long-text comprehension capabilities to video processing. INTP [35] introduces a video token rearrangement technique while proposing a training-free method for extending the LLM context window, allowing LVLMs to process increased visual tokens. However, these methods face challenges in striking a balance between the high computational costs associated with sampling video frames and the limited performance improvements achieved. Due to the inherent redundancy in video content and constraints on model capacity, performance degradation may occur when the number of sampled frames surpasses a certain threshold.

### 2.3 GPT-based Agent Video Understanding

Initial efforts [46, 41, 8, 38, 33] have employed LLMs to interact with tools to process visual information as structured long context for question-answering. MM-VID [20] enhances long video understanding by aligning video frames with corresponding text descriptions. VLog [21] leverages multimodel pre-trained models to capture and interpret visual and audio information, summarizing it into documents for video comprehension. VideoAgent [4], DrVideo [27], and OmAgent [48] integrate multimodal inputs and enable dynamic querying of video segments to support long video reasoning tasks. VideoRAG [33] and VideoRAG [11] achieve a tighter integration between the RAG framework and proprietary models. However, these methods take a long time to process videos while relying on proprietary models (e.g., GPT-4o), thus limiting their efficiency and adaptability to other open-source frameworks.

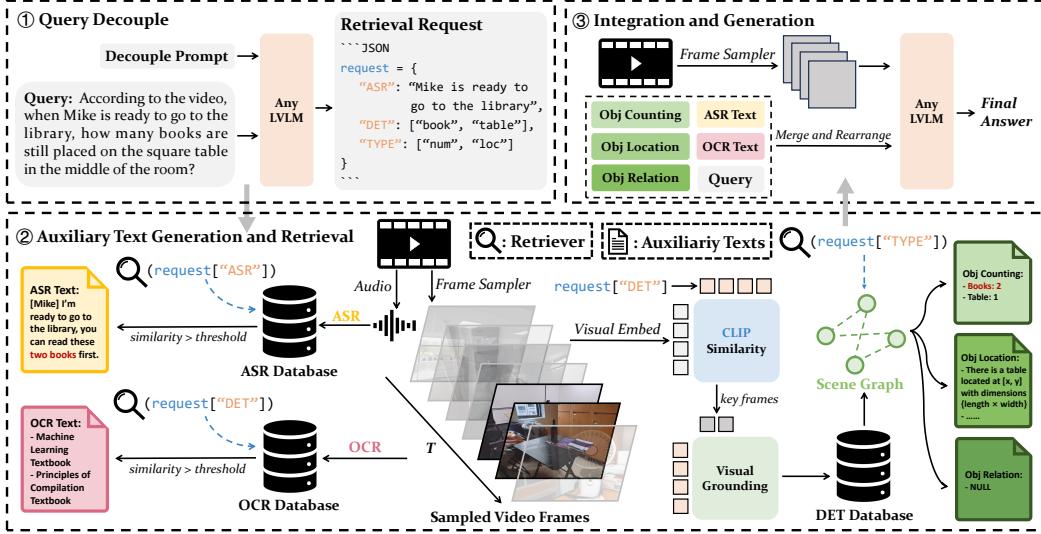


Figure 2: The framework of our Video-RAG. In the query decouple phase, the LVLM is prompted to generate a retrieval request for auxiliary texts. Next, in the auxiliary text generation and retrieval phase, the video is processed **in parallel** to extract three types of textual information (OCR, ASR, and object detection), and the relevant text is retrieved as the auxiliary text. Finally, in the integration and generation phase, auxiliary texts are combined with the query and the video to generate the response.

### 3 Method

We propose a novel, training-free pipeline for large video-language models (LVLMs), named Video-RAG, which can be integrated into any LVLM. As illustrated in Figure 2, our pipeline comprises three key phases: **(i) Query Decouple:** In this phase, the user’s query is decomposed into a retrieval request aimed at extracting auxiliary texts from the target video. **(ii) Auxiliary Text Generation & Retrieval:** Multiple auxiliary texts are generated from the queried video in parallel. Then, the retrieval request is used to obtain relevant external information. **(iii) Integration and Generation:** This phase integrates the retrieved auxiliary texts with the user’s query, feeding this combined input into the LVLMs to generate the final response.

#### 3.1 Large Video-Language Model

Given a video  $\mathbf{V}$ , a frame sampler first sample  $N$  frames  $\mathbf{F}$ . Most existing methods uniformly sample frames from a video for both effectiveness and simplicity. Then, video features are extracted as  $\mathbf{F}_v = \text{VisualEnc}(\mathbf{F})$ , where  $\text{VisualEnc}$  is an image-based visual encoder, such as CLIP-L [30]. Finally, the video features  $\mathbf{F}_v$  and the user’s query  $\mathbf{Q}$  are fed into the LVLM to generate an output  $\mathbf{O}$ :

$$\mathbf{O} = \text{LVLM}(\mathbf{F}_v, \mathbf{Q}) \quad (1)$$

#### 3.2 Query Decouple

In this phase, upon receiving a user’s query about the video, the LVLM begins by decoupling the query and generating retrieval requests, denoted as  $\mathbf{R}$ , for auxiliary texts. During this phase, the LVLM processes only textual information, without access to video frames, and the output requests are formatted in JSON. We prompt the LVLM using a decoupling prompt  $\mathbf{P}$  to generate the following retrieval requests as necessary: (i)  $\mathbf{R}_{asr}$ : Requests about automatic speech recognition, to extract audio information from the video that may pertain to the query. (ii)  $\mathbf{R}_{det}$ : Requests for identifying physical entities within the video that may assist in answering the query. (iii)  $\mathbf{R}_{type}$ : Requests for details about the location, quantity, and relationships of the identified physical entities. These requests, which may be NULL (the corresponding information is not required), are then parsed and forwarded to the auxiliary text retrieval phase. The entire process can be described as:

$$\mathbf{R} = \text{LVLM}(\mathbf{P}, \mathbf{Q}), \quad \mathbf{R} = \{\mathbf{R}_{asr}, \mathbf{R}_{det}, \mathbf{R}_{type}\} \quad (2)$$

### 3.3 Auxiliary Text Generation

In this phase, we first generate the auxiliary texts from the video and then retrieve them to assist the LVLMs according to the retrieval requests  $\mathbf{R}$ . As the length of the video increases, the number of tokens generated from the processed data also grows, leading to an increase in redundant information. Additionally, current open-source models are constrained by the limited length of their context windows, which may prevent them from fully processing all auxiliary texts. To address this issue, we draw inspiration from Retrieval-Augmented Generation (RAG) [14], retrieving only the auxiliary texts relevant to the user’s query. Before retrieval, we construct the necessary databases from the given video in parallel. Specifically, we implement three distinct databases: the Optical Character Recognition (OCR) database, denoted as  $DB_{ocr}$ ; the Automatic Speech Recognition (ASR) database, denoted as  $DB_{asr}$ ; and the Object Detection (DET) database, denoted as  $DB_{det}$ .

