HawkEye: Training Video-Text LLMs for Temporal Grounding with Verbal Referencing

Anonymous ACL submission

Abstract

Video-text large language models (video-text LLMs) have shown remarkable performance in answering questions and holding conversations on videos. However, without targeted training, they perform almost the same as random on time-sensitive tasks like temporal grounding, as these models have not learned to use numbers to represent the start and end timestamps of video segments. In this paper, we investigate using a verbal reference method, such as "at the beginning" or "in the end," as alternatives to timestamps for referencing video segments. We demonstrate that video-text LLMs, even those not trained on video-segment level annotations, possess a substantial capability to perform temporal video grounding tasks with the proposed verbal reference method. To further demonstrate its efficacy and robustness, we propose HawkEye, a video-text LLM that has not only state-of-the-art performance on zero-shot temporal grounding, but also comparable performance with existing video-text LLMs across a spectrum of other video-text tasks. To train HawkEye, we propose InternVid-G, a large-scale video-text corpus with segment-level annotations for temporal grounding training. We also explore some practical training techniques such as mining grounding context spans from whole videos, and data augmentation by random cropping videos.

1 Introduction

Video-text large language models (LLMs) have developing rapidly in recent years to help people process videos more easily and faster. This development process includes the emergence of a number of new training corpora and models. However, most of the training corpora only include short videos with simple contents, in which a single keyframe often retains almost all the semantic information of the entire video. As a result, though models trained on these corpora can hold conversations and answer questions regarding short and simple videos, they do little to help us understand long-form videos like movies, tutorials, and documentaries that play an integral role in our daily lives and convey a wealth of information, knowledge, opinions, and emotions. In fact, understanding long-form videos can be very difficult for computers: they first have to understand the basic content and then the sequence of occurrence of multiple events that appear in the video.

However, existing LLMs perform far from satisfactory. For example, MVBench (Li et al., 2023c) and VITATECS (Li et al., 2023d) point out that even state-of-the-art video-text LLMs perform like chance on localizing actions or determining the order of events in videos, showing that though being the most substantial difference between videos and images, the ability to understand temporal information in videos still lags far behind for most video-text LLMs.

Recently there have been some works that focus on training video-text LLMs to reference video segments, such as TimeChat (Ren et al., 2023) and VTimeLLM (Huang et al., 2023). These approaches train models to reference video segments using numbers or special tokens to generate start and end timestamps. However, these models require extensive training on deliberately constructed instruction-tuning datasets to establish the correlation between the numbers and the corresponding video segments. We also found that the above-mentioned models did not perform well on common video understanding benchmarks like MVBench (Li et al., 2023c) and STAR (Wu et al., 2021).

To tackle this problem, in this paper we investigate enhancing the ability of temporal video grounding, a basic task for long-form video understanding, of video-text LLMs, with minimum hindrance to their performance on other video understanding tasks. We make improvements in the following two aspects: (1) designing better reference methods for LLMs to refer to video segments in text, and (2) constructing a large-scale instruction tuning dataset with segment-level annotations, and jointly train video-text LLMs on it as well as other instruction datasets. To improve aspect (1) we let LLMs use time verbals, such as "at the beginning", "at the middle", "at the end" or "throughout the entire video", to represent segments in videos. With our proposed recursive grounding technique, this reference method can also be used to refer to shorter and finer-grained video segments through multiple rounds of judgments. We show that compared to directly generating timestamps, the proposed verbal reference method enables video-text LLMs that have not been trained on any segment-level data to exhibit substantial capabilities of referring video segments, and is more effective and robust than its alternatives after fine-tuning. To improve aspect (2), we build InternVid-G, a large-scale video corpus with 715k segment-level captions and negative spans, which is suitable for constructing temporal video grounding training samples.

Based on the stage 2 checkpoint of VideoChat2 (Li et al., 2023c), by implementing the above improvements we train HawkEye, a video-text LLM with the ability to accomplish temporal video grounding task in a text-to-text manner. We evaluate its performance on various downstream benchmarks including temporal video grounding, question grounding, and video question answering. Experimental results show that HawkEye performs substantially better than VideoChat2 on temporal video grounding in a fully text-to-text manner without hurting the performance on other video-text tasks.

2 Related Works

With the development of image-text LLMs (Li et al., 2023a; Dai et al., 2023; Liu et al., 2023), many works aim to combine LLMs with video encoders to leverage the comprehension and generation capabilities of LLMs for video-related tasks (Zhang et al., 2023; Maaz et al., 2023; Li et al., 2023b,c; Wang et al., 2022, 2024b; Zhang et al., 2024). However, in most of the works the training data only contains captions or dialogues about the content of the entire video, which is not designed for LLMs to learn to reference certain parts of the video.

Recently there have been some attempts to train multi-modal LLMs to refer to parts of the visual input and enhance their localization abilities. For image-text LLMs, Kosmos-2 (Peng et al., 2023), Pink (Xuan et al., 2023), and the Qwen-VL series (Bai et al., 2023; Wang et al., 2024a) shows that LLMs can accomplish a wider variety of downstream tasks if they possess the ability to refer to regions of the image input in text.

For video-text LLMs, SeViLA (Yu et al., 2023) proposes a localizer to assign a relevance score to each frame in the video, which is then used to filter the relevant frames for video question answering. VTimeLLM (Huang et al., 2023), TimeChat (Ren et al., 2023) and VTG-LLM (Guo et al., 2024) explores to accomplish temporal video grounding in a fully text-to-text manner by using percentages, second numbers or special tokens to denote the start and end timestamps of video segments, and fine-tunes video-text LLMs on data reformatted from existing temporal grounding or dense captioning datasets.

Our work differs from theirs at: (1) using the the proposed verbal format to reference video segments, thus achieving better and more robust performance in temporal video grounding even without fine-tuning on segment-level data; (2) proposing a large-scale dataset InternVid-G with segment annotations that are especially suitable for temporal grounding training; and (3) existing works specifically target video grounding tasks by leveraging LLMs while our motivation is to train a general video-text LLM that still owns versatility on various tasks, so we also pay efforts on formatting visual grounding similar to other tasks and thus jointly training with many other video-text tasks.

3 Verbal Reference Method

When designing reference methods for LLMs to represent a video segment with text, an intuitive method is to tell the LLM in prompt how many frames are there in total and the timestamp of each frame. For example, "The `video contains %d frames sampled at %.1f, %.1f, ... seconds`", where `%d` is an integer representing the number of input frames, and `%.1f` is a float number representing the timestamp of each frame in seconds. This prompt can guide LLMs to output the start and end frames with a format like "`From frame 3 to frame 5`", or seconds with a format like "`4.0 - 12.2 seconds`". However, these **frame** or **second** reference methods are suboptimal, probably due to the numbers and timestamps in the prompt are difficult for LLMs to understand and analyze precisely (Schwartz et al., 2024). It also requires extensive training to make video-text LLMs to learn the correlation between the numbers and the corresponding video segments.

To alleviate these problems we propose **verbal** reference method. We categorize segments of a video into four classes represented by four different time verbs: "**beginning**", "**middle**", "**end**" and "**throughout**". If the length of a video segment is larger than half the length of the entire video, we categorize this segment as "**throughout** the entire video". If the entire segment is in the first half of the video, we categorize this segment as "at the **beginning** of the video". If the entire segment is in the second half of the video, we categorize this segment as "at the **end** of the video". Otherwise, we categorize this segment as "in the **middle** of the video". An illustration of this categorization is shown in Appendix A. On the contrary, if the LLM generates a time verbal such as "at the beginning of the video" or "throughout the entire video", we will know that it is referring to the first half of the video or the entire video.

