VQAGuider: Guiding Multimodal Large Language Models to Answer Complex Video Questions

Yuyan Chen[1]*, **Jiyuan Jia**[2], **Jiaxin Lu**[3], **Siyue Li**[4], **Yu Guan**[5], **Ming Yang**[6], **Qingpei Guo**[6] ✉ †
[1]School of Computer Science, Fudan University,
[2]Southern University of Science and Technology, [3]Trine University,
[4]Northeastern University, [5]University of Warwick, [6]Ant Group
chenyuyan21@m.fudan.edu.cn, jiajy2018@mail.sustech.edu.cn,
jasonlu@vip.126.com, janelovelee2020@gmail.com,
yu.guan@warwick.ac.uk, {m.yang, qingpei.gqp}@antgroup.com

Abstract

Complex video question-answering (VQA) requires in-depth understanding of video contents including object and action recognition as well as video classification and summarization, which exhibits great potential in emerging applications in education and entertainment, etc. Multimodal large language models (MLLMs) may accomplish this task by grasping the intention of a question and decomposing it to a series of visual recognition sub-tasks to find out the answer with the help of an agent. To tackle this task, we first collect a new dedicated Complex VQA dataset named CVQA and then propose VQAGuider, an innovative framework planning a few atomic visual recognition tools by video-related API matching. VQAGuider facilitates a deep engagement with video content and precise responses to complex video-related questions by MLLMs, which is beyond aligning visual and language features for simple VQA tasks. Our experiments demonstrate VQAGuider is capable of navigating the complex VQA tasks by MLLMs and improves the accuracy by 29.6% and 17.2% on CVQA and the existing VQA datasets, respectively, highlighting its potential in advancing MLLMs's capabilities in video understanding.

1 Introduction

Video question-answering (VQA) allows users to explicitly express their interests in certain video contents, which exhibits great potential in many emerging applications. For example, in the education field, AI assistants that understand complex video contents can help students comprehend course material effectively (Peng et al., 2021; Chen et al., 2024b). In the field of entertainments, AI experts that well comprehend sports videos can help the audience understand the competition through analyzing the player's strategy in matches (Li et al., 2024b; Chen et al., 2024c). Complex VQA, as a high-level form of VQA, requires multimodal large language models (MLLMs) to understand the situational relationships within the video content, thereby presenting textual answers to the user's complex questions.

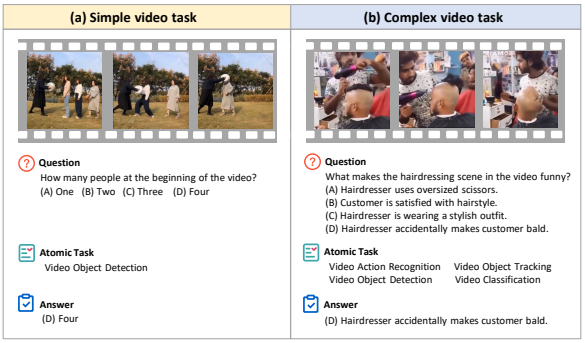

Figure 1: The examples of simple and complex VQA tasks, requiring one or multiple atomic vision tasks to figure out the answers, respectively.

MLLMs have demonstrated tremendous progress in image generation (Xu et al., 2024), image understanding (Sun et al., 2024), and video captioning (Zhou et al., 2024), etc. Nevertheless, their capabilities in VQA, especially complex VQA, have not been fully exploited. In real-world applications, most VQA tasks are complicated, involving planning and coordination of multiple basic visual tasks. For instance, as shown in Fig. 1 (b), answering the video question requires MLLMs to recognize the hair cut action and realize the customer head became bald by accident. This process involves basic visual tasks like video action recognition, video object tracking, and video object detection, etc. However, existing VQA benchmarks (Kim et al., 2024; Li et al., 2024c) do not provide a well-defined complex

* Work done during an internship at Ant Group.
† Qingpei Guo is the corresponding author.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 7821–7834

VQA dataset, making it hard to evaluate and enhance MLLMs' capabilities for complex VQA.

Existing methods for VQA mainly extract visual features and align them with the language ones (Choudhury et al., 2025; Amoroso et al., 2025; Chen et al., 2024a). These methods are effective for simple VQA tasks where questions can be answered directly by observing certain content in video frames. For example, as shown in Fig. 1 (a), only video object detection is able to answer this simple question about "How many people". In contrast, solving complex VQA tasks requires to plan a logical chain of atomic visual tasks like figuring out the objects in a video and tracking their motions, and then aggregates the results. This process can be well formulated as an agent who decomposes complex VQA to multiple atomic tasks and then consolidates MLLMs to produce an answer. In general, we summarize two main issues in tackling the complex VQA task: i) A lack of a suitable complex VQA dataset, making it difficult to measure and enhance the performance of MLLMs in handling such a complex task; ii) A lack of effective methodologies to guide MLLMs to figure out sensible answers in the complex VQA task.

To address the first issue, we define the complex VQA task as one requiring multiple atomic vision tasks to figure out the answer. Then we construct a complex VQA dataset, named CVQA, based on whether multiple atomic tasks are needed to derive the answer for this video question. To solve the second issue, we propose a novel framework, named VQAGuider [1], to guide MLLMs to identify atomic tasks, select appropriate VideoAPIs, and logically arrange the VideoAPI outputs, thereby enabling MLLMs to obtain answers for the complex video questions. To support VQAGuider in assisting MLLMs with answering complex video questions, we also require a set of specialized APIs that allow MLLMs to seek external help when they cannot rely on their internal knowledge. However, existing tools, such as ToolBench (Xu et al., 2023), APIBench (Patil et al., 2023), and API-Bank (Li et al., 2023c), mainly prioritize lower-level API libraries like Pandas in Python. These tools are insufficient for complex VQA tasks, as they do not facilitate the connection of real-world atomic tasks, such as object detection, with video content. Therefore, we design a series of video-related APIs (denoted as VideoAPIs) for each atomic task, linking the video content to these atomic tasks. We conduct experiments on the constructed CVQA and other VQA datasets, demonstrating the challenges of CVQA and the effectiveness of VQAGuider.

	Content		Task			VideoAPI							Path	
	Q	C	2	3	>3	VOD	VOT	VAR	VFD	VCl	VS	VCa	VK	S	P
L/C	18.2	20.1	1335	1187	1208	3267	2965	2874	1098	1342	1102	3034	3258	1796	1934
L/C (%)	/	/	35.8	31.8	32.4	87.6	79.5	77.1	29.4	36.0	29.5	81.3	87.3	48.2	51.8

Table 1: The statistics of CVQA and each module of it. "L/C": Token count in content items or count of items in atomic tasks, videoAPI distribution, and planned path distribution. "Q" and "C" represent questions and choices in CVQA. "VOD", "VOT", "VAR", "VFD", "VCl", "VS", "VCa", and "VK" represent VideoAPIs including VideoObjectDetection, VideoObjectTracking, VideoActionRecognition, VideoFaceDetection, VideoClassification, VideoSegmentation, VideoCaptioning, and VideoKeyFrameCapturing. "S" and "P" indicate sequential and parallel planning, respectively.