**OCR database.** Current LVLMs are still illusory in their ability to accurately recognize characters, and their performance often falls short compared to proprietary models. To better leverage the information contained in video frames and reduce hallucinations, we employ a proprietary OCR model to extract text from each video frame with the same frame-sampled strategy as LVLMs. Specifically, we use EasyOCR [10] as our text recognition model and segmented the recognized texts on a per-frame basis, denoted as  $\mathbf{T}_{ocr}$ . Subsequently, we implemented RAG by utilizing the advanced text encoding model Contriever [9] to encode the fetched OCR texts into text embeddings  $\mathbf{E}_{ocr}$ . These embeddings are then stored in a database with the FAISS index [13], a library designed for efficient similarity search and clustering of dense vectors. The entire building process can be formally described as:

$$\mathbf{T}_{ocr} = \text{EasyOCR}(\mathbf{F}) \quad (3)$$

$$DB_{ocr} \xleftarrow{\text{FAISS}} \mathbf{E}_{ocr} = \text{Contriever}(\mathbf{T}_{ocr}) \quad (4)$$

**ASR database.** Audio information (e.g., subtitles) plays a crucial role in video comprehension, often providing additional context that may not be available through visual cues alone. To incorporate them, we first extract the raw audio  $\mathbf{U}$  from the video and then transcribe them into texts  $\mathbf{T}_{asr}$ . Specifically, we use Whisper [31] as our audio transcription model. Since the recognized texts can be quite extensive, we chunk and encode them into a vector database, following the same procedure used to construct the OCR database. The building process can be formally described as:

$$\mathbf{T}_{asr} = \text{Whisper}(\mathbf{U}) \quad (5)$$

$$DB_{asr} \xleftarrow{\text{FAISS}} \mathbf{E}_{asr} = \text{Contriever}(\mathbf{T}_{asr}) \quad (6)$$

**DET database.** While LVLMs demonstrate strong performance in object recognition, they continue to face challenges such as object counting, precise object localization, and understanding relative relationships between objects. To mitigate the issue of hallucination, which can stem from these challenges, we incorporate object detection information as auxiliary texts. We leverage a visual grounding model to extract both the object categories and their corresponding positions from sampled video frames. This approach helps provide more accurate and context-aware object detection. To enhance processing efficiency, we limit object detection to keyframes only. Specifically, we compute the CLIP similarity [30] between the object retrieval request  $\mathbf{R}_{det}$  and the sampled video frames  $\mathbf{F}$  and select relevant keyframes  $\mathbf{F}_{key}$  based on a threshold  $t$ :

$$\mathbf{F}_{key} = \text{CLIP\_similarity}(\mathbf{R}_{det}, \mathbf{F}) > t \quad (7)$$

Once the keyframes are identified, we utilize APE [36], an efficient open-vocabulary object detection model that accepts object descriptions as prompts to detect relevant objects within frames based on retrieval queries. The capability of APE makes it particularly well-suited to our requirements for on-demand object retrieval. Finally, the detected objects’ categories and their corresponding positional information are stored in the DET database using natural language representations:

$$DB_{det} \leftarrow \mathbf{T}_{det} = \text{APE}(\mathbf{F}_{key}, \mathbf{R}_{det}) \quad (8)$$

### 3.4 Auxiliary Text Retrieval

During the retrieve phase, we employ Contriever [9] to encode the user’s query and the parsed requests for OCR and ASR into text embeddings, then concatenating to form the final query request  $\mathbf{E}_{req} = \text{Contriever}(\text{Concat}(\mathbf{R}, \mathbf{Q}))$ ,  $\mathbf{R} \in \{\mathbf{R}_{ocr}, \mathbf{R}_{asr}\}$ . Then we retrieve the auxiliary texts from  $DB \in \{DB_{ocr}, DB_{asr}\}$  by the FAISS tool, which computes the vector similarity between the query and text chunks stored in the database. Text chunks with a FAISS similarity score greater than threshold  $t$  are indexed as the retrieval results  $\mathbf{A} \in \{\mathbf{A}_{ocr}, \mathbf{A}_{asr}\}$ . The process can be formulated as:

$$\mathbf{A} \xleftarrow{\text{Index}} \text{FAISS\_similarity}(DB, \mathbf{E}_{req}) > t \quad (9)$$

Since the text generated by the detection model is in a raw format (“category: [x\_min, y\_min, length, width]”), it challenges LVLMs to understand the relative relationships between objects. We preprocess the object information using a scene graph, which helps to represent spatial and relational information more explicitly. This preprocessing allows us to construct more coherent and semantically meaningful texts, denoted as  $\mathbf{A}_{det}^p$ , which are more readily interpretable by LVLMs. We incorporate three types of object information for each video keyframe: **(i) Object Location  $\mathbf{A}_{loc}$ :** This refines the positional information of the object, formatted as: “Object {node ID} is a {object category} located at coordinates [x, y] with dimensions {length × width}” **(ii) Object Counting  $\mathbf{A}_{cnt}$ :** This counts the number of objects and generates text in the following format: “Object counting: - {object category}: {number}” **(iii) Relative Positional Relationships  $\mathbf{A}_{rel}$ :** This captures the relative spatial relationships between objects using the format: “Object {node ID} ({object category}) is <positional description> Object {node ID} ({object category})”. By combining this information, we construct a detailed representation of the objects in the frame, denoted as  $\mathbf{A}_{det}^p = \{\mathbf{A}_{loc}, \mathbf{A}_{cnt}, \mathbf{A}_{rel}\}$ :

$$\mathbf{A}_{det}^p = \text{SceneGraph}(DB_{det}) \quad (10)$$

Finally, we acquire the object auxiliary texts based on the object information type retrieval requests  $\mathbf{R}_{type}$ , which selects and finalizes the object auxiliary information  $\mathbf{A}_{det}$ .  $\mathbf{A}_{det}$  is one of the elements of the power set  $\mathcal{P}$  of  $\mathbf{A}_{det}^p$  selected by  $\mathbf{R}_{type}$ , and the retrieve process can be formulated as:

$$\mathbf{A}_{det} = \mathbf{R}_{type}(\mathcal{P}(\mathbf{A}_{det}^p)) \in \mathcal{P}(\mathbf{A}_{det}^p) \quad (11)$$

### 3.5 Integration and Generation

After obtaining different types of auxiliary texts, we organize them chronologically using natural language to create a unified auxiliary input, denoted as  $\mathbf{A}_m = \text{Concat}(\mathbf{A}_{ocr}, \mathbf{A}_{asr}, \mathbf{A}_{det})$ . These merged auxiliary inputs, along with the user’s query and the sampled video frames, are then fed into the LVLM to produce the final result. The overall process can be formulated as:

$$\mathbf{O} = \text{LVLM}(\mathbf{F}_v, \text{Concat}(\mathbf{A}_m, \mathbf{Q})) \quad (12)$$

## 4 Experiments

### 4.1 Datasets

**Video-MME** [6] is a widely used benchmark for assessing the ability of LVLMs to handle detailed videos in real-world scenarios. It is divided into three subsets based on video length, with durations ranging from 11 seconds to 1 hour. **MLVU** [54] is a long video understanding benchmark with a large wide of 9 distinct tasks. It is created based on long videos of diversified lengths, ranging from 3 minutes to 2 hours with about 12 minutes average video length. **LongVideoBench** [43] is a benchmark designed to accurately retrieve and reason over detailed multimodal information from long videos, with 6,678 human-annotated multiple-choice questions in 17 fine-grained categories.

### 4.2 Implementation Details

We performed all experiments on NVIDIA A100 80G GPUs. During the auxiliary text generation phase, we first restrict the detection requests  $\mathbf{R}_{det}$  generated by LVLMs in decouple prompt then

Table 1: Performance on the Video-MME [6] benchmark in without subtitles (w/o S), with subtitles (w/ S) and equipped with our Video-RAG (Ours). **Frames** and **Gain** means the input frame number and performance gain by applying Video-RAG compared to the baseline with subtitles. By applying Video-RAG to seven LVLMs, we observed an average performance improvement of 2.8% by adding only an average of ~2.0K auxiliary texts compared to ~3.0K full-subtitled tokens per sample. In particular, we perform better when applying Video-RAG with 72B LLaVA-Video [52] than the proprietary method GPT-4o [29] (77.4% vs. 77.2%). All results are our republication.