However, by solely using this reference method, an LLM can only represent a segment with full length or half-length of the video. Motivated by the idea of binary search, we propose **recursive grounding**, which enables the model to represent shorter video segments via multiple rounds of expression. The pseudo-code and a real case of recursive grounding are shown in Appendix A. Intuitively, the model first watches the entire video by sampling frames and determining an approximate time interval of the segment of interest. In the next round, the model focuses on the time interval found in the previous round, and further narrow down the range of the interval again, serving like a video binary search. This process is repeated until the model outputs "throughout the entire video" to break the loop or a maximum number of rounds is reached. Though recursive grounding may not cover all corner cases, In Sec. 6.1 we will show that its theoretical performance upper bound is much higher than the performance of all existing models, and it shows better and more robust performance than its alternatives of using second or frame numbers, which serves as a good trade-off between precision and expression difficulty.

4 InternVid-G Dataset

4.1 Dataset Construction

The pressing matter for improving LLMs' ability on temporal video grounding is to construct a large-scale training dataset. Different from existing large-scale video-text datasets like WebVid (Bain et al., 2021) which only contains short videos and corresponding captions, the dataset we plan to use for temporal video grounding training needs to meet the following requirements: (1) The videos should be long and contain multiple events; (2) Captions are annotated at segment level, *i.e.,* each caption should be paired with a segment of the video with a certain start position and end position; (3) Captions need to correspond to the semantic content of video scenes, instead of simply using ASR results of the corresponding audio like HowTo100M (Miech et al., 2019), and (4) The content of each caption only describes one segment (its paired segment) in the video input to the model. We construct InternVid-G (G for Grounding), a dataset that meets all of the above requirements with diverse topics and backgrounds. Fig. 1 shows an overview of InternVid-G with 8 consecutive video segments. This dataset is constructed through the following steps:

Scene Segmentation. We randomly download 100k (1%) videos from InternVid-10M-FLT (Wang et al., 2023), as these videos cover diverse categories and cultural backgrounds. We use PySceneDetect[1] to split video into scenes. However, PySceneDetect splits the video by detecting abrupt changes in pixels of adjacent frames. In contrast, as we aim to segment videos into scenes with different semantic content, this toolkit results in a number of false positive segmentations (*e.g.,* splitting the video of the same event from different camera angles into different scenes, which is not what we expect). To tackle with this problem, we use CLIP (Radford et al., 2021) to calculate the semantic similarity between each pair of adjacent scenes, and merge the adjacent scenes if the similarity score between them is higher than a threshold.

Scene Captioning and Filtering. As the internal differences of the segments obtained through the above segmentation are subtle, for each video segment we sample the center frame and use BLIP-2 (Li et al., 2023a) to generate a caption due to its

[1]https://github.com/Breakthrough/PySceneDetect

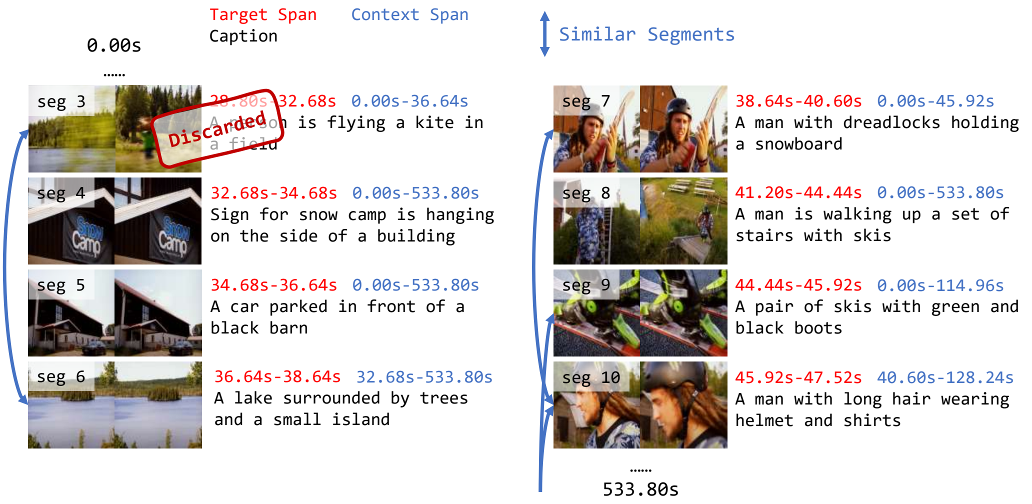

Figure 1: An overview of the InternVid-G dataset. Segment 3 is discarded due to its similarity with the given caption is lower than a threshold. Segment 10's context span starts from $40.60s$ (in blue) as this timestamp is the end position of its similar segment 7, which should not be included since the caption "`A man with long hair wearing helmet and shirts`" of segment 10 also owns a high similarity with segment 7. Conversely, the end position of the context span of segment 7 is also the start position of segment 10.

ability of generating short and low-hallucination text. To ensure the quality of the captions, we use CLIP to calculate the similarity between the caption and the video segment, and only keep half of all the captions that has higher similarity than a threshold to the video segment. For example, In Fig. 1 the caption of segment 3 owns a relatively lower similarity score with the video so it is discarded. Note that this does not indicate the video segment of this sample has lost its value, as it still can be included in the context span of other samples.

Context Span Mining. Temporal video grounding requires a model to reference a video segment which is relevant to the given text query from a long video context. To construct samples for this task, in addition to the video segment and its corresponding caption as query, we also need several other segments before and after this segment as the *video context* to retrieve from. We term the video segment corresponding to the query as the *target span* of an example, and (the context segments before & after it + target span) as the *context span*, and models are required to locate the target span inside the context span to perform video grounding.

One notable issue is the context span should not contain other video segments that are too similar to the target span, otherwise these segments can also correspond to the query and will introduce noises. To prevent this, we calculate the similarity with CLIP between the target span and all other segments in the video, and label the segments with a similarity score above a threshold as *similar segments*. Thus, for a particular text query paired with its target span, the start position of its paired context span should be the end position of the last *similar segment* before its target span. Similarly, the end position of its context span should be the start position of the first *similar segment* after its target span. We term these two positions as `ctx_start` and `ctx_end`, and the start & end position of the target span as `tar_start` and `tar_end`. If there are no similar segments before or after the target span, then the `ctx_start` or `ctx_end` will be set to 0 (*i.e.,* the beginning of the video) or the end position of the last segment in the video. The context span mining are also shown in Fig. 1. For example, as segment 3 is a similar segment of segment 6, the `ctx_start` of segment 6 is set as the `tar_end` as segment 3 to ensure that the video of segment 3 is not included in the context span of segment 6.

4.2 Data Statistics and Features

Table 1 shows the dataset statistics. Our InternVid-G is the largest dataset in size compared to other temporal grounding datasets, and has the strongest diversity in video as they are sourced from the largest video platform YouTube. For detailed video statistics such as video categories, please refer to (Wang et al., 2023), as the videos used in InternVid-

	#videos	#queries	tar./ctx. avg. span len (s)	video source
DiDeMo	10642	41206	6.9/29.3	Flickr
Charades-STA	9848	16124	8.1/30.6	Activity
ANet-Captions	14926	71957	37.1/117.6	Activity
InternVid-G	83614	715489	4.4/203.4	YouTube

Table 1: Dataset Statistics compared with several temporal grounding datasets.

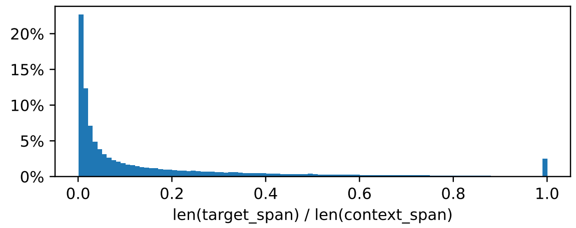

Figure 2: The length ratio of target spans in context spans of InternVid-G.