2 Dataset Construction

2.1 Video Atomic Tasks and VideoAPIs

We first propose eight atomic tasks which are given by human experts as the basic tasks required to complete a complex VQA task, including: *Video Object Detection*, which identifies a specific target in a video; *Video Object Tracking*, which identifies and tracks a target within a video sequence; *Video Action Recognition*, which recognizes human or equipment gestures in a video; *Video Face Detection*, which detects human faces in a video, including emotion analysis; *Video Classification*, which categorizes videos, such as movie genres, sports events, news, etc.; *Video Segmentation*, which divides a video frame into different parts based on scenes or objects; *Video Captioning*, which automatically generates text descriptions to explain the video content; *Video Summarization*, which extracts key frames of a video. Specifically, we consult three researchers skilled in VQA to rate the rationality of these atomic tasks on a scale of 1 to 5. They consistently rate these atomic tasks scores of 4 and above, which suggests the meaningfulness of these atomic tasks for answering complex video questions. We visualize these atomic tasks in Fig. 2.

Next, we link each atomic task to a corresponding VideoAPI. VideoAPI is defined as a function interface that can solve an atomic task related to a video. Specifically, VideoAPIs in this work are built based on LangChain [2]. We use GPT-4o in LangChain to extract relevant information, such as the position of an object in a frame of video, and also allow GPT-4o to interact with external APIs

[1] https://github.com/Yukyin/VQAGuider

[2] https://github.com/langchain-ai/langchain/tree/master

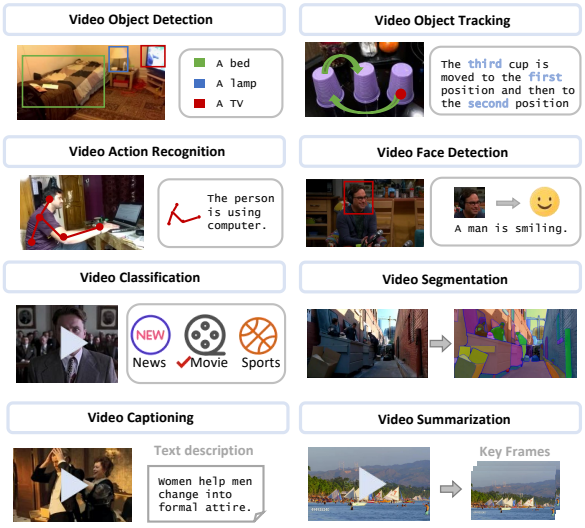

Figure 2: The illustrations of the atomic tasks.

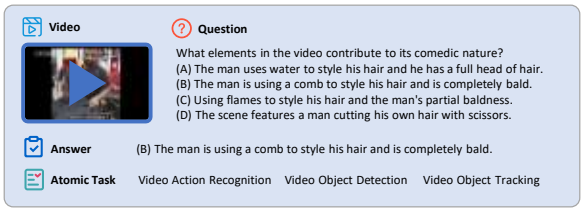

Figure 3: A sample case in CVQA with its atomic tasks.

like Pandas to solve a specific atomic task. For example, "VideoActionRecognition" is a VideoAPI of the video action recognition task, aiming to recognize human actions in the video. We input the video path into the VideoAPI "VideoActionRecognition" and obtain the output "Showing [something] next to [something] 0.72; Showing [something] behind [something] 0.13; Showing [something] on top of [something] 0.03", representing the probability of a series of actions in the video.

2.2 Complex VQA Dataset

After that, we construct a complex VQA dataset named CVQA. Compared with open-ended VQA dataset (Maaz et al., 2023), evaluating the performance of MLLMs is easier with multiple-choice VQA dataset due to the explicit answers. Therefore, we first collect multiple-choice VQA dataset from public sources such as MVBench (Li et al., 2024c), STAR (Wu et al., 2024), PAXION (Wang et al., 2024b), Moments in Time V1 (Monfort et al., 2019), FunQA (Xie et al., 2023), CLEVRER (Yi et al., 2019), Perception Test (Patraucean et al., 2024), Charades-STA (Gao et al., 2017), MoVQA (Zhang et al., 2023b), NTU RGB+D (Liu et al., 2019), VLN-CE (Krantz et al., 2020), and TVQA (Lei et al., 2018). Next, we input

VideoAPI	Input	Output	Explanation
VideoObjectDetection	File path of video	{Frame 20:["laptop", 0.84], Frame 40:["sofa", 0.93]...}	The name of specific objects in each frame.
VideoObjectTracking	File path of video	{Frame 0:["person", [114,453,248,927], 0.84], Frame 50:["oven", [0,469,98,901], 0.93]...}	Positions and movements of specific objects in each frame.
VideoActionRecognition	File path of video	{"Showing [something] next to [something]": 0.21, "Rolling [something] on a flat surface": 0.05, "Letting [something] roll along a flat surface": 0.04...}	Action in each frame.
VideoFaceDetection	File path of video	{"bounding_box": (100, 200, 50, 50), "id": "John_Doe", "confidence": 0.9, "emotion_type": happy}	The face and emotion of persons.
VideoClassification	File path of video	{"tai chi":0.69,"country line dancing":0.05,"dancing charleston":0.01...}	Category of the video.
VideoSegmentation	File path of video	{Frame 0:["trashbin":[14,23,42,66], 0.77], "street":[100,231,199,308], 0.65],... }	Different parts of each frame.
VideoCaptioning	File path of video	"{[00:00:00]": "(Music playing)","[00:00:05]": "Interviewer: Welcome to our show today. We have a special guest with us.", ...,"[00:00:50]": "(Music fades out)", "[00:00:55]": "Caption: The Importance of Renewable Energy in Urban Areas"}	Description of object, persons, actions and etc. in each frame.
VideoSummarization	File path of video	{Frame 20, Frame 50, Frame 80}	Keyframe of the video.

Table 2: The input, output and explanation of the constructed VideoAPIs.

a video and its question as well as the description of the complex VQA task into VideoChat2 (Li et al., 2023b), making it determine whether the current question is complex with the prompt "A complex video question consists of multiple atomic tasks including... determine if this question is a complex one. If so, output 1; otherwise, output 0." Afterwards, three video experts validate our classification with an average score of 4.2 out of 5, supporting its rationality.

To ensure a balanced comparison of VQA pairs containing varying numbers of atomic tasks, different types of video-related APIs, and diverse paths, we generate a series of complex video QA pairs with ChatGPT and the prompt "A complex video question consists of multiple atomic tasks including... Please generate a collection of complex video question based on the given video". Videos are sourced from the above-mentioned datasets. We make manual verification to ensure the reasonableness of the QA pairs and their alignment with the complexity definition. Only video QA pairs that receive a score of 4 or higher (on a scale of 1-5) from three video QA experts are considered to be of acceptable quality.

Ultimately, we have curated a complex VQA dataset named CVQA comprising 3,730 QA pairs, each associated with a corresponding video. Within this dataset, 2,325 QA pairs (62.3%) are derived from existing datasets. The average duration of each video is 32 seconds. The dataset statistics are summarized in Table 1, with a sample provided in Fig. 3. Table 3 presents the final proportions and the average size of atomic tasks covered for each

public source dataset. In order to ensure a relatively balanced distribution in terms of the number and types of atomic tasks as well as the proportion of planning paths, we adjust the proportion of each dataset accordingly. We also list statistics of commonly used open-sourced VQA datasets as shown in Table 4. It can be seen that the average size of atomic tasks covered and the average steps of planning path in CVQA dataset both exceed those of existing datasets, indicating there is a lack of complex VQA dataset.

3 Method

In this section, we propose a novel framework named VQAGuider for assisting MLLMs in solving the complex VQA task as shown in Fig. 4.

3.1 Atomic Task Recognition

The first step of VQAGuider is to guide MLLMs to identify the atomic tasks. In this step, we design certain prompts to help each MLLM choose appropriate atomic tasks from a list of candidates that can answer a complex question. First, we input the video, question, and candidate atomic tasks into the MLLM. The MLLM then decomposes the complex VQA task into a series of atomic tasks denoted as $\{x_1, x_2, ..., x_n\}$, forming an atomic task group denoted as X. For example, given the question "What is the action performed by the person in the video? (A) blowing (B) sailing (C) swimming (D) competing" shown in Fig. 4, the MLLM ultimately outputs the atomic task group $X = \{$Video Action Recognition, Video Object Tracking$\}$ for this question.