Model	#Params Frames		Short			Medium			Long			Overall			Gain
	w/o S	w/ S	Ours	w/o S	w/ S	Ours	w/o S	w/ S	Ours	w/o S	w/ S	Ours	w/o S	w/ S	Ours
<i>Proprietary LVLMs</i>															
GPT-4o [29]	-	384	80.0	82.8	-	70.3	76.6	-	65.3	72.1	-	71.9	77.2	-	-
Gemini-1.5-Pro [32]	-	0.5 fps	81.7	84.5	-	74.3	81.0	-	67.4	77.4	-	75.0	81.3	-	-
<i>Open-Source LVLMs</i>															
Video-LLaVA [18]	7B	8	45.3	46.1	49.5	38.0	40.7	43.0	36.2	38.1	42.5	39.9	41.6	45.0	+3.4
LLaVA-NeXT-Video [51]	7B	16	49.4	51.8	56.6	43.0	46.4	47.4	36.7	44.9	46.0	43.0	47.7	50.0	+2.3
VITA-1.5 [7]	7B	16	67.0	69.9	71.0	54.2	55.7	55.4	47.1	50.4	52.4	56.1	58.7	59.6	+0.9
LongVA [49]	7B	128	61.1	61.2	66.1	50.4	53.8	60.4	46.2	52.9	59.4	52.6	56.0	62.0	+6.0
Long-LLaVA [45]	7B	64	61.9	62.4	67.1	51.4	56.2	60.4	45.4	54.7	60.1	52.9	57.8	62.6	+4.8
Qwen2-VL [40]	72B	32	75.0	76.7	77.4	63.3	69.9	70.2	56.3	69.2	71.0	64.9	71.9	72.9	+1.0
LLaVA-Video [52]	72B	64	80.7	81.8	82.8	68.7	73.8	76.3	62.1	72.2	73.1	70.3	75.9	77.4	+1.5

further filter them using spaCy, ensuring they correspond to CLIP-sensitive physical entities, avoiding the inclusion of abstract concepts. In the auxiliary text retrieval phase, we set both the CLIP similarity threshold and the FAISS similarity threshold  $t$  to 0.3. We employ the IndexFlatIP as the similarity calculating method of FAISS [13]. Note that we don't include the GPT-based Agent methods for comparison due to their resource-intensive nature (complete execution of Video-MME [6] costs around \$2000 for API purchasing when using VideoAgent [4]). Still, we include a mini-experiment of VideoAgent in the Supplemental Material that compares the overall performance, inference time, and GPU requirements with two common long-context LVLMs and our Video-RAG.

Table 2: The overall performance in the multiple-choice task of the MLVU [54] benchmark. \* denotes the results of our replication.

Model	#Params		Frames	Overall
	Proprietary	LVLMs		
<i>Proprietary LVLMs</i>				
GPT-4o [29]	-	0.5 fps		64.6
<i>Open-Source LVLMs</i>				
VITA-1.5 [7]	7B	16		60.4
Video-CCAM [5]	14B	96		63.1
Video-XL [37]	7B	256		64.9
Aria [15]	25.3B	256		70.6
LLaVA-Video* [52]	7B	64		70.8
Oryx-1.5 [26]	32B	128		72.3
LLaVA-Video* [52]	72B	64		73.1
LLaVA-Video + Video-RAG	7B	64		72.4
LLaVA-Video + Video-RAG	72B	64		<b>73.8</b>

Table 3: The overall performance on the validation set of LongVideoBench [43]. \* denotes the results of our replication.

Model	#Params		Frames	Overall
	Proprietary	LVLMs		
<i>Proprietary LVLMs</i>				
Gemini-1.5-Pro [32]	-	256		64.0
GPT-4o [29]	-	256		<b>66.7</b>
<i>Open-Source LVLMs</i>				
VideoChat2-Mistral [16]	7B	8		39.3
ShareGPT4Video [2]	7B	8		39.7
LLaVA-Next-Mistral [22]	7B	8		49.1
PLLaVA [44]	34B	16		53.2
VITA-1.5 [7]	7B	16		53.6
LLaVA-Video* [52]	7B	64		56.6
LLaVA-Video* [52]	72B	64		61.9
LLaVA-Video + Video-RAG	7B	64		58.7
LLaVA-Video + Video-RAG	72B	64		<b>65.4</b>

### 4.3 Main Results

**Video-MME.** We evaluate our Video-RAG in five 7B open-source LVLMs, including Video-LLaVA [18], LLaVA-NeXT-Video [51], LongVA [49], Long-LLaVA [45], and two 72B LVLM Qwen2-VL [40] and LLaVA-Video [52]. Constraining by computational resources, we evaluate the LVLMs with their official frame rate setting in Video-MME except for 72B Qwen2-VL, which requires about 3K GPU memory with 768 video frame input (~38 A100 GPUs). Results are shown in Table 1. Specifically, after applying our Video-RAG in 72B LLaVA-Video [52], we perform better than the proprietary model GPT-4o [29] (77.4% vs. 77.2%). Across the seven LVLMs used in our experiments, we gained an average performance boost of 2.8% compared to results with subtitles, especially a significant gain on long videos, demonstrating its effectiveness. This performance improvement is

achieved by incorporating token counts from approximately 14 additional video frames (equivalent to 2.0K tokens), each contributing around 144 tokens under most LVLM configurations. We obtain such a large performance enhancement because most LVLMs are pre-trained primarily within the text space and aligned with visual information, often lacking explicit alignment between embedding spaces. Auxiliary texts can serve as semantic supplements sensitive to LVLMs, facilitating model activation and easing the understanding of complex videos.

**MLVU.** We evaluate Video-RAG when integrating into the 7B and 72B LLaVA-Video [52] of MLVU [54], a benchmark that is close to performance saturation. As shown in Table 4, Video-RAG’s 1.6% improvement at 7B-scale is substantial, considering that it outperforms the 32B Qryx-1.5 [26] by 0.1%, while recent 7B-scale models average only a 1.3% gain (across 15 approaches in MLVU’s leaderboard). Additionally, the 72B LLaVA-Video also has a performance gain of 0.7%, which sets a new state-of-the-art.

**LongVideoBench.** We evaluate Video-RAG when applied in the 7B and 72B LLaVA-Video [52] of LongVideoBench [43]. We omit the interleaved input format introduced in LongVideoBench when applying Video-RAG. The evaluation results in Table 5 demonstrate that 72B LLaVA-Video with our Video-RAG achieves an overall performance of 65.4% on the validation set. This result surpasses the proprietary LVLM Gemini-1.5-Pro [32] by 1.4%, securing the second place, just 1.3% behind GPT-4o [29]. Meanwhile, the 7B LLaVA-Video also has a performance enhancement of 2.1% when equipped with our Video-RAG.

#### 4.4 Ablation Studies

**Effect of different sampling frame number.** To explore the effect of the number of sampling frames on Video-RAG, we experience sampling frames number of 8, 16, 32, 64, 128, and 256 in 7B model LongVA [49], results are shown in Figure 3. As demonstrated, Video-RAG consistently delivers performance improvements across all frame rates, especially in long videos. The experimental results also indicate that Video-RAG can achieve higher performance gains with fewer frames, demonstrating its potential for applications under resource-constrained conditions.