G are an unbiased-sampled subset of InternVid-10M-FLT.

5 HawkEye

In this section, we describe the training process of HawkEye, a video-text LLM that is fine-tuned with the verbal reference method to refer to video segments. Note that our aim is not to train a state-of-the-art video-text LLM, as performance improvements can always be achieved by using better initialization LLMs and training data. Instead, we aim to demonstrate the performance and robustness of the verbal reference method with a limited computation budget.

We initialize HawkEye with the stage 2 checkpoint of VideoChat2 (Li et al., 2023c). We make modifications to the instruction tuning data (VideoChat2-IT) used in stage 3. Due to limited computation budget and targeting at video segmentation representation, we only use video instruction data and remove image data from VideoChat2-IT. Unless otherwise specified, we sample 12 frames from the video as visual input. We fine-tune the Q-Former, query tokens and use LoRA (Hu et al., 2022) to fine-tune the LLM, while keep the visual encoder frozen. Details of datasets used in the training process are listed in Appendix B.

We add two time-aware tasks based on InternVid-G to VideoChat2-IT: **temporal video grounding** and **video segment captioning**. When training on the temporal video grounding task with verbal reference method, each training sample is formatted as a multiple-choice question. We use the query as input and ask the model to choose one of the following 4 temporal statements: "At the beginning of the video.", "In the middle of the video.", "At the end of the video." and "Throughout the entire video".

Random Cropping as Data Augmentation. There are 2 problems that prevent us from directly applying InternVid-G for training. The first problem is the length proportion of target spans in context spans is very unevenly distributed. As shown in Fig. 2, for most of the examples the target span only takes up a very small portion of the context span, as videos from YouTube usually have tens of minutes in length and many segments only last for a few seconds. The second problem is since the position of target span in the context span is fixed for each example, the model may tend to overfit on shortcut relations between the text description and this position if it sees the same example multiple times in different epochs, especially when the amount of training data is small (*e.g.,* fine-tuning on small temporal grounding datasets like Charades-STA (Gao et al., 2017)). To solve the above problems and perform data augmentation, we propose a random cropping method: we crop the video input by sampling the start position of the video input in the interval of [`ctx_start`, `tar_start`] and end position in the interval of [`tar_end`, `ctx_end`]. Note that after which, the cropped video input always include the target span. This cropping method enables the same text query to have different answers in different epochs, and can also make the four temporal categories of video segments have roughly the same probability to occur in each epoch. A demonstration of the cropping process is shown in Fig. 3. Random cropped sample 1 and 2 both include the targeted video segment from $38s$ to $40s$ (in red rectangle) but with different start and end positions ($31s$ to $40s$ in purple for sample 1, $36s$ to $42s$ in green for sample 2) from the original video, thus their video inputs and the answers are different from each other.

As the training data size of temporal grounding is significantly larger than other tasks in VideoChat2-IT, to prevent the distribution of training data from being too biased against this multiple-choice task which may hurt the model's versatility, we also add a video segment captioning task: the model is asked to generate a caption of the target span given the video clip cropped from the context span

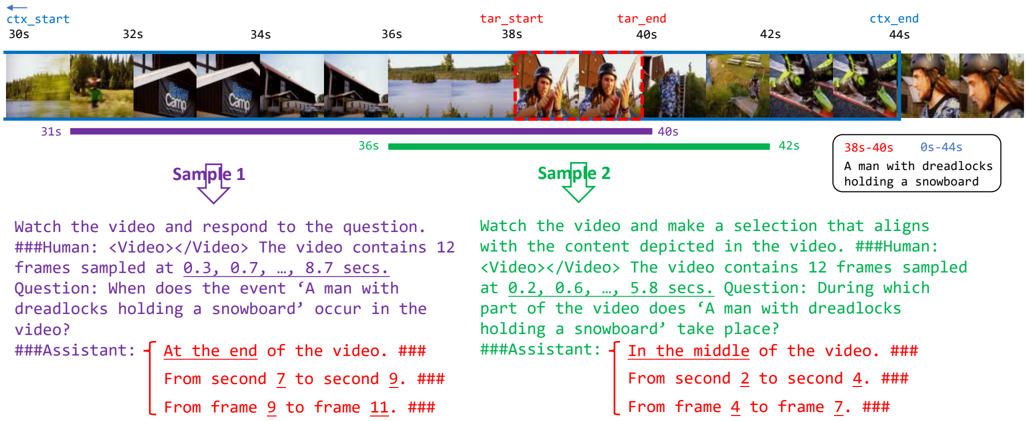

Figure 3: A demonstration of random cropping when training on temporal video grouding. The differences between two samples (sample 1 in purple and sample 2 in green) are emphasized with underlines, they share the same query and target video segment but not the same answers. The frame-level and second-level representation are also presented in red.

with random cropping data augmentation method and the verbal statement (which is the ground truth answer in temporal video grounding task).

6 Experiments

6.1 Temporal Video Grounding

We validate the temporal video grounding ability of HawkEye and other models on two popular benchmarks: Charades-STA (Gao et al., 2017) and ActivityNet-Captions (Krishna et al., 2017).

6.1.1 Comparison of Reference Methods

Verbal method enables models not trained explicitly on temporal video grounding to reference video segments better. We compare temporal grounding results of VideoChat2 (Li et al., 2023c), LLaVA-OneVision 7B (LLaVA-OV) (Li et al., 2024) and InternLM-XComposer 2.5 (IXC2.5) (Zhang et al., 2024) using second and verbal reference method in Table 2. For the verbal reference method, we tried setting the max number of recursive grounding rounds in {1,2,3} and the best result (1 round for Charades-STA and 2 rounds for ActivityNet-Captions) is reported. The "random" baseline denotes randomly choosing a span with the average length of ground truth spans from the train set. The "verb. upbound" baseline is the best result that running recursive grounding for 3 rounds can achieve. It is obtained by choosing the best result of $4^3 = 64$ possible answers.

Table 2: Zero-shot performance on temporal video grounding for video-text LLMs that have never trained on any segment-level annotations. Four metrics reported are mIoU/R@IoU>0.3/0.5/0.7, the higher the better. †: training data of this model contains videos from the training set of this benchmark, thus this is not under a strict zero-shot setting. *: ActivityNet-Captions is used when training TimeChat thus the authors did not report this performance.