3.2 VideoAPI Matching

The second step is VideoAPI Matching with recognized atomic tasks, aiming at providing reasonable outputs for each atomic task for which MLLMs cannot complete due to insufficient coding capabilities. In this way, we bypass the step where MLLMs must directly perform video-based reasoning, thereby enhancing their performance in handling complex VQA tasks. Specifically, by querying the constructed VideoAPI library, we match the obtained atomic task group X with its corresponding VideoAPIs denoted as $\{a_1, a_2\}$, constructing the VideoAPI group denoted as A. For example, the atomic task group $X = \{$Video Action Recognition, Video Object Tracking$\}$ can be matched to obtain the VideoAPI group $A = \{$VideoActionRecognition, VideoObjectTracking$\}$

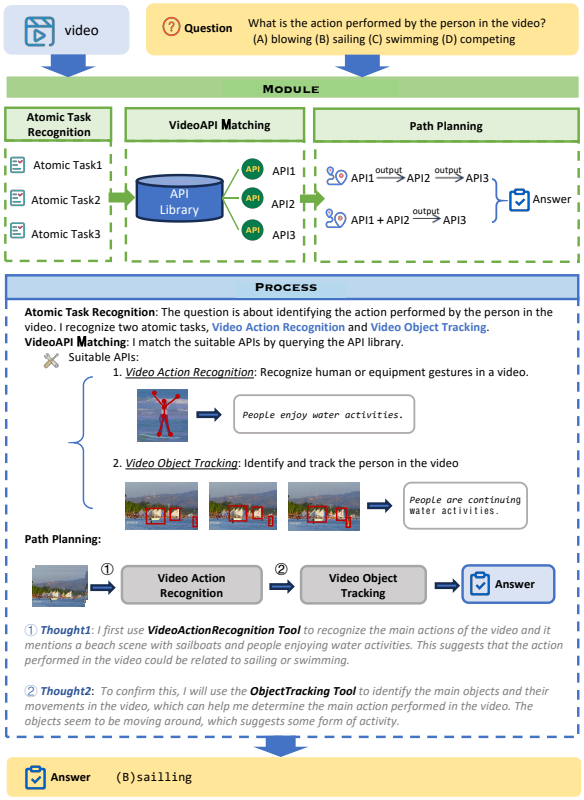

Figure 4: The detailed process of the using VQAGuider in assisting an MLLM to answer a complex video question.

in the VideoAPI library. The output of "VideoActionRecognition" is "People enjoy water activities on the water, conf: 0.85.", indicating the person's current action relates to water, and the output of "VideoObjectTracking" is "Frame id: 20 object id: 3, conf: 0.51, c: 0 position: (499, 278, 513, 289); Frame id: 20 object id: 2, conf: 0.52, c: 0 position: (84, 253, 96, 278);.....;Frame id: 60 object id: 5, conf: 0.68, c: 0 position: (525, 272, 553, 327)", suggesting that people are continuing water activities. The input and output of all VideoAPIs are shown in Table 2. After that, we adopt GPT-4o to convert the output of each VideoAPI in the VideoAPI group A into natural language for MLLMs to recognize with an instruction "Please interpret the output of each VideoAPI [CONTENT] with natural language".

3.3 Path Planning

The final step is to guide MLLMs to plan the transformed outputs of each VideoAPI for comprehensive reasoning. Generally, path planning includes two approaches for a question: sequential and parallel. In the sequential approach, the MLLM takes the transformed output of the first VideoAPI as the first thought, then uses the first thought as known information and obtains the second thought based

Dataset	MVBench	STAR	PAXION	Moments in Time V1	FunQA	CLEVRER	Perception Test	Charades-STA	MoVQA	NTU RGB+D	VLN-CE	TVQA	**CVQA**
Portion	39.2	2.4	1.1	7.2	2.6	9.2	7.3	3.5	1.4	1.3	6.7	18.1	100
AT	3.8	1.8	1.3	2.4	2.2	2.8	2.4	2.1	1.5	1.3	2.8	3.2	4.8

Table 3: The portions and the average size of atomic tasks covered for each public source dataset in CVQA. AT: Average size of atomic tasks.

Dataset	V	Q	L	AT	PP
MSVD-QA (Xu et al., 2017)	1,970	50K	10	2.4	1.2
MSRVTT-QA (Xu et al., 2016)	10,000	243K	15	2.5	1.1
ActivityNet-QA (Yu et al., 2019)	5,800	58K	180	2.8	1.5
NExT-QA (Xiao et al., 2021)	5,440	47K	44	3.1	1.9
AGQA (Grunde-McLaughlin et al., 2021)	9,615	192M	30	2.9	1.8
CVQA (Ours)	3,730	3,730	32	**4.8**	**3.1**

Table 4: Statistics of commonly used VQA datasets. V: Size of Videos. Q: Size of Questions. L: Average lengths of videos. AT: Average size of atomic tasks. PP: Average steps of planning paths.

on the transformed output of the second VideoAPI. Next, the second thought is used as known information to obtain the answer. The typical representation is as follows: Task 1's output $\rightarrow$ Task 2's output, Task 2's output $\rightarrow$ Task 3's output, Task 3's output $\rightarrow$ Task 4's output. In the parallel approach, both the first and second thought are used as known information to get the answer. The typical representation is as follows: Task 1's output $\rightarrow$ Task 2's output, (Task 1's output $\oplus$ Task 2's output) $\rightarrow$ Task 3's output, (Task 1's output $\oplus$ Task 2's output $\oplus$ Task 3's output) $\rightarrow$ Task 4's output, where "$\oplus$" denotes the integration of transformed outputs from multiple VideoAPIs. For the combination of typical sequential and typical parallel tasks, we also consider it as parallel, such as Task 1's output $\rightarrow$ Task 2's output, (Task 1's output $\oplus$ Task 2's output) $\rightarrow$ Task 3's output, Task 3's output $\rightarrow$ Task 4's output. For the previous question, MLLM's path planning goes as follows: First, it adopts the "VideoActionRecognition" VideoAPI with the output indicating actions of people engaging in water activities. Next, it undertakes the "VideoObjectTracking" VideoAPI, which indicates that people are always active in the water throughout the video. Combining these two atomic tasks in parallel, VQAGuider guides MLLMs to consider that people engage in water activities, and this action persists throughout the entire video, leading to the inference that the main action is likely sailing.

4 Experiment

4.1 Experimental Setups

We conduct our experiments on 8x Nvidia A100 GPUs, each with 80GB of memory, implemented using PyTorch in Python. For MLLMs capable of processing videos directly, we input the videos in their entirety without any frame selection strategy. For MLLMs that are unable to process videos directly, such as GPT-4v, we uniformly sample frames every 3 seconds, resulting in an average of 10 frames as input. We set the maximum sequence length for both input and output sequences to 1024 tokens. For path planning, we experiment with both sequential and parallel approaches for each MLLM, selecting the better option for demonstration.