**Effect of different components of Video-RAG.** To explore the effectiveness of auxiliary texts, we add DET, OCR, and ASR as auxiliary texts before and after retrieving by the RAG to evaluate Long-LLaVA-7B [45] with 32-frame setting in the Video-MME [6] benchmark. As shown in Table 4, the performance of Long-LLaVA progressively improves as auxiliary texts after retrieving by the RAG system are incrementally added ( $52.0\% \rightarrow 52.9\% \rightarrow 55.7\% \rightarrow 62.1\%$ ). Among these components, ASR auxiliary texts contribute to a general improvement for different video durations, especially for long videos. When all components are integrated, we obtain an optimal performance, as shown in the last row of Table 4. Meanwhile, the experiment shows a 2.3% improvement (59.8% vs 62.1%) in performance after incorporating RAG for retrieval, demonstrating that auxiliary texts after retrieving by the RAG system are query-aligned, which helps cross-modality alignment. We also evaluate across sub-tasks within Video-MME [6] and other video benchmarks like MLVU [54], LongVideoBench[43], and VN Bench [53], more details are shown in the Supplemental Material.

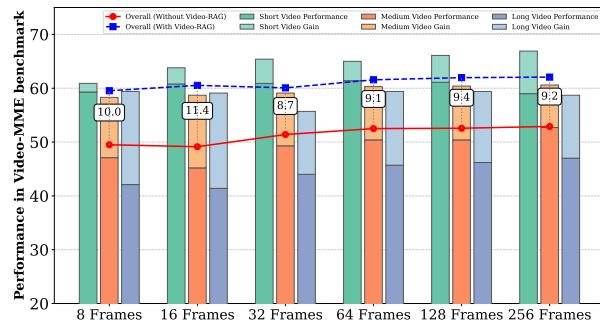


Figure 3: Performance gain with different sampling frames rate on Video-MME [6] when implement LongVA-7B [45].

**Effect of different thresholds of RAG processing.** When retrieving, we specify a similarity threshold  $t$  as a criterion for information selection. In the retrieval for OCR and ASR texts, information is selected if its FAISS similarity exceeds  $t$ . For object detection, frames are selected as keyframes based on their CLIP similarity surpassing  $t$ , and the relevant information is then extracted. Setting  $t$  too high may hinder the retrieval of relevant information while setting it too low can result in information redundancy and increased reasoning complexity. To investigate this trade-off, we conduct ablation experiments to evaluate the impact of different threshold values. The results are shown in Table 5.

Table 4: Results on combinations of different auxiliary texts in Video-MME [6] when using Long-LLaVA-7B [45] as the LVLM.

RAG	DET	OCR	ASR	Short	Medium	Long	Overall
✓	✓	✓	✓	60.3	51.4	44.1	52.0
				62.2	55.4	54.4	57.4
	✓	✓	✓	64.0	56.2	55.0	58.4
				63.0	57.3	<b>56.4</b>	58.9
✓	✓	✓	✓	<b>64.3</b>	<b>58.8</b>	<b>56.3</b>	<b>59.8</b>
✓	✓	✓	✓	61.4	51.9	45.2	52.9
✓	✓	✓	✓	63.2	53.2	46.3	54.3
✓	✓	✓	✓	65.1	59.1	<b>60.7</b>	61.6
✓	✓	✓	✓	64.1	54.6	48.4	55.7
✓	✓	✓	✓	64.9	59.0	<b>60.7</b>	61.5
✓	✓	✓	✓	66.3	<b>60.3</b>	59.3	62.0
✓	✓	✓	✓	<b>66.4</b>	60.2	59.8	<b>62.1</b>

Table 5: Performance with different thresholds of retrieval on Video-MME [6] when using Long-LLaVA-7B [45] as the LVLM. #Token and Time denote the total token number of the auxiliary texts and the average inference time per question, respectively.

t	#Token	Time	Short	Medium	Long	Overall
0.0	3.6K	36s	<b>67.6</b>	59.4	59.1	62.0
0.1	3.4K	30s	<b>67.0</b>	<b>59.7</b>	59.1	61.9
0.2	2.7K	18s	66.0	<b>60.2</b>	<b>59.2</b>	61.8
0.3	1.9K	11s	66.4	<b>60.2</b>	<b>59.8</b>	<b>62.1</b>
0.4	0.8K	8s	65.6	58.0	58.3	60.6
0.5	0.3K	7s	63.1	54.9	50.2	56.1
1.0	0.0K	6s	60.3	51.4	44.1	52.0
rnd	1.9K	11s	65.7	55.8	56.0	59.1

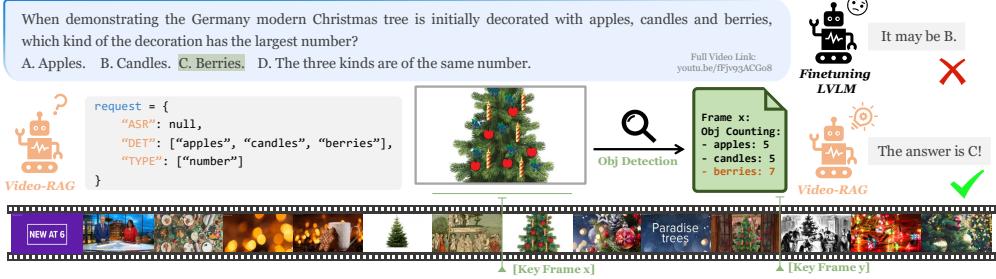


Figure 4: Qualitative result on Video-MME [6] when applying Video-RAG with LLaVA-Video [52].

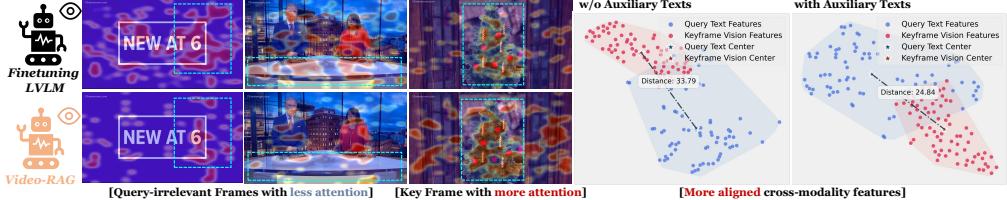


Figure 5: Grad-CAM visualizations of the last hidden state heatmap along with t-SNE visualizations of the user’s query and keyframe features of the example shown in Figure 4. The retrieved auxiliary texts help **cross-modality alignment** by assisting the model to **pay more attention to query-relevant keyframes** and thus generate more robust and accurate answers to the user’s query.

Notably,  $t = 0$  and  $t = 1$  correspond to all auxiliary texts input into the model and no auxiliary texts input, respectively. To balance performance with information density and processing time (especially APE [36] detection in keyframes), we selected a threshold of 0.3 for our implementation. More details about similarity scores are shown in the Supplemental Material. Under this configuration, the additional text length of approximately 1.9K tokens typically remains within the context window limits of open-source LVLMs. For models with more stringent context window limitations, a threshold of 0.4 may also be a viable option. We also randomly sample an equivalent token number of auxiliary texts to serve as inputs for assessing the effectiveness of RAG retrieval, as shown in the last row of Table 5.

#### 4.5 Qualitative Evaluation

We present qualitative results in the case of Video-MME [6] in Figure 4 and Figure 5. As illustrated, augmenting LLaVA-Video with external tools to process and retrieve auxiliary texts from videos significantly enhances its ability to reduce visual hallucinations, thereby enabling more accurate responses to user queries. Grad-CAM [34] and t-SNE [39] visualization results also show that applying Video-RAG helps the LVLM’s cross-modality alignment.

## 5 Conclusion

In this paper, we present Video-RAG for effective long video understanding through integrating retrieved auxiliary texts with LVLMs, achieving proprietary-level performance with 72B open-source LVLM. Unlike traditional methods that are resource-intensive with limited gains, Video-RAG offers a resource-efficient, plug-and-play solution leveraging only open-source tools to extract visually-aligned auxiliary texts from video data. However, Video-RAG may be limited by the visual tools we choose and their performance, which lacks adaptation. In the future, we will explore how to more efficiently integrate auxiliary texts and provide an adaptive frame selection strategy for LVLMs.