	Charades-STA	ActivityNet-Captions
Baselines		
random	20.1/30.0/18.8/6.2	23.0/29.0/15.1/6.1
verb. upbound	74.8/100.0/97.0/69.2	71.9/91.5/84.6/68.4
VTimeLLM	31.2/51.0/27.5/11.4	30.4/44.0/27.8/14.3
TimeChat	- / - /32.2/13.4	Not Applicable*
VideoChat2		
second	15.2/20.2/8.0/2.9	12.6/16.9/9.3/4.5
verbal	24.6/38.0/14.3/3.8	27.9/40.8/27.8/9.3
LLaVA-OV		
second	13.1/18.4/6.7/2.0†	14.6/18.3/8.6/3.7†
verbal	34.6/53.1/34.0/13.2†	32.2/45.0/29.6/14.5†
IXC 2.5		
second	13.9/19.7/8.2/3.0	22.8/33.2/14.5/6.0†
verbal	31.2/46.2/31.5/11.7	34.1/47.4/27.2/13.5†

Though the verbal reference method may not sound very precise for representing time spans, its potential precision is sufficient for temporal video grounding tasks, especially when the IoU threshold is not very high. For instance, the accuracy upperbound of R@IoU>0.5 on Charades-STA reaches 97.0 after 3 recursive turns, which is nearly perfect and is already much higher than the performance of all state-of-the-art temporal grounding methods. Given the substantial differences in base LLMs as initialization and the training data, the performance across different models can not be compared directly. However, it is clearly shown that **all models achieve significantly better performance when using verbal instead of second reference method.**

Ref. Method	IT only (zero-shot)	FT only	IT+FT
Frame	24.8/40.3/23.7/8.8	27.5/44.7/21.2/6.3*	47.2/72.7/53.0/25.3
Second	20.6/35.7/14.0/2.1	42.3/63.0/47.0/24.6	49.0/72.1/54.5/29.6
Verbal	33.2/**54.1**/23.8/9.1	42.4/71.9/37.0/13.9	43.1/72.7/38.2/14.4
Verbal+RG.	**33.7**/50.6/**31.4**/**14.5**	**48.2**/**72.2**/**55.8**/**27.1**	**50.3**/**74.8**/**60.3**/29.5

Table 3: Performance of different video segment reference methods on the testset of Charades-STA. IT denotes adding InternVid-G into stage 3 instruction tuning data, FT denotes fine-tuning on the train set of Charades-STA before testing, and RG. denotes recursive grounding. All models are initialized with stage 2 checkpoint of VideoChat2. Four metrics reported are mIoU and R@IoU >0.3/0.5/0.7, where the higher the better. * fails to generate well-formatted outputs for almost half samples, so we can only take the correctly formatted ones into account.

	SODA$_c$	CIDEr	METEOR
VideoChat	0.9	2.2	0.9
VideoLLaMA	1.9	5.8	1.9
VideoChatGPT	1.9	5.8	2.1
HawkEye	**5.8**	**8.8**	**4.6**

Table 4: Performance of dense video captioning on ActivityNet-Captions.

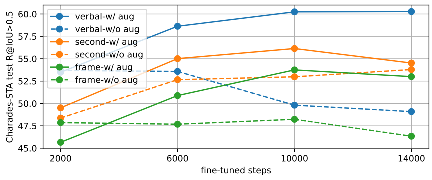

Figure 4: Temporal video grounding with and without random cropping.

	Charades-STA	ActivityNet-Captions
VideoChat2	24.6/38.0/14.3/3.8	27.9/40.8/27.8/9.3
VideoChat2 †	23.4/35.4/12.6/3.0	28.2/41.7/28.7/9.4
SeViLA	18.3/27.0/15.0/5.8	23.0/31.6/19.0/10.1
VTimeLLM	31.2/51.0/27.5/11.4	30.4/44.0/27.8/14.3
TimeChat	- / - /32.2/13.4	Not Applicable*
HawkEye	**33.7**/50.6/31.4/**14.5**	**32.7**/**49.1**/**29.3**/10.7

Table 5: Zero-shot performance of temporal video grounding. Four metrics reported are mIoU and R@IoU >0.3/0.5/0.7, where the higher the better. *ActivityNet-Captions is used when training TimeChat, and thus the authors did not report this performance. †: Our Implementation.

Verbal method is more robust and data-efficient for training. For models that are explicitly trained on temporal video grounding task, we investigate models trained with three reference methods: verbal (with and without recursive grounding), frame, and second reference method. Table 3 shows though the performances of all reference methods after IT+FT are comparable, verbal reference method performs significantly better than other alternatives when only IT or FT is applied, which shows that our verbal reference method is **more robust in zero-shot manner** across the different distributions of videos (IT only) and **much easier to be understood by the LLM** especially when the amount of training data is small (FT only). We also conduct experiments with different number of input frames, maximum recursive grounding rounds and test the inference speed to further demonstrate the robustness and efficiency of HawkEye, which are elaborated in Appendix C.

6.2 Comparison with Baseline Models

We compare zero-shot temporal video grounding performance of HawkEye with other video-text LLMs, as shown in Table 5. SeViLA Localizer (Yu et al., 2023) is a BLIP2-based LLM that generates a float number as the score for each frame. Following their original implementation, we use 64 frames as input, and uses their method of aggregation for video moment retrieval. TimeChat (Ren et al., 2023) and VTimeLLM (Huang et al., 2023) are two video-text LLMs that are also able to perform temporal video grounding in a text-to-text manner with using either seconds or the percentage of the video. For VideoChat2 and HawkEye, we use 12 frames as video input. To provide a fair comparison with HawkEye, in order to reveal the role of using InternVid-G in instruction tuning, we also re-implement VideoChat2 by reperforming the training stage 3. Our implementation only uses the video instruction data of VideoChat2-IT, and similarly we only fine-tune query tokens, Q-Formers and LoRA of LLMs. In a word, the only difference between our implemented VideoChat2 and HawkEye is the use of InternVid-G in stage 3 instruction tuning. To evaluate VideoChat2 and HawkEye, we use recursive grounding and report results with the best maximum number of recursive rounds.

Compared to other video-text LLMs, **HawkEye achieves the best performance on zero-shot temporal video grounding, despite having a conservative training plan by not training on any human-annotated temporal video grounding data and using only 12 frames as video input** (in contrast, 100 frames for VTimeLLM, 96 frames for TimeChat and 64 frames for SeViLA Localizer). We also conducted experiments of fine-tuned

	mIoU/R@IoU >0.3/0.5
Fine-Tuned Models	
Temp[CLIP] NG+	12.1/17.5/8.9
FrozenBiLM NG+	9.6/13.5/6.1
Zero-Shot LLMs	
SeViLA Localizer	21.7/29.2/13.8
VideoChat2	24.1/36.3/16.2
VideoChat2 †	18.7/26.6/13.5
HawkEye	**25.7/37.0/19.5**

Table 6: Performance of temporal video grounding of questions on the NExT-GQA. †: Our implementation

	MVBench	NExT-QA*	TVQA	STAR
TimeChat	35.93	43.59	31.33	37.97
VideoChat2	51.10	66.50	40.60	59.00
VideoChat2†	49.75	66.91	40.15	55.24
HawkEye	47.55	**67.93**	**41.10**	56.59

Table 7: Performance on various video question answering benchmarks without fine-tuning. *: As the instruction tuning dataset VideoChat2-IT contains training data from NExT-QA, therefore this benchmark is not tested under a strict zero-shot setting for HawkEye and VideoChat2. †: Our implementation.

temporal video grounding to further demonstrate HawkEye's outstanding performance in Appendix C.

6.3 Temporal Video Grounding of Questions

In addition to retrieving video segments with descriptive statements, grounding questions in videos is a more difficult yet important problem, as it is a crucial step towards explainable video QA. Table 6 shows experiments on temporal video grounding of questions from the testset of NExT-GQA (Xiao et al., 2023), where models are required to find out a segment from the video that addresses the given question. HawkEye outperforms baseline methods, both fine-tuned models (Xiao et al., 2023; Yang et al., 2022) and zero-shot LLMs, with a large margin.

6.4 Video Question Answering

As HawkEye is trained by adding two time-aware tasks from InternVid-G to stage 3 of VideoChat2, one may wonder whether introducing such a large amount of training examples from InternVid-G may lead to an unbalanced distribution of instruction tuning dataset, which will eventually impair the model's performance on other video-text tasks. Results in Table 7 show that HawkEye has comparable performance with VideoChat2 (our implementation) on a variety of video question answering tasks (Li et al., 2023c; Xiao et al., 2021; Lei et al., 2018; Wu et al., 2021), **which shows that introducing InternVid-G in instruction tuning does not impair the model's versatility.** In contrast, TimeChat only focuses on time-aware tasks and wasn't instruction-tuned on diverse tasks does not have this versatility as it performs poorly on QA tasks.