4.2 Datasets, Baselines and Metrics

We select several MLLMs that have demonstrated strong performance on video tasks, including GPT-4o, LLaVA OneVision (Li et al., 2024a), Video-ChatGPT (Maaz et al., 2023), Video-LLaMA (Zhang et al., 2023a), PandaGPT (Su et al., 2023), Otter-I (Li et al., 2023a), VideoChat (Li et al., 2023b), VideoChat2 (Li et al., 2024c), mPLUG-Owl-I (Ye et al., 2023), Valley (Luo et al., 2023), GPT-4v (Yang et al., 2023), Chat-UniVi (Jin et al., 2023), MovieChat (Song et al., 2023b), and LLaMA-VID (Li et al., 2023d), as backbones to validate the effectiveness of VQAGuider. For comparing other powerful VQA methods with the proposed VQAGuider, we also choose COT (Wei et al., 2022), MoReVQA (Min et al., 2024), GSMT (Nguyen et al., 2024), VAA (Fan et al., 2024), FreeVA (Wu, 2024), and VAL (Wang et al., 2024a). COT is to remove the explicit annotation of atomic task types and API names in the prompts to further test whether MLLMs can achieve good performance solely through chain-of-thought reasoning. An MLM is requested to directly decompose the original question into atomic tasks, utilizing external tools to obtain outputs, and then reasoning over the outputs to generate answers. Besides validating the effect of VQAGuider in CVQA, we select open-sourced videoQA datasets as shown in Table 4 for further experiments. We follow the same test set as used in Min et al. (2024).

We use accuracy as the evaluation metric for the overall performance of VQAGuider. For each individual step like atomic task recognition, we measure the accuracy as the proportion of MLLM outputs that match the ground truth in CVQA. For videoAPI matching, we evaluate whether MLLMs correctly select the appropriate VideoAPIs and ex-

Model	CVQA				MSVD-QA				MSRVTT-QA				ActivityNet-QA				NExT-QA				AGQA			
	O	VG	↑	↑(%)	O	VG	↑	↑(%)	O	VG	↑	↑(%)	O	VG	↑	↑(%)	O	VG	↑	↑(%)	O	VG	↑	↑(%)
GPT-4o	61.5	63.7	2.2	3.6	83.9	86.2	2.3	2.7	66.0	70.7	4.7	7.1	63.1	66.3	3.2	5.1	69.4	71.4	2.0	2.9	72.9	73.8	0.9	1.2
LLaVA OneVision	58.1	62.2	4.1	7.1	79.5	83.7	4.2	5.3	64.6	68.5	3.9	6.0	59.4	63.5	4.1	6.9	67.4	70.2	2.8	4.2	66.3	69.5	3.2	4.8
VideoChat2	49.2	58.2	9.0	18.2	70	79.3	9.3	13.3	54.1	66.2	12.1	22.4	49.1	56.8	7.7	15.7	62.2	69.8	7.6	12.2	60.2	65.8	5.6	9.3
Panda-GPT	44.3	53.6	9.3	21.0	68.7	78.8	10.1	14.7	52.3	65.6	13.3	25.4	45.8	55.1	9.3	20.3	59.4	69.2	9.8	16.5	59.1	63.7	4.6	7.8
GPT4v	38.7	50.2	11.5	29.8	69.5	80.1	10.6	15.3	57.8	70.6	12.8	22.2	53.9	61.4	7.5	13.9	63.7	70.5	6.8	10.7	65.5	70.2	4.7	7.2
Chat-UniVi	35.3	46.6	11.2	31.8	65	77.6	12.6	19.4	54.6	62.7	8.1	14.8	45.8	54.7	8.9	19.4	61.5	68.8	7.3	11.9	53.9	59.8	5.9	11.0
Valley	34.7	48.2	13.5	38.9	60.5	69.5	9.0	14.9	51.1	63.6	12.5	24.5	45.1	52.8	7.7	17.1	57.7	66.3	8.6	14.9	58.4	62.4	4.0	6.9
LLaMA-VID	36.5	47.2	10.7	29.3	69.7	77.4	7.7	11.1	57.7	68.9	11.2	19.4	47.4	58.6	11.2	23.6	50.3	58.7	8.4	16.7	57.8	63.5	5.7	9.9
VideoChat	34.6	44.1	9.5	27.5	56.3	69.5	13.2	23.5	45	55.3	10.3	22.9	26.5	40.3	13.8	52.1	54.9	62.8	7.9	14.4	56.7	61.3	4.6	8.1
Video-LLaMA	31.1	44.2	13.1	41.9	51.6	65.2	13.6	26.4	29.6	40.7	11.1	37.5	12.4	28.4	16.0	129.0	47.6	53.2	5.6	11.8	52.7	58.7	6.0	11.4
MovieChat	30.6	42.0	11.4	37.4	75.2	84.1	8.9	11.8	52.7	61.6	8.9	16.9	45.7	54.1	8.4	18.4	55.7	62.7	7.0	12.6	55.1	59.3	4.2	7.6
Video-ChatGPT	29.8	38.5	8.7	29.3	64.9	77.3	12.4	19.1	49.3	56.8	7.5	15.2	35.2	46.5	11.3	32.1	58.6	64.6	6.0	10.2	59.2	65.5	6.3	10.6
mPLUG-Owl-I	24.8	33.2	8.4	33.9	57.3	65.6	8.3	14.5	37.6	50.2	12.6	33.5	29.6	37.1	7.5	25.3	51.3	60.0	8.7	17.0	54	59.5	5.5	10.2
Otter-I	21.4	35.2	13.9	64.9	56.9	65.4	8.5	14.9	37.9	48.6	10.7	28.2	25.7	35.5	9.8	38.1	52.8	61.2	8.4	15.9	51.5	55.7	4.2	8.2
Average	37.9	47.7	**9.8**	**29.6**	66.4	75.7	**9.3**	**14.8**	50.7	60.7	**10.0**	**21.2**	41.8	50.8	**9.0**	**29.8**	58.0	65.0	**6.9**	**12.3**	58.8	63.5	**4.7**	**8.2**

Table 5: The accuracies before (denoted as "O") and after (denoted as "VG") utilizing VQAGuider in CVQA, descriptive QA (i.e. MSVD-QA, MSRVTT-QA, ActivityNet-QA) and inferential QA (i.e. NExT-QA, AGQA) dataset. "VG": VQAGuider.

Model	MSVD-QA				MSRVTT-QA				ActivityNet-QA				NExT-QA				AGQA			
	O	VG	↑	↑(%)	O	VG	↑	↑(%)	O	VG	↑	↑(%)	O	VG	↑	↑(%)	O	VG	↑	↑(%)
Simple	68.5	**76.2**	7.7	11.2	53.7	**62.3**	8.6	16.0	47.9	**52.5**	4.6	9.6	62.7	**66.1**	3.4	5.4	63.8	**66.2**	2.4	3.8
Complex	64.2	**75.2**	11.0	17.1	47.8	**59.1**	11.3	23.7	35.6	**49.1**	13.5	37.8	53.4	**63.8**	10.4	19.6	53.8	**60.8**	6.9	12.9

Table 6: The accuracies before (denoted as "O") and after (denoted as "VG") utilizing VQAGuider in simple and complex VQA tasks.

	CVQA	MSVD-QA	MSRVTT-QA	ActivityNet-QA	NExT-QA	AGQA
O	37.9	66.4	50.7	41.8	58.0	58.8
COT	40.4	68.8	52.5	42.6	59.8	59.1
MoReVQA	43.8	-	-	-	63.1	-
GSMT	44.7	-	-	-	-	61.2
VAA	45.5	-	-	-	-	-
FreeVA	45.7	73.6	60.1	48.6	-	-
VAL	46.1	-	-	-	-	-
VQAGuider	**47.7**	**75.7**	**60.7**	**50.8**	**65.0**	**63.5**

Table 7: The accuracies before (denoted as "O") and after utilizing VQAGuider and other baseline methods in VQA datasets.

tract the results into readable natural language. The readability is assessed using GPT-4o on a 1-5 scale (1 being the worst and 5 being the best). For path planning, GPT-4o is also used to score the reasonableness of the designed path on a 1-5 scale.