## 6 Acknowledge

This work was supported by the National Science Fund for Distinguished Young Scholars (No.62025603), the National Natural Science Foundation of China (No. U21B2037, No. U22B2051, No. U23A20383, No. U21A20472, No. 62176222, No. 62176223, No. 62176226, No. 62072386, No. 62072387, No. 62072389, No. 62002305 and No. 62272401), the Natural Science Foundation of Fujian Province of China (No. 2021J06003, No.2022J06001), and the Fundamental Research Funds for the Central Universities.

## References

- [1] Md Adnan Arefeen, Biplob Debnath, Md Yusuf Sarwar Uddin, and Srimat Chakradhar. irag: An incremental retrieval augmented generation system for videos. *arXiv preprint arXiv:2404.12309*, 2024.
- [2] Lin Chen, Xilin Wei, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Bin Lin, Zhenyu Tang, et al. Sharegpt4video: Improving video understanding and generation with better captions. *arXiv preprint arXiv:2406.04325*, 2024.
- [3] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24185–24198, 2024.
- [4] Yue Fan, Xiaojian Ma, Rujie Wu, Yuntao Du, Jiaqi Li, Zhi Gao, and Qing Li. Videoagent: A memory-augmented multimodal agent for video understanding. In *European Conference on Computer Vision*, pages 75–92. Springer, 2025.
- [5] Jiajun Fei, Dian Li, Zhidong Deng, Zekun Wang, Gang Liu, and Hui Wang. Video-ccam: Enhancing video-language understanding with causal cross-attention masks for short and long videos. *arXiv preprint arXiv:2408.14023*, 2024.
- [6] Chaoyou Fu, Yuhan Dai, Yondong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. *arXiv preprint arXiv:2405.21075*, 2024.
- [7] Chaoyou Fu, Haojia Lin, Xiong Wang, Yi-Fan Zhang, Yunhang Shen, Xiaoyu Liu, Haoyu Cao, Zuwei Long, Heting Gao, Ke Li, et al. Vita-1.5: Towards gpt-4o level real-time vision and speech interaction. *arXiv preprint arXiv:2501.01957*, 2025.
- [8] Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14953–14962, 2023.
- [9] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*, 2021.
- [10] JaidedAI. Easyocr. <https://github.com/JaidedAI/EasyOCR>, 2023.

- [11] Soyeong Jeong, Kangsan Kim, Jinheon Baek, and Sung Ju Hwang. Videorag: Retrieval-augmented generation over video corpus. *arXiv preprint arXiv:2501.05874*, 2025.
- [12] Peng Jin, Ryuichi Takanobu, Caiwan Zhang, Xiaochun Cao, and Li Yuan. Chat-univi: Unified visual representation empowers large language models with image and video understanding. *arXiv preprint arXiv:2311.08046*, 2023.
- [13] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.
- [14] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.
- [15] Dongxu Li, Yudong Liu, Haoning Wu, Yue Wang, Zhiqi Shen, Bowen Qu, Xinyao Niu, Guoyin Wang, Bei Chen, and Junnan Li. Aria: An open multimodal native mixture-of-experts model. *arXiv preprint arXiv:2410.05993*, 2024.
- [16] KunChang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhui Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*, 2023.
- [17] Yanwei Li, Chengyao Wang, and Jiaya Jia. Llama-vid: An image is worth 2 tokens in large language models. In *European Conference on Computer Vision*, pages 323–340. Springer, 2025.
- [18] Bin Lin, Bin Zhu, Yang Ye, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*, 2023.
- [19] Ji Lin, Hongxu Yin, Wei Ping, Pavlo Molchanov, Mohammad Shoeybi, and Song Han. Vila: On pre-training for visual language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26689–26699, 2024.
- [20] Kevin Lin, Faisal Ahmed, Linjie Li, Chung-Ching Lin, Ehsan Azarnasab, Zhengyuan Yang, Jianfeng Wang, Lin Liang, Zicheng Liu, Yumao Lu, et al. Mm-vid: Advancing video understanding with gpt-4v (ision). *arXiv preprint arXiv:2310.19773*, 2023.
- [21] Qinghong Lin. Vlog: Transform video as a document with chatgpt, clip, blip2, grit, whisper, langchain. <https://github.com/showlab/VLog>, 2023.
- [22] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024. URL <https://llava-vl.github.io/blog/2024-01-30-llava-next/>.
- [23] Jiajun Liu, Yibing Wang, Hanghang Ma, Xiaoping Wu, Xiaoqi Ma, xiaoming Wei, Jianbin Jiao, Enhua Wu, and Jie Hu. Kangaroo: A powerful video-language model supporting long-context video input. *arXiv preprint arXiv:2408.15542*, 2024.
- [24] Ruyang Liu, Chen Li, Haoran Tang, Yixiao Ge, Ying Shan, and Ge Li. St-llm: Large language models are effective temporal learners. In *European Conference on Computer Vision*, pages 1–18. Springer, 2025.
- [25] Zuyan Liu, Yuhao Dong, Ziwei Liu, Winston Hu, Jiwen Lu, and Yongming Rao. Oryx mllm: On-demand spatial-temporal understanding at arbitrary resolution. *arXiv preprint arXiv:2409.12961*, 2024.
- [26] Zuyan Liu, Yuhao Dong, Ziwei Liu, Winston Hu, Jiwen Lu, and Yongming Rao. Oryx mllm: On-demand spatial-temporal understanding at arbitrary resolution. *arXiv preprint arXiv:2409.12961*, 2024.
- [27] Ziyu Ma, Chenhui Gou, Hengcan Shi, Bin Sun, Shutao Li, Hamid Rezatofighi, and Jianfei Cai. Drvideo: Document retrieval based long video understanding. *arXiv preprint arXiv:2406.12846*, 2024.

- [28] Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. *arXiv preprint arXiv:2306.05424*, 2023.
- [29] OpenAI. Gpt-4o system card. <https://openai.com/index/gpt-4o-system-card/>, 2024.
- [30] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- [31] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pages 28492–28518. PMLR, 2023.
- [32] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.
- [33] Xubin Ren, Lingrui Xu, Long Xia, Shuaiqiang Wang, Dawei Yin, and Chao Huang. Videorag: Retrieval-augmented generation with extreme long-context videos. *arXiv preprint arXiv:2502.01549*, 2025.
- [34] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.
- [35] Yuzhang Shang, Bingxin Xu, Weitai Kang, Mu Cai, Yuheng Li, Zehao Wen, Zhen Dong, Kurt Keutzer, Yong Jae Lee, and Yan Yan. Interpolating video-lmms: Toward longer-sequence lmms in a training-free manner. *arXiv preprint arXiv:2409.12963*, 2024.
- [36] Yunhang Shen, Chaoyou Fu, Peixian Chen, Mengdan Zhang, Ke Li, Xing Sun, Yunsheng Wu, Shaohui Lin, and Rongrong Ji. Aligning and prompting everything all at once for universal visual perception. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13193–13203, 2024.
- [37] Yan Shu, Peitian Zhang, Zheng Liu, Minghao Qin, Junjie Zhou, Tiejun Huang, and Bo Zhao. Video-xl: Extra-long vision language model for hour-scale video understanding. *arXiv preprint arXiv:2409.14485*, 2024.
- [38] Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11888–11898, 2023.
- [39] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.
- [40] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Kejin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model’s perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.
- [41] Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. Videoagent: Long-form video understanding with large language model as agent. *arXiv preprint arXiv:2403.10517*, 2024.
- [42] Xidong Wang, Dingjie Song, Shunian Chen, Chen Zhang, and Benyou Wang. Longllava: Scaling multi-modal lmms to 1000 images efficiently via hybrid architecture. *arXiv preprint arXiv:2409.02889*, 2024.
- [43] Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. Longvideobench: A benchmark for long-context interleaved video-language understanding. *arXiv preprint arXiv:2407.15754*, 2024.