6.5 Dense Video Captioning

Though HawkEye was not trained on any dense video caption data, we also tested it on ActivityNet by generating 3 captions for the beginning, the middle, and the end of the video. The results are listed in Table 4, which shows that HawkEye outperforms all other video-text LLMs that are also not trained on dense video captioning.

6.6 Ablation Studies

Random cropping is useful for all reference methods. As explained earlier when introducing random cropping in Sec. 5, random cropping is essential when instruction-tuning on InternVid-G, so we perform ablation studies of random cropping by fine-tuning on Charades-STA after instruction tuning. As shown in Fig. 4, using random cropping consistently leads to better performance for all reference methods.

7 Conclusion

In this paper, we propose a simple and efficient method to enhance temporal video grounding of video-text LLMs, with minimal affections to other abilities. We propose time verbal representation method that refers to video segments with time verbals instead of numbers for timestamps, which is easier for LLMs to follow even without targeted training on segment-level annotations. We construct InternVid-G, a large-scale video-text corpus with segment-level captions and context spans, which is very suitable for training on temporal video grounding tasks. We propose a set of feasible practices, including the determination of context spans, the random cropping data augmentation method, and two time-aware training objectives (*i.e.,* temporal video grounding and video segment captioning), to train video-text and enhance their temporal video grounding abilities. Based on the above efforts we train HawkEye, a video-text LLM that is able to perform temporal video grounding with time verbal representation format, and evaluate on a variety of temporal video grounding and video QA benchmarks.

Limitations

A potential limitation is that it is hard to represent multiple video segments within one response, such as when performing tasks like highlight detection (Lei et al., 2021), with verbal reference method.

Instruction data used for training HawkEye is also relatively simple. Adding some multi-round conversations or replies involving multiple video segments can make the model more useful and better need real-word needs.

References

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*.

Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. 2021. Frozen in time: A joint video and image encoder for end-to-end retrieval. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1708–1718.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. 2023. InstructBLIP: Towards general-purpose vision-language models with instruction tuning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

J. Gao, Chen Sun, Zhenheng Yang, and Ramakant Nevatia. 2017. Tall: Temporal activity localization via language query. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5277–5285.

Yongxin Guo, Jingyu Liu, Mingda Li, Xiaoying Tang, Xi Chen, and Bo Zhao. 2024. Vtg-llm: Integrating timestamp knowledge into video llms for enhanced video temporal grounding. *ArXiv*, abs/2405.13382.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models.

Bin Huang, Xin Wang, Hong Chen, Zihan Song, and Wenwu Zhu. 2023. Vtimellm: Empower llm to grasp video moments. *ArXiv*, abs/2311.18445.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. 2017. Dense-captioning events in videos. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 706–715.

Jie Lei, Tamara L. Berg, and Mohit Bansal. 2021. Qvhighlights: Detecting moments and highlights in videos via natural language queries. *ArXiv*, abs/2107.09609.

Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L. Berg. 2018. Tvqa: Localized, compositional video question answering. In *Conference on Empirical Methods in Natural Language Processing*.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. 2024. Llava-onevision: Easy visual task transfer. *ArXiv*, abs/2408.03326.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. 2023a. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International Conference on Machine Learning*.

Kunchang Li, Yinan He, Yi Wang, Yizhuo Li, Wen Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. 2023b. Videochat: Chat-centric video understanding. *ArXiv*, abs/2305.06355.

Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, Limin Wang, and Yu Qiao. 2023c. Mvbench: A comprehensive multi-modal video understanding benchmark. *ArXiv*, abs/2311.17005.

Shicheng Li, Lei Li, Shuhuai Ren, Yuanxin Liu, Yi Liu, Rundong Gao, Xu Sun, and Lu Hou. 2023d. Vitatecs: A diagnostic dataset for temporal concept understanding of video-language models. *ArXiv*, abs/2311.17404.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. 2019. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2630–2640.

WonJun Moon, Sangeek Hyun, Sang shin Paldal-gu Suwon-city Park, Dongchan Park, and Jae-Pil Heo. 2023. Query - dependent video representation for moment retrieval and highlight detection. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23023–23033.

Guoshun Nan, Rui Qiao, Yao Xiao, Jun Liu, Sicong Leng, Hao Howard Zhang, and Wei Lu. 2021. Interventional video grounding with dual contrastive learning. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2764–2774.

Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. 2023. Kosmos-2: Grounding multimodal large language models to the world. *ArXiv*, abs/2306.14824.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*.

Shuhuai Ren, Linli Yao, Shicheng Li, Xu Sun, and Lu Hou. 2023. Timechat: A time-sensitive multimodal large language model for long video understanding. *ArXiv*, abs/2312.02051.

Eli Schwartz, Leshem Choshen, Joseph Shtok, Sivan Doveh, Leonid Karlinsky, and Assaf Arbelle. 2024. Numerologic: Number encoding for enhanced llms' numerical reasoning. *ArXiv*, abs/2404.00459.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Ke-Yang Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024a. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *ArXiv*, abs/2409.12191.

Yi Wang, Yinan He, Yizhuo Li, Kunchang Li, Jiashuo Yu, Xin Jian Ma, Xinyuan Chen, Yaohui Wang, Ping Luo, Ziwei Liu, Yali Wang, Limin Wang, and Y. Qiao. 2023. Internvid: A large-scale video-text dataset for multimodal understanding and generation. *ArXiv*, abs/2307.06942.

Yi Wang, Kunchang Li, Xinhao Li, Jiashuo Yu, Yinan He, Guo Chen, Baoqi Pei, Rongkun Zheng, Jilan Xu, Zun Wang, Yansong Shi, Tianxiang Jiang, Songze Li, Hongjie Zhang, Yifei Huang, Yu Qiao, Yali Wang, and Limin Wang. 2024b. Internvideo2: Scaling video foundation models for multimodal video understanding. *ArXiv*, abs/2403.15377.

Yi Wang, Kunchang Li, Yizhuo Li, Yinan He, Bingkun Huang, Zhiyu Zhao, Hongjie Zhang, Jilan Xu, Yi Liu, Zun Wang, Sen Xing, Guo Chen, Junting Pan, Jiashuo Yu, Yali Wang, Limin Wang, and Yu Qiao. 2022. Internvideo: General video foundation models via generative and discriminative learning. *ArXiv*, abs/2212.03191.

Bo Wu, Shoubin Yu, Tenenbaum-Joshua B Chen, Zhenfang, and Chuang Gan. 2021. Star: A benchmark for situated reasoning in real-world videos. In *Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS)*.

Junbin Xiao, Xindi Shang, Angela Yao, and Tat seng Chua. 2021. Next-qa: Next phase of question-answering to explaining temporal actions. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9772–9781.

Junbin Xiao, Angela Yao, Yicong Li, and Tat-Seng Chua. 2023. Can i trust your answer? visually grounded video question answering. *ArXiv*, abs/2309.01327.

Shiyu Xuan, Qingpei Guo, Ming Yang, and Shiliang Zhang. 2023. Pink: Unveiling the power of referential comprehension for multi-modal llms. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13838–13848.

Antoine Yang, Antoine Miech, Josef Sivic, Ivan Laptev, and Cordelia Schmid. 2022. Zero-shot video question answering via frozen bidirectional language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Shoubin Yu, Jaemin Cho, Prateek Yadav, and Mohit Bansal. 2023. Self-chained image-language model for video localization and question answering. *ArXiv*, abs/2305.06988.