4.3 Main Results

From Table 5, we observe that VQAGuider significantly enhances the complex VQA performance of MLLMs and the improvements are statistically significant with p value is less than 0.05 in the t-test (the same below). On average, VQAGuider leads to performance improvements of 33.7%, 16.6%, 23.6%, 33.8%, 13.7%, and 9.0% on CVQA, MSVD-QA, MSRVTT-QA, ActivityNet-QA, NExT-QA, and AGQA datasets, respectively, confirming the effectiveness of VQAGuider in boosting MLLMs' complex VQA capabilities. We observe that even for more powerful MLLMs, such as GPT-4o and LLaVA OneVision, VQAGuider also leads to a slight performance improvement. The possible reason for the great effect of VQAGuider lies in that it effectively decomposes a video question into smaller sub-tasks. Even though these powerful MLLMs possess sufficient capabilities, they may still make errors due to overlooking some details of certain video questions. We also compare other baseline methods with the proposed VQAGuider for each dataset in Table 7. The results also show that VQAGuider consistently improves the performance across all datasets when compared to the baseline methods.

Next, for the same dataset, different MLLMs demonstrate different improvement. For instance, on CVQA, Otter-I shows the most significant improvement, with an increase of 64.9%, while PandaGPT has the smallest improvement on this dataset, with only an 21.0% increase. We observe that the VQAGuider is more effective for more powerful MLLMs, and less useful for those needing instruction-following help, where Supervised Finetuning is needed. For a same MLLM, the degree of improvement varies across different datasets. For example, VideoChat2 exhibits the most significant improvement on CVQA, with a 18.2% increase, whereas on the NExT-QA dataset, it has the smallest improvement, with just a 12.2% increase. Moreover, we also find that the VQAGuider demonstrates a greater improvement in descriptive QA than in inferential QA. We consider the possible reason is that inferential QA requires not just atomic tasks but also deeper reasoning, and may necessitate the integration of world knowledge and commonsense to be effectively addressed.

After that, we present MLLMs' average metrics for respective simple and complex questions on five datasets in Table 6. It shows that VQAGuider improves MLLMs' performance in both questions, with a greater impact on complex ones. We also analyze the performance of different steps in VQAGuider as shown in Table 8 (Column 2-

Model	AT	API	P	w/o AT	w/o API	w/o P	V	P_s	P_p
GPT-4o	72.3	4.4	4.5	60.0	56.7	61.3	62.5	61.9	62.3
LLaVA OneVision	70.2	4.2	4.5	58.7	54.8	59.2	61.7	61.0	61.8
VideoChat2	67.2	3.9	4.4	55.1	48.3	55.3	56.9	56.8	57.4
Panda-GPT	63.1	3.4	4.1	50.2	45.5	52.2	52.8	51.5	52.5
GPT4v	64.5	3.8	4.2	46.8	42.7	48.5	50.3	49.3	50.7
Chat-UniVi	56.3	3.4	3.6	43.3	38.1	44.1	45.5	44.2	45.1
Valley	60.1	3.6	3.9	45.1	42.6	47.9	47.3	47.4	47.9
LLaMA-VID	63.6	3.9	3.8	40.3	38.7	45.6	46.4	45.5	46.7
VideoChat	56.2	3.6	3.7	41.4	37.7	42.9	43.2	42.7	43.4
Video-LLaMA	54.3	3.8	3.8	43.6	36.8	43.7	43.9	43.2	44.1
MovieChat	47.4	3.2	3.6	38.1	36.3	40.5	40.0	39.6	40.5
Video-ChatGPT	50.5	3.4	3.8	35.8	30.1	36.9	37.1	36.4	38.2
mPLUG-Owl-I	46.7	3.2	3.6	28.5	25.5	33.1	32.2	32.8	33.1
Otter-I	49.3	3.2	3.5	29.3	24.8	33.5	32.8	36.7	33.8
Average	58.7	3.6	3.9	44.0	39.9	46.1	46.6	46.4	47.0
↓	-	-	-	**3.6**	**7.8**	**1.6**	**1.0**	**1.3**	**0.7**
↓(%)	-	-	-	**8.3**	**19.4**	**3.5**	**2.2**	**2.8**	**1.5**

Table 8: The accuracies of MLLMs in each step of the proposed VQAGuider and the effect of each step in CVQA for various MLLMs. "AT", "API", and "P" represent performance of atomic task recognition, VideoAPI matching, path planning, respectively.

4). For atomic task recognition (denoted as AT), GPT-4o performs the best, which is above the average level of 58.7. mPLUG-Owl-I shows weaker performance in this step. For VideoAPI matching (denoted as API), all MLLMs score closely, with an average of 3.6, indicating significant room for improvement in the program language understanding of VideoAPI outputs. For path planning (denoted as P), GPT-4o and LLaVA OneVision both perform the best, showing their advantage in planning effective answer paths. However, most MLLMs score between 3.6 and 3.9, suggesting that path planning is not the main factor constraining the complex VQA performance of MLLMs.

4.4 Ablation Study

We first observe that each step of VQAGuider contributes significantly to the overall performance as shown in Table 8 (Column 5-10). When the atomic task recognition step (denoted as w/o AT) is removed, there is an average performance decrease of 8.3% across all MLLMs, which possibly attribute the atomic task recognition to the role that helps MLLMs locate key information in the video more accurately. The removal of the VideoAPI matching step (denoted as w/o API) results in the largest average performance drop of 19.4% among three steps. The reason might be that VideoAPIs provide more professional or detailed data processing outcomes, which a single MLLM struggle to perform independently. Eliminating the path planning step (denoted as w/o P) leads to an average performance decline of 3.5%. Although this decrease is relatively small, it still highlights path planning is important in integrating different information sources into a final answer. We also experiment with replace planning paths to only sequential ones (denoted as P_s) or parallel ones (denoted as P_p) processing for all questions. We find that the results deteriorate in both cases, with a more significant decline when using only sequential processing compared with using VQAGuider. This indicates that most complex video tasks require the simultaneous handling and integration of multiple atomic tasks. Additionally, during the construction of the VideoAPIs, we replace LangChain to another powerful code generation tool ViperGPT (Surís et al., 2023) (denoted as V) and observe that ViperGPT's performance is slightly inferior. The possible reason is that ViperGPT focuses on the image domain, so its capability to generate code for atomic tasks related to videos still requires improvement.

Next, we study the average performance of the cases with different numbers of atomic tasks with VQAGuider as shown in Fig. 5. We can observe that the number of atomic tasks significantly negatively affects MLLMs' performance. When a question contains two atomic tasks, the accuracy (denoted as Acc) on CVQA is the highest among the three scenarios. After that, the performance consistently declines when the number of atomic tasks increases to three and exceeds three, suggesting that as the number of atomic tasks increases, the processing becomes more complex, and VQAGuider requires more optimization. After that, we explore the average performance of the cases involving different atomic tasks before and after using VQAGuider, respectively, as shown in Fig. 6. It can be observed that the performance of each API declines without the use of VQAGuider, demonstrating the effectiveness of VQAGuider in VideoAPIs matching. With the help of VQAGuider, video captioning and action recognition in videos shows good performance. However, video classification and video segmentation has the lowest accuracy, possibly due to the diversity and complexity of video content. Future research could explore how to optimize MLLMs' performance with VQAGuider on these atomic tasks.