- [44] Lin Xu, Yilin Zhao, Daquan Zhou, Zhijie Lin, See Kiong Ng, and Jiashi Feng. Pllava: Parameter-free llava extension from images to videos for video dense captioning. *arXiv preprint arXiv:2404.16994*, 2024.
- [45] Yin Song and Chen Wu and Eden Duthie. aws-prototyping/long-llava-qwen2-7b, 2024. URL <https://huggingface.co/aws-prototyping/long-llava-qwen2-7b>.
- [46] Ce Zhang, Taixi Lu, Md Mohaiminul Islam, Ziyang Wang, Shoubin Yu, Mohit Bansal, and Gedas Bertasius. A simple llm framework for long-range video question-answering. *arXiv preprint arXiv:2312.17235*, 2023.
- [47] Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding. *arXiv preprint arXiv:2306.02858*, 2023.
- [48] Lu Zhang, Tiancheng Zhao, Heting Ying, Yibo Ma, and Kyusong Lee. Omagent: A multi-modal agent framework for complex video understanding with task divide-and-conquer. *arXiv preprint arXiv:2406.16620*, 2024.
- [49] Peiyuan Zhang, Kaichen Zhang, Bo Li, Guangtao Zeng, Jingkang Yang, Yuanhan Zhang, Ziyue Wang, Haoran Tan, Chunyuan Li, and Ziwei Liu. Long context transfer from language to vision. *arXiv preprint arXiv:2406.16852*, 2024.
- [50] Yi-Fan Zhang, Qingsong Wen, Chaoyou Fu, Xue Wang, Zhang Zhang, Liang Wang, and Rong Jin. Beyond llava-hd: Diving into high-resolution large multimodal models. *arXiv preprint arXiv:2406.08487*, 2024.
- [51] Yuanhan Zhang, Bo Li, haotian Liu, Yong jae Lee, Liangke Gui, Di Fu, Jiashi Feng, Ziwei Liu, and Chunyuan Li. Llava-next: A strong zero-shot video understanding model, April 2024. URL <https://llava-vl.github.io/blog/2024-04-30-llava-next-video/>.
- [52] Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. Video instruction tuning with synthetic data, 2024. URL <https://arxiv.org/abs/2410.02713>.
- [53] Zijia Zhao, Haoyu Lu, Yuqi Huo, Yifan Du, Tongtian Yue, Longteng Guo, Bingning Wang, Weipeng Chen, and Jing Liu. Needle in a video haystack: A scalable synthetic framework for benchmarking video mllms. *arXiv preprint*, 2024.
- [54] Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Shitao Xiao, Xi Yang, Yongping Xiong, Bo Zhang, Tiejun Huang, and Zheng Liu. Mlvu: A comprehensive benchmark for multi-task long video understanding. *arXiv preprint arXiv:2406.04264*, 2024.
- [55] Yongshuo Zong, Ismail Elezi, Yongxin Yang, Jiankang Deng, and Timothy Hospedales. Long-context vision large language models: Empirical insights and a baseline. In *Workshop on Long Context Foundation Models*, 2024.

# Supplemental Material

## A Decouple Query

In the initial phase of the proposed Video-RAG, we employ a decouple prompt, denoted as  $\mathbf{P}$ , to guide the LVLM in generating retrieval requests. In this section, we present one example of a prompt designed for multiple-choice questions, as illustrated in Figure 8.

## B Sub-set of Video-MME

As outlined in the implementation details, we randomly sampled a subset of the Video-MME [6] dataset to evaluate a computationally resource-intensive, agent-based method with long-context LVLMs. Specifically, we selected 10% of the full dataset, comprising 30 short, 30 medium-length, and 30 long videos. Each video contains three multiple-choice questions. Importantly, we ensured that the performance ranking of the methods on the subset mirrored that of the full dataset. As shown in Tables 6 and 7, we evaluated four distinct 7B models Chat-Uni-v1.5 [12], LLaVA-NeXT-Video [51], LongVA [49], and Long-LLaVA [45] using a frame sampling rate of 16 for both the subset and the full set. Our results indicate that the performance rankings remained consistent across both evaluations.

Table 6: Performance of Video-MME sub-set.

Method	Short	Medium	Long	Overall
Chat-Uni-v1.5 [12]	50.0	33.3	17.8	33.7
LLaVA-NeXT-Video [51]	54.4	33.3	23.3	37.0
LongVA [49]	56.7	50.0	38.9	48.5
Long-LLaVA [45]	58.9	52.2	40.0	50.4

Table 7: Performance of Video-MME full-set.

Method	Short	Medium	Long	Overall
Chat-Uni-v1.5 [12]	45.7	39.0	35.7	40.1
LLaVA-NeXT-Video [51]	51.1	41.8	36.8	43.2
LongVA [49]	60.8	45.2	41.4	49.1
Long-LLaVA [45]	59.3	49.3	44.4	51.0

## C Results on Video-MME Sub-Set

We examine Video-RAG against two representative methods in terms of inference time, GPU resource requirements, and overall performance. Given that GPT-based Agent methods are resource-intensive, we randomly sampled a sub-set of the Video-MME [6] for evaluation, as described in Section B. As demonstrated in Figure 6, VideoAgent [4], a typically GPT-based Agent method, requires significant time to process video and deliver suboptimal performance. Meanwhile, LongVA [49], a representative long-context LVLM, shows limited improvement from increasing the frame rate and even experiences performance degradation. Integrating our Video-RAG into the 16-frame LongVA results in substantial performance improvements while reducing GPU resource consumption. Specifically, with only increasing 8GB GPU memory compared to the base (16-frames LongVA), we achieve 11.5% overall performance improvement, while outperforming another long-context LVLM Long-LLaVA-7B [45] in 16-frames setting by 9.6% with less GPU memory requirements and compatible total inference time. These results demonstrated that our Video-RAG is lightweight with lower computing overhead than the other typical methods. Moreover, we provide detailed time consuming to construct three types of databases (which can be built in parallel) and inference per query, as shown in Table 8.

Table 8: Overall performance, databases construct and average inference time (include building databases) per query (#Time) in Video-MME-mini.

Model	ASR	OCR	DET	Total Time	w/o subs		w/ Video-RAG	
					#Time	Overall	#Time	Overall
VideoAgent	-	-	-	-	14min	47.7	-	-
LongVA-16fs	21min	2min	3min	max(21, 2, 3)=21min	1s	48.5	1s + 5s	60.0
LongVA-128fs	21min	16min	16min	max(21, 16, 16)=21min	8s	54.1	8s + 5s	63.3
LongVA-384fs	42min	48min	24min	max(42, 48, 24)=48min	20s	53.7	20s + 11s	63.6

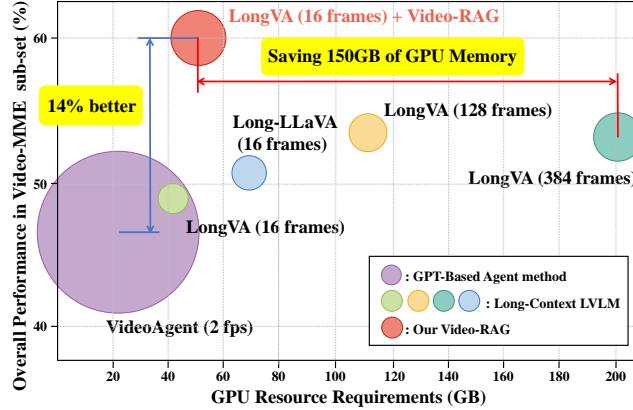


Figure 6: The comparison of our Video-RAG with two common approaches. The size of the bubbles represents the total time consumed for completing inference on the Video-MME [6] sub-set.