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

Pan Zhang, Xiao wen Dong, Yuhang Zang, Yuhang Cao, Rui Qian, Lin Chen, Qipeng Guo, Haodong Duan, Bin Wang, Linke Ouyang, Songyang Zhang, Wenwei Zhang, Yining Li, Yang Gao, Peng Sun, Xinyue Zhang, Wei Li, Jingwen Li, Wenhai Wang, Hang Yan, Conghui He, Xingcheng Zhang, Kai Chen, Jifeng Dai, Yu Qiao, Dahua Lin, and Jiaqi Wang. 2024. Internlm-xcomposer-2.5: A versatile large vision language model supporting long-contextual input and output. *ArXiv*, abs/2407.03320.

A More Details of the Verbal Reference Method

An illustration of the conversion between video segments and time verbal categories is shown in Fig. 6.

An real example of the recursive grounding process is shown in Fig. 5. The pseudo-code of the recursive grounding process is as follow:

```
def recursive_grounding(video, query, max_rounds):
    start, end = 0, get_length(video)
    for round_i in range(max_rounds):
        clip_length = end - start
        res = choice(video[start: end], query)
        if res == 'throughout':
            break
        elif res == 'beginning':
            end -= clip_length / 2
        elif res == 'end':
            start += clip_length / 2
        elif res == 'middle':
            start += clip_length / 4
```

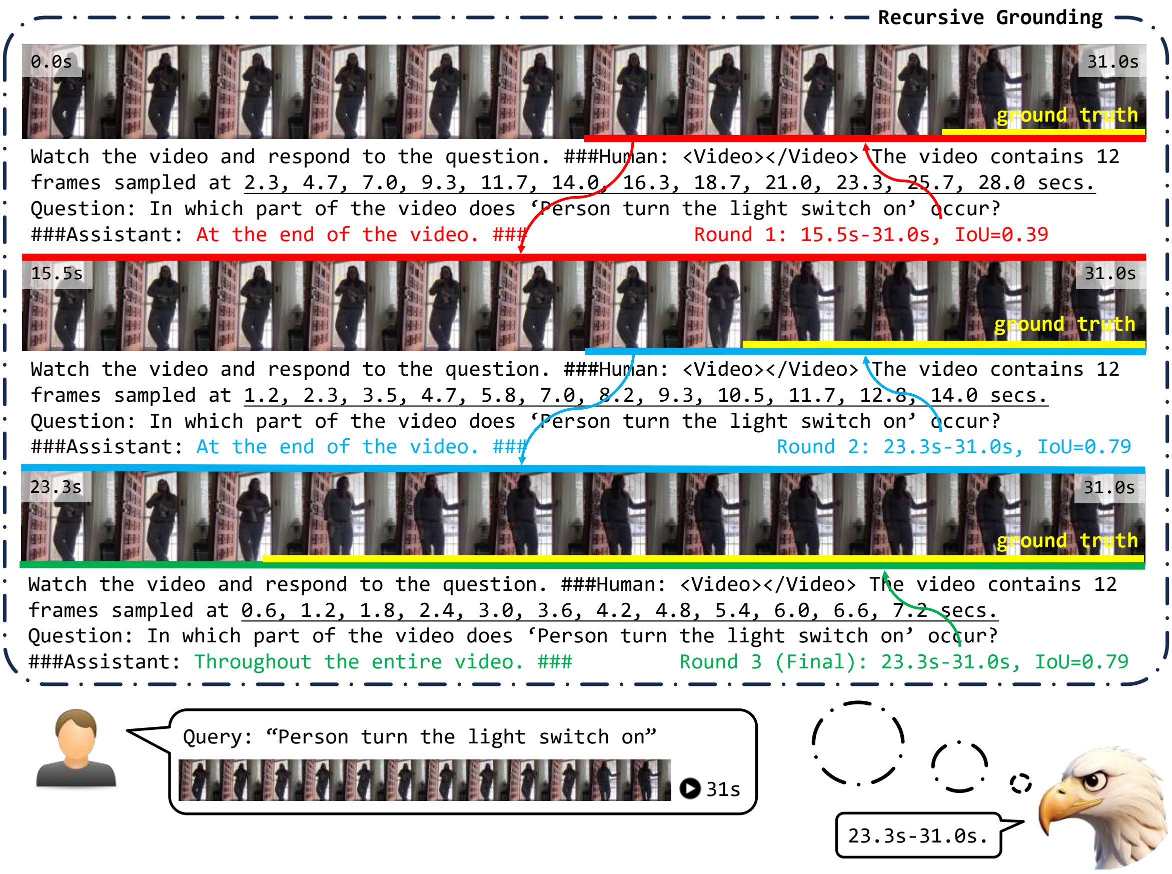

Figure 5: An example of the recursive grounding process of HawkEye. After receving the video and the query from the user, HawkEye automatically performs the grounding process with recursive grounding in the background, and converts the target span into a user-friendly `start_sec - end_sec` format. The four choices of the verbal reference method are also described in the prompt, but here we omit them to make the presentation simpler.

```
            end -= clip_length / 4
    return start, end
```

B Training and Experiment Details

B.1 Datasets Used

We list the datasets used in the training process of HawkEye in Fig. 7. We train HawkEye for 1M steps, which takes about 7 days with 8 V100 GPUs.

B.2 Prompts used in training HawkEye

When training HawkEye on temporal video grounding task, we use the following prompt:

```
<Instruction>
###Human: <Video></Video> The video contains 12 frames
sampled from %.1f, %.1f, %.1f, %.1f, %.1f, %.1f, %.1f,
%.1f, %.1f, %.1f, %.1f, %.1f seconds.
###Human: Question: <Question> Options: <Options>
###Assistant: <Answer> ###
```

Where `%.1f` denotes the timestamp of a frame rounded to 1 decimal places, `<Instruction>` is randomly sampled from the following 10 instructions:

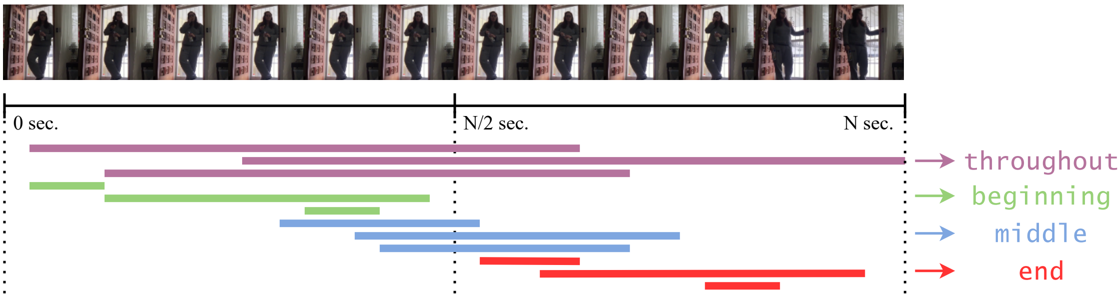

Figure 6: Illustration of the mapping from video segments to the categories of choices in our verbal reference method.