4.5 Case Study

In this section, we present a case processed by VQAGuider, as shown in Fig. 7. First, VQAGuider guides an MLLM, such as VideoChat2, to recognize atomic tasks and acquires a group of atomic tasks "Video summarization, Video object tracking", followed by VideoAPI Matching to obtain a group of VideoAPIs "VideoSummarization,

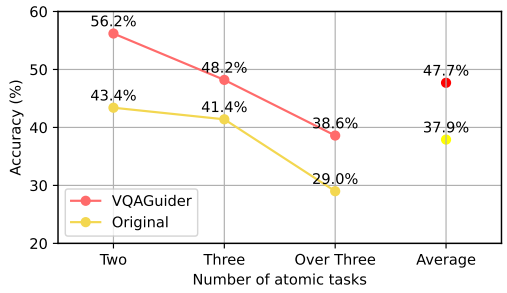

Figure 5: The average performance of the cases with different numbers of atomic tasks after using VQAGuider.

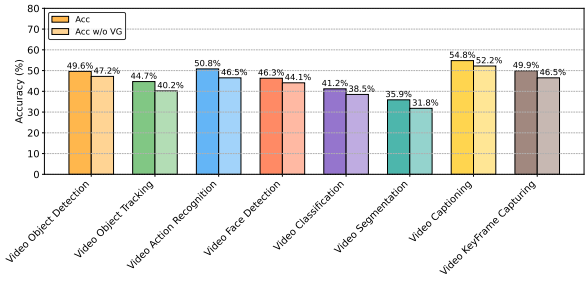

Figure 6: The average performance of the cases with different VideoAPI for MLLMs before and after using VQAGuider, respectively. VG: VQAGuider.

VideoObjectTracking". The program language output by the VideoAPI library is "Frame 260, 280, 480" and "Frame id: 260, object id: 2, ...", respectively. Afterwards, it guides the MLLM to transform the program language into natural language as "A person is using a laptop" and"A laptop on the leg; ...". Next, for the path planning, the MLLM establishes a sequential relationship for the transformed output as "The person is using a laptop, and he throws a pillow from the couch and continues using his laptop.". Hence, the MLLM is guided to output the correct answer "(B) Put down the laptop."

However, VQAGuider also encounters some errors, such as the non-applicable instructions to some MLLMs, resulting in some samples where MLLMs cannot output atomic tasks, or convert the VideoAPI's output into natural language, or output the exact answer after path planning (the MLLM may output an entire paragraph instead of the answer itself). In the future, we will improve VQAGuider's instruction adaptability to better complete the guidance for MLLMs in the complex VQA task.

4.6 Exploratory Analysis

We also conduct an exploratory analysis to evaluate the importance of video grounding and whether the introduced atomic tasks capture the complexity of videos in the vertical domain. For the first aspect, we find that some questions can be answered effectively through language understanding alone without video grounding. We test 100 questions with MLLMs, achieving an average accuracy of 16.6% without video input, compared to 34.6% with video. Notably, 11.0% of the questions are correctly answered by more than half of the MLLMs without video context. This indicates that determining when video grounding is necessary for answering video questions is an important research area.

For the second aspect, we choose five domains, including finance, agriculture, technology, sports, and healthcare with 20 short TikTok videos per category, generating five complex questions for each video, resulting in 500 questions. We then validate the complexity by having three human evaluators determine whether answering these questions required the predefined atomic tasks. The results show that 477 out of 500 questions (95.4%) rely on these tasks, suggesting their adequacy for most cases. However, for the remaining 4.6%, VQAGuider needs a reasoning module that builds on atomic task outputs, rather than relying solely on path planning, which is a future optimization.

Moreover, we provide an analysis of VQAGuider's efficiency and computational cost. While the original MLLMs require only a single prompt to generate an output, VQAGuider involves three prompts to produce the final result. On average, the original MLLMs take about 8 seconds, whereas VQAGuider takes approximately 20 seconds. Despite the increased time, there is no significant difference in computational resource consumption, demonstrating the its high efficiency alongside its improved performance. In future work, we plan to further optimize VQAGuider to enhance efficiency.

5 Related Work

VQA. Choudhury et al. (2025) introduced a procedural program-based approach for video question answering. Amoroso et al. (2025) introduced a temporal querying transformer that extracting relevant information over time and space in the VQA task. Yu et al. (2024) proposed CREMA for multimodal compositional video reasoning; Liang et al. (2024) presented a reference-free method for evaluating video QA and caption data quality. Others focused on constructing benchmark of VQA. For example, Han et al. (2023) intro-

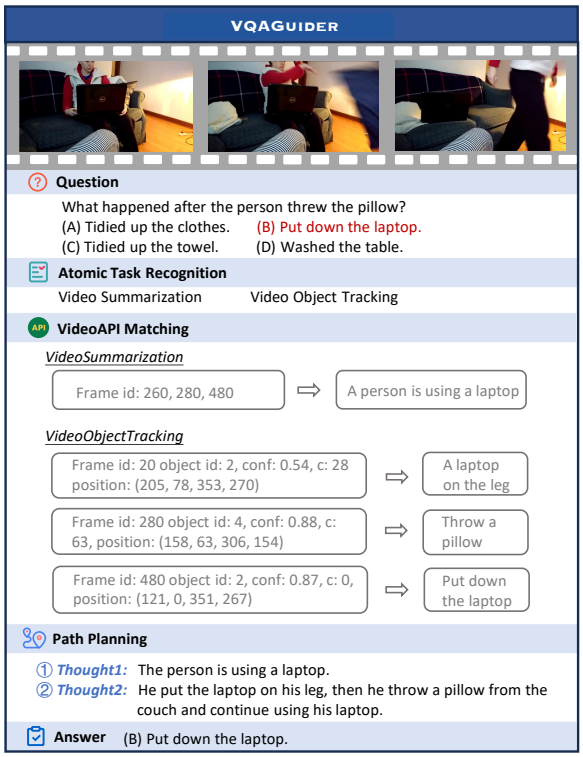

Figure 7: The average performance of the cases with different VideoAPI after using VQAGuider.

duced Shot2Story20K, a multi-shot video benchmark with comprehensive summaries; However, the existing benchmark do not contain a complex VQA task.

Agent. Wei et al. (2025) proposed a multi-agent multimodal framework to perform advanced understanding and indexing of presentation-style videos. Ma et al. (2024) introduced an MLLM-based autonomous agent for comprehensive cognition and reliable perception. Yue et al. (2024) fine-tuned a general-purpose MLLM to enhance multimodal retrieval for embodied agents in unseen scenarios. Gu et al. (2024) demonstrated how large-scale deployments of MLLM agents can lead to exponential jailbreaks. Zheng et al. (2024) introduced a multimodal large language agent framework for closed-loop vehicle motion planning. Shen et al. (2023) proposed HuggingGPT, an LLM-powered agent that leverages ChatGPT in solving intricate tasks; Song et al. (2023a) adopted LLMs as planners for embodied agents. Inspired by the various applications of agents, we proposed a novel VQAGuider for enhancing MLLMs' complex VQA capability. Although Wang et al. (2025) and Fan et al. (2025) also address video QA, they focus on improving video understanding rather than solving complex VQA tasks.

Tool usage. Wu et al. (2025) introduced a new prompting paradigm that enhances the detection ability of MLLMs through strategic tool combinations; Chen et al. (2025b) presented a DPD strategy for reducing hallucinations for QA dataset; Chen et al. (2025a) developed MedTransTab for cross-table tabular data generation in the medical context; Patil et al. (2023) introduced Gorilla, a finetuned LLaMA-based model that outperforms GPT-4o in writing API calls; Qin et al. (2023) collected real-world APIs, prompting ChatGPT to generate diverse human commands involving these APIs; Schick et al. (2023) introduced Toolformer, deciding when and how to use external tools to enhance LLMs' performance. Although some work construct tool-related benchmark (Patil et al., 2023; Li et al., 2023c; Xu et al., 2023), there is a lack of tools related to videos, especially complex VQA.