## D Details of Similarity Score Calculation

In the process of using the RAG system to retrieve auxiliary texts extracted from videos, we define a similarity threshold  $t$  to ensure the selection of relevant texts. Specifically, we employ FAISS-based [13] similarity to select OCR and ASR texts, while CLIP [30] similarity is used for keyframe selection. In our implementation, the similarity threshold  $t$  is set to 0.3. As for OCR and ASR selection, For any given list of the retrieve request  $\mathbf{R}$  and auxiliary texts  $\mathbf{A}$ , the Contriever [9] framework maps the text to a text embedding as:

$$\begin{aligned}\mathbf{E}_{a_i} &= \text{Contriever}(\mathbf{A}_i), \quad i = 1, 2, \dots, n \\ \mathbf{E}_{r_i} &= \text{Contriever}(\mathbf{R}_i), \quad i = 1, 2, \dots, n\end{aligned}$$

The average embedding of the retrieve request is then computed as:

$$\mathbf{E}_r = \frac{1}{n} \sum_{i=1}^n \mathbf{E}_{r_i}$$

After that, the embedding of the request and the list of auxiliary texts is normalized:

$$\mathbf{E}_{a_i} = \frac{\mathbf{E}_{a_i}}{\|\mathbf{E}_{a_i}\|}, \quad \mathbf{E}_r = \frac{\mathbf{E}_r}{\|\mathbf{E}_r\|}$$

The similarity between the query embedding  $\mathbf{E}_r$  and the document vector  $\mathbf{E}_a$  is computed using the inner product, the FAISS library is employed to efficiently perform this search and return the indices of the auxiliary texts meeting the criterion:

$$S(\mathbf{E}_r, \mathbf{E}_{a_i}) = \mathbf{E}_r \cdot \mathbf{E}_{a_i} > t$$

As for object detection, we use CLIP to select the video keyframe. During this process, we first filter the object detection request  $\mathbf{R}_{det}$  to ensure they correspond to CLIP-sensitive physical entities, avoiding the inclusion of abstract concepts. Specifically, if it is a single word, direct part-of-speech

filtering is applied; if it is a compound word, certain rules are followed to check for compliance, such as whether it is an adjective plus a noun, or a noun plus a noun. We use the Spacy library to achieve this. After this, we put the text “A picture of” before each object detection request.

Then, we extract embedding from both the video frames  $\mathbf{F}$  and the detection request  $\mathbf{R}_{det}$ :

$$\begin{aligned}\mathbf{E}_{\mathbf{F}_j} &= \text{CLIP}(\mathbf{F}_j), \quad j = 1, 2, \dots, m \\ \mathbf{E}_{\mathbf{R}_i} &= \text{CLIP}(\mathbf{R}_{det_i}), \quad i = 1, 2, \dots, n\end{aligned}$$

The similarity between each video frame and the detection retrieve requests is computed using the dot product between the image and text feature embeddings. For each frame  $\mathbf{F}_j$ , and for each retrieve request  $\mathbf{E}_{\mathbf{R}_i}$ , the similarity score is given by:

$$S_{ij} = \mathbf{E}_{\mathbf{F}_j} \cdot \mathbf{E}_{\mathbf{R}_i}$$

where  $\cdot$  denotes the dot product. The final similarity score for each frame is the average similarity across all requests:

$$S_j = \frac{1}{n} \sum_{i=1}^n S_{ij}$$

This computes the mean similarity for each frame across all text descriptions, resulting in a similarity vector  $\mathbf{S} = [S_1, S_2, \dots, S_m]$ . The similarity scores are adjusted by a scaling factor  $\alpha$ , which is computed based on the number of frames  $m$  and a base frame number  $b$  (which is set to 16 and 4.0, respectively) to adapt different video sampling rate of LVLMS:

$$\alpha = \beta \times \frac{m}{b}$$

where  $\beta$  is a predefined scaling parameter.

Next, the similarity scores are scaled and normalized to ensure that they sum to 1:

$$S_j^{\text{norm}} = \frac{\alpha \times S_j}{\sum_{k=1}^m S_k}$$

where  $S_j^{\text{norm}}$  represents the normalized similarity score for frame  $\mathbf{F}_j$ .

The final step is to select the keyframes based on the normalized similarity scores. A threshold  $t$  is applied to the normalized similarities, such that frames with similarity scores above the threshold are selected as keyframes:

$$\text{Keyframe: } \mathbf{F}_j \quad \text{if} \quad S_j^{\text{norm}} > t$$

Thus, the set of selected keyframes is given by:

$$\mathbf{F}_{key} = \{\mathbf{F}_j \mid S_j^{\text{norm}} > t, j = 1, 2, \dots, m\}$$

## E More Ablation Studies

**Effect of different components of Video-RAG.** We evaluate the performance across sub-tasks within Video-MME [6], as shown in Figure 7. The results reveal that object detection auxiliary texts significantly enhance spatial perception and object counting, while OCR auxiliary texts specifically improve performance on text recognition tasks. Additionally, ASR auxiliary texts contribute to a general improvement in inference tasks, underscoring the critical role of audio transcription in video understanding. Given that audio transcription is considerably more time-consuming than character recognition or object detection, these texts should be selected based on the requirements of the application.

Besides studying the inference of different components of Video-RAG in the Video-MME [6] benchmark, we also experiment with a different type of video benchmark. We first evaluate LLaVA-Video in MLVU [54] and LongVideoBench [43] in both 7B and 72B scale with the 64-frame setting, results are shown in Table 9. As demonstrated, when all components are combined, we get optimal

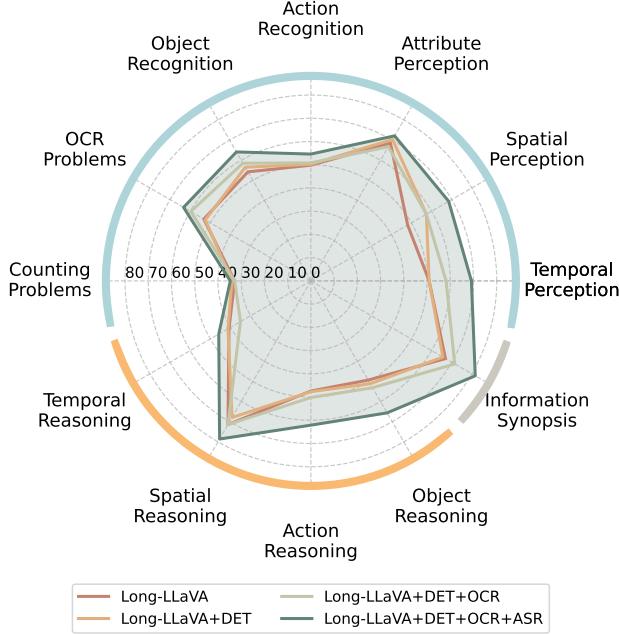


Figure 7: Performance on 12 sub-tasks in Video-MME [6] benchmark after applying different components in Long-LLaVA.

Table 9: Ablation study in MLVU and LongVideoBench.