Conversation	Num
VideoChat	13884
VideoChatGPT	13303

Captioning	Num
WebVid	400000
VideoChat	6905
YouCook2	8760
TextVR	39648

VQA	Num
TGIF-trans	39149
TGIF-frame	52696
WebVidQA	10000
EgoQA	7813

Classification	Num
Kinetics-710	40000
SthSthV2	40000

Reasoning	Num
NExTQA	34132
CLEVRER-MC	42620
CLEVRER-QA	40000

Time-Aware	Num
InternVid-G Grounding	715489
InternVid-G Captioning	715489

Existing New

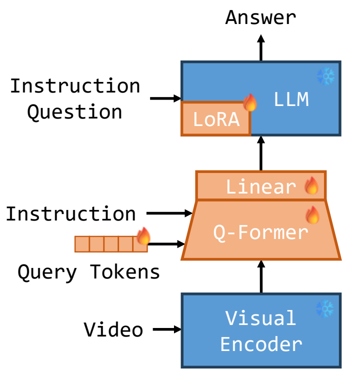

Figure 7: Datasets used and parameters trained in the instruction tuning of HawkEye.

```
Evaluate the video content and select the most suitable option based on what is presented in the video.
Examine the video and choose the most appropriate choice in accordance with the video's content.
Watch the video and make a selection that aligns with the content depicted in the video.
Analyze the video and opt for the choice that best corresponds to the content captured in the footage.
Assess the video content and choose the option that aligns most closely with what is presented in the video.
Evaluate the video, then select the most fitting choice based on the content portrayed.
Examine the video and make a decision based on the content presented in the footage.
Watch the video and choose the most fitting option based on the observed content.
Assess the video content and choose a suitable option based on what is portrayed in the video.
Analyze the video and select the most appropriate choice in relation to the content featured in the video.
```

<Question> is randomly sampled from the following 10 questions (%s denotes the query):

```
When does '%s' happen in the video?
At what time does the occurrence of '%s' take place in the video?
During which part of the video does '%s' occur?
At what point in the video does the event '%s' happen?
When in the video does the '%s' incident occur?
At which moment does '%s' take place in the video?
During which phase of the video does '%s' happen?
When does the '%s' event occur in the video?
At what time does '%s' occur in the video sequence?
When does the '%s' situation take place in the video?
```

<Options> are the following 4 statements randomly shuffled (Of course the letters will also be modified accordingly):

```
(A) At the beginning of the video.
(B) At the middle of the video.
(C) At the end of the video.
(D) Throughout the entire video.
```

and <Answer> is the ground truth statement.

When training HawkEye on video segment captioning task, we use the following prompt:

```
<Instruction>
###Human: <Video></Video> The video contains 12 frames
sampled from %.1f, %.1f, %.1f, %.1f, %.1f, %.1f, %.1f,
%.1f, %.1f, %.1f, %.1f, %.1f seconds.
###Human: <Statement>
###Assistant: <Answer> ###
```

where `<Instruction>` is randomly sampled from the following 10 instructions:

```
Analyze the video content within the specified time frame
and provide a detailed description of the scenes during
that period.
Given a specific time span, describe the activities or
events taking place in the corresponding section of the
video.
Examine the scenes within the indicated time range and
generate a textual overview of the objects, actions, and
context.
Provide a comprehensive narrative of the content depicted
in the video during the given time span, emphasizing key
elements and notable occurrences.
For the specified video duration, outline the main themes
and subjects present.
Annotate the content within the indicated time interval,
focusing on the details of people, objects, and actions
captured in the video during that specific duration.
Describe the visual and contextual aspects of the video
scenes within the provided time range.
Summarize the content of the video within the specified
time span.
Examine the video scenes within the given time frame and
provide a detailed description of that segment.
Offer a textual analysis of the video content
corresponding to the specified time duration.
```

`<Statement>` is randomly sampled from the following 4 temporal statements:

```
At the beginning of the video.
At the middle of the video.
At the end of the video.
Throughout the entire video.
```

and `<Answer>` is the ground truth caption (text query).

Only `<Answer> ###` is calculated in the training loss.

C More Experiments about the verbal reference method

C.1 Robustness across Different Input Frames

It is a good feature if the number of input frames can be adjusted according to computing constraints or accuracy requirements. In this experiment we train all models using IT+FT with 12 frames, and test with 8, 12 or 16 frames from the video as input. Fig. 8 shows that though the number of input frames and timestamps of all frames are provided in the prompt, the performance of using frame numbers or seconds to represent video segments tends to drop drastically when the number of input frames changes, probably due to the numbers and timestamps in the prompt are difficult for LLMs to understand and analyze. In contrast, our verbal reference method is robust to different numbers of input frames.

C.2 Affect of Maximun Recursive Grounding Rounds

we take a deeper look at how the `max_rounds` hyper-parameter of recursive grounding affects the grounding performance, as shown in Fig. 9. When HawkEye is not fine-tuned, using smaller `max_rounds` hyper-parameter is more likely to achieve better grounding results, as the more rounds there are, the more likely it is for the model to make mistakes on judging the location of the segment related to a query. However, after fine-tuned on the training set of the benchmark, the grounding performance hardly drops when the `max_rounds` hyper-parameter goes larger. This is probably because the model has already gained a good understanding of the video content, so it is less likely to make mistakes and knows in which round the video is already trimmed to the targeted segment relevant to the query, and then the model can reply a "throughout" to break the loop.

Comparing the performance of using only one and more than one round, we find that when only one round of localization is available (e.g., referring to a video segment in the middle of a conversation), fine-tuned HawkEye can not achieve its optimal performance. However, for the non-fine-tuned HawkEye which can be used under the majority of zero-shot conditions, this performance loss is less severe.

C.3 Fine-Tuned Temporal Video Grounding Experiments

We also compare the performance of fine-tuned HawkEye with SOTA specialist models and other LLMs. As shown in Table 8, HawkEye outperforms VideoChat2 (our implementation) and TimeChat, while still underperforms SOTA specialists. This indicates that though training with a large

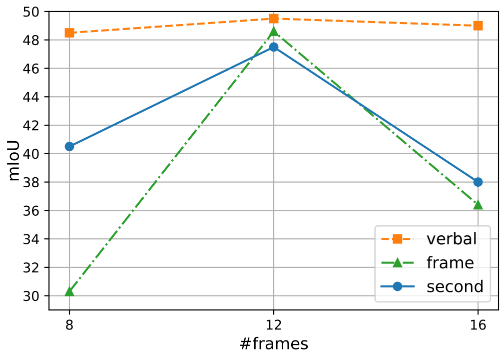

Figure 8: Performance of HawkEye on the test set of Charades-STA with different numbers of frames as video input during inference.

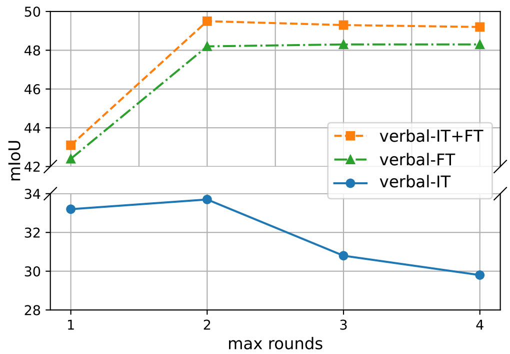

Figure 9: Performance of HawkEye on the test set of Charades-STA with different maximum rounds of recursive grounding.

Model	Charades-STA	ActivityNet-Captions
SOTA Specialist	- / - /57.3/32.5	44.2/63.2/43.8/27.1
TimeChat	- / - /46.7/23.7	-
VideoChat2 †	48.3/71.8/56.4/27.7	38.9/55.5/34.7/17.7
HawkEye	49.3/72.5/58.3/28.8	39.1/55.9/34.7/17.9

Table 8: Fine-tuned performance of temporal video grounding. Four metrics reported are mIoU and R@IoU >0.3/0.5/0.7, where the higher the better. ActivityNet-Captions is again not applicable for TimeChat as mentioned in Table 5. The SOTA specialists are (Moon et al., 2023; Nan et al., 2021). †: Our implementation.

amount of time-aware samples from InternVid-G is effective, some task-specific approaches such as contrastive learning and finer-grained visual-text alignment might be required to further improve grounding performance. The results also show that **our verbal reference method is easy for LLMs to learn and follow, as VideoChat2 without specifically instruction-tuned on time-related data have already outperformed TimeChat after fine-tuning on only thousands of examples.**

C.4 Inference Speed

We compare the inference speed of zero-shot HawkEye and TimeChat on Charades-STA. We first preprocess the videos into densely-extracted frames, and directly read the frames from hard disk during inference. We use AMD EPYC 7763 CPU and NVIDIA A100 GPU, and set batch size as 1. We use the default prompt of TimeChat and HawkEye respectly.