6 Conclusions and Future Work

The study of complex VQA offers insights into the application of artificial intelligence technologies in areas such as education, entertainment, and security monitoring. In this research, we first define the complex VQA task and construct the corresponding dataset named CVQA, aiming to enhance the performance of MLLMs in handling such task. Next, we introduce VQAGuider to effectively guide MLLMs in solving CVQA through atomic task recognition, VideoAPI matching, and path planning. Experiments show that our proposed VQAGuider is effective in CVQA and other VQA datasets. Future work will focus on optimizing the performance of VQAGuider and expanding its application scope. We will also work on enriching CVQA to cover more types of complex video-related scenarios.

Limitations

Although our VQAGuider shows significant effectiveness in handling the complex VQA task, our study still has several limitations. For example, CVQA's scale and diversity need further expansion. The current dataset may not fully cover all types of complex VQA scenarios. Moreover, how to further optimize the MLLMs' performance and broaden its application scope in more extensive video content understanding and analysis tasks remains an important direction for our future research.

Acknowledgements

This work is supported by Ant Group Research Intern Program.

References

Roberto Amoroso, Gengyuan Zhang, Rajat Koner, Lorenzo Baraldi, Rita Cucchiara, Volker Tresp, et al. 2025. Perceive, query & reason: Enhancing video qa with question-guided temporal queries. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision 2025*.

Yuyan Chen, Qingpei Guo, Shuangjie You, and Zhixu Li. 2025a. Medtranstab: Advancing medical cross-table tabular data generation. In *Proceedings of the 18th ACM International Conference on Web Search and Data Mining*.

Yuyan Chen, Zehao Li, Shuangjie You, Zhengyu Chen, Jingwen Chang, Yi Zhang, Weinan Dai, Qingpei Guo, and Yanghua Xiao. 2025b. Attributive reasoning for hallucination diagnosis of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Yuyan Chen, Songzhou Yan, Qingpei Guo, Jiyuan Jia, Zhixu Li, and Yanghua Xiao. 2024a. Hotvcom: Generating buzzworthy comments for videos. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

Yuyan Chen, Songzhou Yan, Panjun Liu, and Yanghua Xiao. 2024b. Dr.academy: A benchmark for evaluating questioning capability in education for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

Yuyan Chen, Songzhou Yan, Zhihong Zhu, Zhixu Li, and Yanghua Xiao. 2024c. Xmecap: Meme caption generation with sub-image adaptability. In *Proceedings of the 32nd ACM Multimedia*.

Rohan Choudhury, Koichiro Niinuma, Kris M Kitani, and László A Jeni. 2025. Video question answering with procedural programs. In *European Conference on Computer Vision*, pages 315–332. Springer.

Yue Fan, Xiaojian Ma, Rujie Wu, Yuntao Du, Jiaqi Li, Zhi Gao, and Qing Li. 2024. Videoagent: A memory-augmented multimodal agent for video understanding. In *European Conference on Computer Vision*, pages 75–92. Springer.

Yue Fan, Xiaojian Ma, Rujie Wu, Yuntao Du, Jiaqi Li, Zhi Gao, and Qing Li. 2025. Videoagent: A memory-augmented multimodal agent for video understanding. In *European Conference on Computer Vision*, pages 75–92. Springer.

Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. 2017. Tall: Temporal activity localization via language query. In *Proceedings of the IEEE international conference on computer vision*, pages 5267–5275.

Madeleine Grunde-McLaughlin, Ranjay Krishna, and Maneesh Agrawala. 2021. Agqa: A benchmark for compositional spatio-temporal reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11287–11297.

Xiangming Gu, Xiaosen Zheng, Tianyu Pang, Chao Du, Qian Liu, Ye Wang, Jing Jiang, and Min Lin. 2024. Agent smith: A single image can jailbreak one million multimodal llm agents exponentially fast. *arXiv preprint arXiv:2402.08567*.

Mingfei Han, Xiaojun Chang, Heng Wang, and Linjie Yang. 2023. Shot2story20k: A new benchmark for comprehensive understanding of multi-shot videos. *arXiv preprint arXiv:2312.10300*.

Peng Jin, Ryuichi Takanobu, Caiwan Zhang, Xiaochun Cao, and Li Yuan. 2023. Chat-univi: Unified visual representation empowers large language models with image and video understanding. *arXiv preprint arXiv:2311.08046*.

Wonkyun Kim, Changin Choi, Wonseok Lee, and Wonjong Rhee. 2024. An image grid can be worth a video: Zero-shot video question answering using a vlm. *arXiv preprint arXiv:2403.18406*.

Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. 2020. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*, pages 104–120. Springer.

Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L Berg. 2018. Tvqa: Localized, compositional video question answering. *arXiv preprint arXiv:1809.01696*.

Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Jingkang Yang, and Ziwei Liu. 2023a. Otter: A multi-modal model with in-context instruction tuning. *arXiv preprint arXiv:2305.03726*.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. 2024a. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*.

Haopeng Li, Andong Deng, Qiuhong Ke, Jun Liu, Hossein Rahmani, Yulan Guo, Bernt Schiele, and Chen Chen. 2024b. Sports-qa: A large-scale video question answering benchmark for complex and professional sports. *arXiv preprint arXiv:2401.01505*.

KunChang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. 2023b. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*.

Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, et al. 2024c. Mvbench: A comprehensive multi-modal video understanding benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22195–22206.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023c. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116.

Yanwei Li, Chengyao Wang, and Jiaya Jia. 2023d. Llama-vid: An image is worth 2 tokens in large language models. *arXiv preprint arXiv:2311.17043*.

Hao Liang, Zirong Chen, and Wentao Zhang. 2024. Evqascore: Efficient video question answering data evaluation. *arXiv preprint arXiv:2411.06908*.

Jun Liu, Amir Shahroudy, Mauricio Perez, Gang Wang, Ling-Yu Duan, and Alex C Kot. 2019. Ntu rgb+ d 120: A large-scale benchmark for 3d human activity understanding. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2684–2701.

Ruipu Luo, Ziwang Zhao, Min Yang, Junwei Dong, Minghui Qiu, Pengcheng Lu, Tao Wang, and Zhongyu Wei. 2023. Valley: Video assistant with large language model enhanced ability. *arXiv preprint arXiv:2306.07207*.

Xinbei Ma, Zhuosheng Zhang, and Hai Zhao. 2024. Coco-agent: A comprehensive cognitive mllm agent for smartphone gui automation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 9097–9110.

Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. 2023. Video-chatgpt: Towards detailed video understanding via large vision and language models. *arXiv preprint arXiv:2306.05424*.

Juhong Min, Shyamal Buch, Arsha Nagrani, Minsu Cho, and Cordelia Schmid. 2024. Morevqa: Exploring modular reasoning models for video question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13235–13245.

Mathew Monfort, Alex Andonian, Bolei Zhou, Kandan Ramakrishnan, Sarah Adel Bargal, Tom Yan, Lisa Brown, Quanfu Fan, Dan Gutfreund, Carl Vondrick, et al. 2019. Moments in time dataset: one million videos for event understanding. *IEEE transactions on pattern analysis and machine intelligence*, 42(2):502–508.