RAG	DET	OCR	ASR	7B		72B	
				MLVU	LVB	MLVU	LVB
✓	✓			70.8	56.6	73.1	61.9
✓	✓	✓		71.0	56.5	73.4	63.2
✓	✓	✓	✓	71.3	56.8	73.5	63.4
<b>✓</b>				<b>72.4</b>	<b>58.7</b>	<b>73.8</b>	<b>65.4</b>
✓	✓	✓	✓	70.3	58.3	72.9	64.0

performance in both datasets, including 7B and 72B scales. Specifically, the performance in MLVU [54] even declined when the RAG system was not implemented.

Then, to better point out the role of DET and OCR, we evaluate Video-RAG in VNBench [53] with Long-LLaVA-7B [45]. VNBench is a synthetic benchmark designed to evaluate models’ long-context abilities, covering tasks such as retrieval, ordering, and counting. VNBench randomly inserts stickers or text into the video that has nothing to do with the original content of the video, thus typically challenging the model’s needle-in-the-haystack capability. As shown in Table 10, we find that applying DET and OCR as auxiliary texts can significantly improve the performance in retrieval, ordering, and counting tasks. However, the ASR component will decline the performance due to the subtitles are not ancillary to this particular task. These results demonstrated that our proposed distinct types of auxiliary texts can be selected according to the application needs to meet the requirements better.

## F More Qualitative Results

In this section, we show more results of LLaVA-Vdieu-7B when applying Video-RAG in different examples in Figure 9. The figure highlights several representative cases involving detailed video comprehension from Video-MME [6]. As illustrated, augmenting LLaVA-Video with external tools to process and retrieve auxiliary texts from videos significantly enhances its ability to reduce visual hallucinations, thereby enabling more accurate and confident responses to user queries.

### Decouple Prompt of the Multiple-choice Question

To answer the question step by step, list all the physical entities related to the question you want to retrieve, you can provide your retrieve request to assist you by the following JSON format:

```
{  
    "ASR": Optional[str]. The subtitles of the video that may relevant to the question you want to retrieve, in two sentences. If you no need for this information, please return null.  
    "DET": Optional[list]. (The output must include only physical entities, not abstract concepts, less than five entities) All the physical entities and their location related to the question you want to retrieve, not abstract concepts. If you no need for this information, please return null.  
    "TYPE": Optional[list]. (The output must be specified as null or a list containing only one or more of the following strings: 'location', 'number', 'relation'. No other values are valid for this field) The information you want to obtain about the detected objects. If you need the object location in the video frame, output "location"; if you need the number of specific object, output "number"; if you need the positional relationship between objects, output "relation".  
}  
## Example 1:  
Question: How many blue balloons are over the long table in the middle of the room at the end of this video? A. 1. B. 2. C. 3. D. 4.  
Your retrieve can be:  
{  
    "ASR": "The location and the color of balloons, the number of the blue balloons.",  
    "DET": ["blue balloons", "long table"],  
    "TYPE": ["relation", "number"]  
}  
## Example 2:  
Question: In the lower left corner of the video, what color is the woman wearing on the right side of the man in black clothes? A. Blue. B. White. C. Red. D. Yellow.  
Your retrieve can be:  
{  
    "ASR": null,  
    "DET": ["the man in black", "woman"],  
    "TYPE": ["location", "relation"]  
}  
## Example 3:  
Question: In which country is the comedy featured in the video recognized worldwide? A. China. B. UK. C. Germany. D. United States.  
Your retrieve can be:  
{  
    "ASR": "The country recognized worldwide for its comedy.",  
    "DET": null,  
    "TYPE": null  
}  
Note that you don't need to answer the question in this step, so you don't need any information about the video or image. You only need to provide your retrieve request (it's optional), and I will help you retrieve the information you want. Please provide the json format.
```

Figure 8: Decouple prompt of the multiple-choice question for LVLMS.

Table 10: Results on combinations of different auxiliary texts in VNBench [53] with 1-try setting when applying 7B Long-LLaVA [45] as LVLM under the 32-frames setting. **Ret**, **Ord**, and **Cnt** represent retrieval, ordering, and counting tasks, respectively.

RAG	DET	OCR	ASR	Ret	Ord	Cnt	Overall
✓	✓			65.1	25.6	24.2	38.3
✓	✓	✓		66.9	28.4	23.8	39.7
✓	✓	✓	✓	68.2	31.3	28.9	42.8
✓	✓	✓	✓	66.7	31.3	29.6	42.5

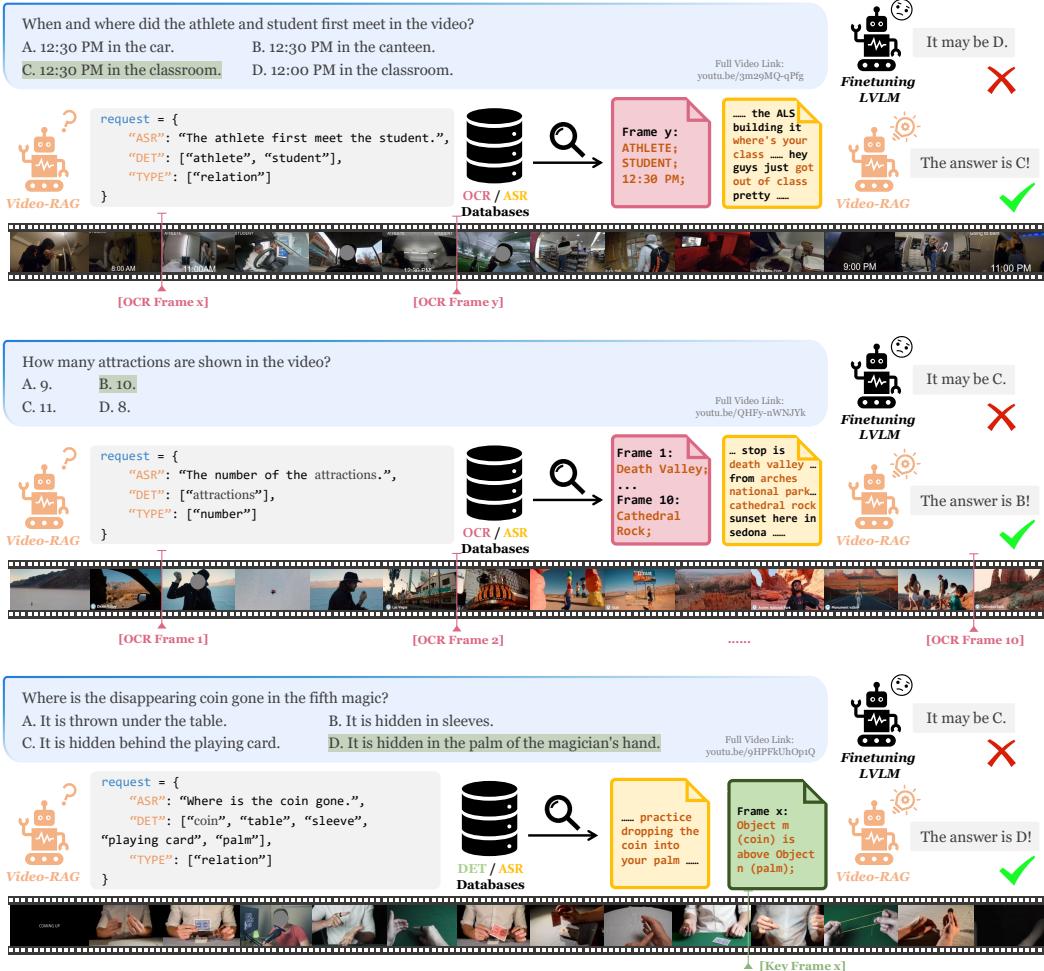


Figure 9: Qualitative results of LLaVA-Vvideo when applying Video-RAG.