Though a maximum of 2 rounds of recursive grounding are conducted to achieve optimal performance, HawkEye only takes an average of **1.54 seconds** for each example, while TimeChat takes an average of **1.63 seconds** as it needs to load and preprocess more frames (96 frames for TimeChat), and its prompt is much longer to record the prcecise and miscellaneous timestamps of all the frames. Note that the speed may vary depending on the performance of hard disk, CPU and GPU, but the general conclusion is it takes about the same amount of time to run HawkEye for two rounds or TimeChat for 1 round, which confirms the efficiency of the verbal reference method used by HawkEye even it uses recursive grounding that requires more than one round of model inference.

D Case Studies

We provide more examples of comparison between HawkEye and VideoChat2 in the appendix. Incorrect contents are marked in red. Hawkeye has similar abilites on answering factual questions, but performs much better on time-aware questions and understanding long-form videos.

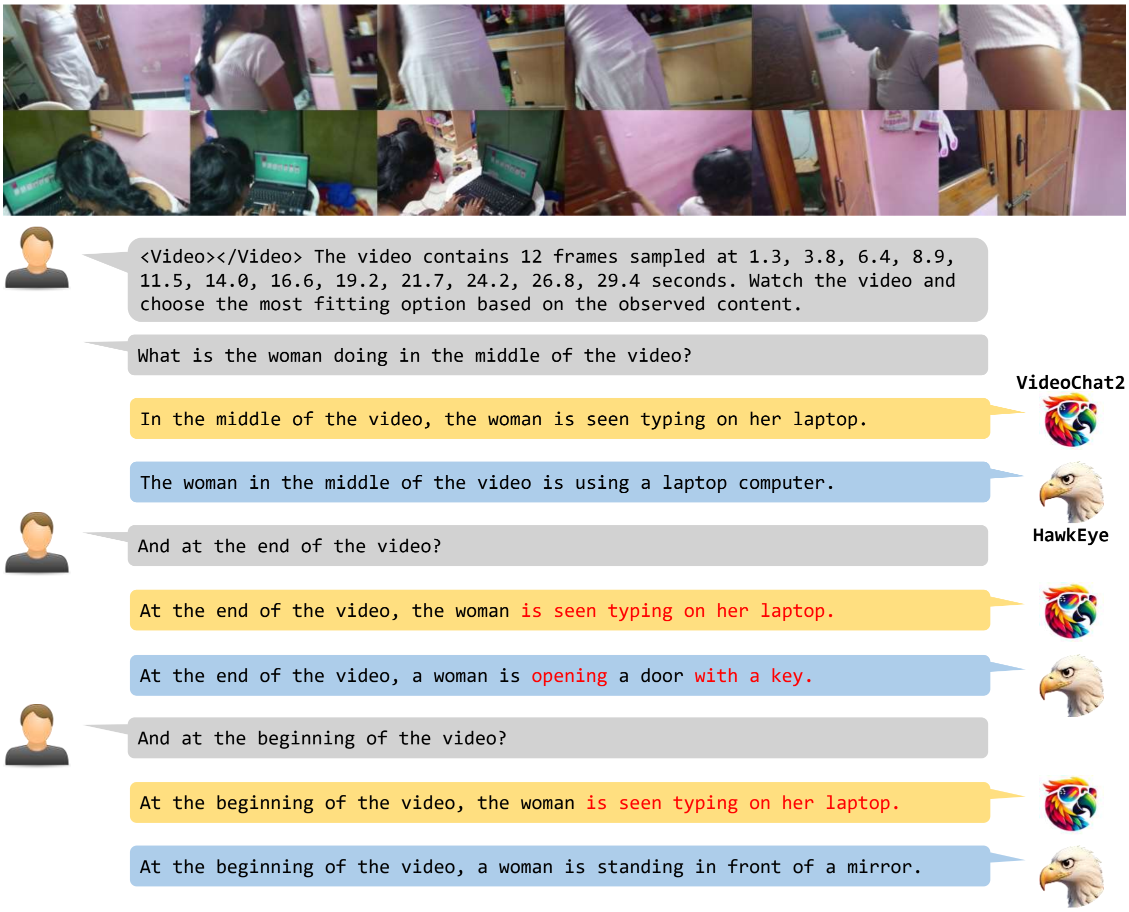

Figure 10: A Conversation using VideoChat2 and HawkEye about a daily activity video.

<Video></Video> The video contains 12 frames sampled at 5.4, 16.3, 27.3, 38.2, 49.1, 60.0, 70.9, 81.8, 92.7, 103.6, 114.5, 125.4 seconds. Watch the video and choose the most fitting option based on the observed content.

What is this video about?

VideoChat2: The video is about a weather forecast.

HawkEye: The video is about a weather forecast for the next few days.

What will the weather be like according to the video?

VideoChat2: According to the video, the weather will be sunny.

HawkEye: The video shows a forecast for the next few days, including a tropical storm and a heavy rain event.

At which segment of the video is the temperature shown, and how will it be?

VideoChat2: The temperature is shown in the segment that starts at 00:00 and ends at 00:03. The temperature is expected to be 77 degrees Fahrenheit.

HawkEye: The temperature is shown at the end of the video, and it is expected to be around 70 degrees.

Figure 11: A Conversation using VideoChat2 and HawkEye about a news video.

<Video></Video> The video contains 20 frames sampled at 3.7, 11.2, 18.7, 26.2, 33.7, 41.2, 48.7, 56.2, 63.7, 71.2, 78.7, 86.2, 93.7, 101.2, 108.7, 116.2, 123.7, 131.2, 138.7, 146.2 seconds. Examine the video and make a decision based on the content presented in the footage.

During which part of the video does 'the woman shows different dishes' occur, at the beginning, the middle or the end of the video?

VideoChat2: The woman shows different dishes at the end of the video.

HawkEye: The woman shows different dishes at the beginning of the video.

What happens at the end of the video?

VideoChat2: At the end of the video, the woman is seen eating a meal.

HawkEye: At the end of the video, a person is shown cutting a piece of food on a plate.

Figure 12: A Conversation using VideoChat2 and HawkEye about a vlog video.

<Video></Video> The video contains 20 frames sampled at 3.7, 11.2, 18.7, 26.2, 33.7, 41.2, 48.7, 56.2, 63.7, 71.2, 78.7, 86.2, 93.7, 101.2, 108.7, 116.2, 123.7, 131.2, 138.7, 146.2 seconds. Examine the video and make a decision based on the content presented in the footage.

What is this video about?

VideoChat2: The video is about a woman who is cooking noodles and talking to the camera.

HawkEye: The video is about a girl who is using a laptop and a phone to make a video call.

What happens at the beginning of the video?

VideoChat2: At the beginning of the video, a woman is seen talking to the camera while holding a bowl of noodles. She then proceeds to cook the noodles in a pot of boiling water.

HawkEye: At the beginning of the video, a girl is shown with a cat on her phone.

In which part of the video does a pot of noodles appear?

VideoChat2: A pot of noodles appears in the video when the woman is cooking noodles in a pot of boiling water.

HawkEye: A pot of noodles appears in the video at the end.

Figure 13: A Conversation using VideoChat2 and HawkEye about a vlog video.