Thong Thanh Nguyen, Zhiyuan Hu, Xiaobao Wu, Cong-Duy T Nguyen, See-Kiong Ng, and Anh Tuan Luu. 2024. Encoding and controlling global semantics for long-form video question answering. *arXiv preprint arXiv:2405.19723*.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.

Viorica Patraucean, Lucas Smaira, Ankush Gupta, Adria Recasens, Larisa Markeeva, Dylan Banarse, Skanda Koppula, Mateusz Malinowski, Yi Yang, Carl Doersch, et al. 2024. Perception test: A diagnostic benchmark for multimodal video models. *Advances in Neural Information Processing Systems*, 36.

Min Peng, Chongyang Wang, Yuan Gao, Yu Shi, and Xiang-Dong Zhou. 2021. Temporal pyramid transformer with multimodal interaction for video question answering. *arXiv preprint arXiv:2109.04735*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2023a. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009.

Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Xun Guo, Tian Ye, Yan Lu, Jenq-Neng Hwang, et al. 2023b. Moviechat: From dense token to sparse memory for long video understanding. *arXiv preprint arXiv:2307.16449*.

Yixuan Su, Tian Lan, Huayang Li, Jialu Xu, Yan Wang, and Deng Cai. 2023. Pandagpt: One model to instruction-follow them all. *arXiv preprint arXiv:2305.16355*.

Keqiang Sun, Junting Pan, Yuying Ge, Hao Li, Haodong Duan, Xiaoshi Wu, Renrui Zhang, Aojun Zhou, Zipeng Qin, Yi Wang, et al. 2024. Journeydb: A benchmark for generative image understanding. *Advances in Neural Information Processing Systems*, 36.

Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. Vipergpt: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11888–11898.

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

Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. 2025. Videoagent: Long-form video understanding with large language model as agent. In *European Conference on Computer Vision*, pages 58–76. Springer.

Zhenhailong Wang, Ansel Blume, Sha Li, Genglin Liu, Jaemin Cho, Zineng Tang, Mohit Bansal, and Heng Ji. 2024b. Paxion: Patching action knowledge in video-language foundation models. *Advances in Neural Information Processing Systems*, 36.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Kangda Wei, Zhengyu Zhou, Bingqing Wang, Jun Araki, Lukas Lange, Ruihong Huang, and Zhe Feng. 2025. Premind: Multi-agent video understanding for advanced indexing of presentation-style videos. *arXiv preprint arXiv:2503.00162*.

Bo Wu, Shoubin Yu, Zhenfang Chen, Joshua B Tenenbaum, and Chuang Gan. 2024. Star: A benchmark for situated reasoning in real-world videos. *arXiv preprint arXiv:2405.09711*.

Wenhao Wu. 2024. Freeva: Offline mllm as training-free video assistant. *arXiv preprint arXiv:2405.07798*.

Yixuan Wu, Yizhou Wang, Shixiang Tang, Wenhao Wu, Tong He, Wanli Ouyang, Philip Torr, and Jian Wu. 2025. Dettoolchain: A new prompting paradigm to unleash detection ability of mllm. In *European Conference on Computer Vision*, pages 164–182. Springer.

Junbin Xiao, Xindi Shang, Angela Yao, and Tat-Seng Chua. 2021. Next-qa: Next phase of question-answering to explaining temporal actions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9777–9786.

Binzhu Xie, Sicheng Zhang, Zitang Zhou, Bo Li, Yuanhan Zhang, Jack Hessel, Jingkang Yang, and Ziwei Liu. 2023. Funqa: Towards surprising video comprehension. *arXiv preprint arXiv:2306.14899*.

Dejing Xu, Zhou Zhao, Jun Xiao, Fei Wu, Hanwang Zhang, Xiangnan He, and Yueting Zhuang. 2017. Video question answering via gradually refined attention over appearance and motion. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1645–1653.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. 2024. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36.

Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5288–5296.

Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. 2023. On the tool manipulation capability of open-source large language models. *arXiv preprint arXiv:2305.16504*.

Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. 2023. The dawn of lmms: Preliminary explorations with gpt-4v (ision). *arXiv preprint arXiv:2309.17421*, 9(1):1.

Qinghao Ye, Haiyang Xu, Guohai Xu, Jiabo Ye, Ming Yan, Yiyang Zhou, Junyang Wang, Anwen Hu, Pengcheng Shi, Yaya Shi, et al. 2023. mplug-owl: Modularization empowers large language models with multimodality. *arXiv preprint arXiv:2304.14178*.

Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B Tenenbaum. 2019. Clevrer: Collision events for video representation and reasoning. *arXiv preprint arXiv:1910.01442*.

Shoubin Yu, Jaehong Yoon, and Mohit Bansal. 2024. Crema: Multimodal compositional video reasoning via efficient modular adaptation and fusion. *arXiv preprint arXiv:2402.05889*.

Zhou Yu, Dejing Xu, Jun Yu, Ting Yu, Zhou Zhao, Yueting Zhuang, and Dacheng Tao. 2019. Activitynet-qa: A dataset for understanding complex web videos via question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9127–9134.

Junpeng Yue, Xinru Xu, Börje F Karlsson, and Zongqing Lu. 2024. Mllm as retriever: Interactively learning multimodal retrieval for embodied agents. *arXiv preprint arXiv:2410.03450*.

Hang Zhang, Xin Li, and Lidong Bing. 2023a. Video-llama: An instruction-tuned audio-visual language model for video understanding. *arXiv preprint arXiv:2306.02858*.

Hongjie Zhang, Yi Liu, Lu Dong, Yifei Huang, Zhen-Hua Ling, Yali Wang, Limin Wang, and Yu Qiao. 2023b. Movqa: A benchmark of versatile question-answering for long-form movie understanding. *arXiv preprint arXiv:2312.04817*.

Yupeng Zheng, Zebin Xing, Qichao Zhang, Bu Jin, Pengfei Li, Yuhang Zheng, Zhongpu Xia, Kun Zhan, Xianpeng Lang, Yaran Chen, et al. 2024. Planagent: A multi-modal large language agent for closed-loop vehicle motion planning. *arXiv preprint arXiv:2406.01587*.

Xingyi Zhou, Anurag Arnab, Shyamal Buch, Shen Yan, Austin Myers, Xuehan Xiong, Arsha Nagrani, and Cordelia Schmid. 2024. Streaming dense video captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18243–18252.

Atomic Task Recognition	You are going to decompose a complex question step by step about video into following atomic tasks, including: Video Classification: Classifying videos into categories such as movie genres, sports events, news, etc; Object Detection: Identifying specific objects in videos; Object Tracking: Identifying and tracking specific objects in videos, typically involving the analysis of object positions and movements; Action Recognition: Recognizing actions in videos, understanding the behavior of people or objects in the video; Video Summarization: Extracting key frames or segments from videos to create a summary and reduce the video's duration; Video Segmentation: Dividing videos into different parts, usually based on scenes or objects; Face Detection: Detecting and recognizing faces in videos, which may include emotion analysis; Video Captioning: Automatically generating textual descriptions for videos, explaining the content. You need to first choose more than one of the following atomic tasks to solve the problem.
VideoAPI Matching	The API tools contains: VideoClassification, VideoObjectDetection, VideoObjectTracking, VideoActionRecognition, VideoSummarization, VideoSegmentation, VideoFaceDetection, VideoCaptioning. You need to use the above-mentioned corresponding tools based on the chosen atomic tasks to solve the problem and transform the outputs of each tool into a readable sentence.
Path Planning	There are two possible ways for path planning with the information: sequential and parallel. You need to design a possible path with all outputs of API tools to get the final answer for the question.

Table 9: The prompts for each step of the proposed VQAGuider.