TimeLens: Rethinking Video Temporal Grounding with Multimodal LLMs

Supplementary Material

A. Conclusion

In this work, we presented TimeLens, a systematic investigation into building MLLMs with robust video temporal grounding capabilities. On the data front, we first exposed severe quality issues in existing VTG benchmarks. Through meticulous manual re-annotation guided by strict quality criteria, we created TimeLens-Bench, a reliable evaluation suite that dramatically reshapes model rankings and provides trustworthy evaluation for future research. We also developed an automated re-annotation pipeline for noisy training data, yielding TimeLens-100K, a large-scale, high-quality training dataset. On the algorithmic front, our comprehensive exploration yielded several key insights, which culminate in TimeLens models, a family of MLLMs with state-of-the-art VTG performance among open-source models and even surpasses leading proprietary models like GPT-5 and Gemini-2.5-Flash. By open-sourcing our code, data, and models, we hope TimeLens can serve as a strong foundation for building MLLMs with stronger temporal video grounding capability.

B. Details of TimeLens-Bench

In this section, we provide details on our proposed TimeLens-Bench, a comprehensive, high-quality evaluation benchmark for video temporal grounding (VTG), comprising refined versions of three mainstream benchmarks: Charades-TimeLens, ActivityNet-TimeLens, and QVHighlights-TimeLens.

Source datasets. We construct our TimeLens-Bench based on Charades-STA [11], ActivityNet Captions [25], and QVHighlights [27]. For Charades-STA and ActivityNet Captions, we utilize their test splits, while for QVHighlights, we use the validation split, as its test split annotations are not publicly available and prior works [18, 37] also adopt the validation split. Since the test set of ActivityNet Captions is excessively large, resulting in prohibitively high evaluation cost, we uniformly partition videos based on their duration and sample an equal number of videos from each duration bin. This yields a subset with a video count comparable to Charades-STA and QVHighlights, while maintaining a balanced distribution of video durations. Although QVHighlights was originally annotated for both video temporal grounding and video highlight detection, our work focuses exclusively on the temporal grounding task.

Detailed Statistics. In Tab. 4, we present detailed statistics of our TimeLens-Bench and its source dataset counterparts. For each source dataset, annotations with high-quality queries had their corresponding temporal segments refined. For the remaining annotations with low-quality queries, we either revised the queries or rewrote them entirely. A small fraction of queries that were deemed unfixable were subsequently discarded. Overall, TimeLens-Bench comprises a total of 4,279 videos with an average duration of 107.8 seconds, and 9,404 annotations.

Dataset	# Videos	Avg. Duration	# Annotations	# Rewritten Queries	# Refined Time Segments	Domain
Charades-STA [11]	1334	29.5	3720	-	-	Daily Life
Charades-TimeLens	1313	29.6	3363	2467	896	Daily Life
ActivityNet Captions [25]	4885	118.1	17031	-	-	Activity
ActivityNet-TimeLens	1455	134.9	4500	3137	1363	Activity
QVHighlights [27]	1529	149.6	1542	-	-	Mixed
QVHighlights-TimeLens	1511	149.6	1541	859	682	Mixed
Total (TimeLens-Bench)	4279	107.8	9404	6463	2941	Mixed

Table 4. Statistics of the datasets in our proposed TimeLens-Bench, compared against their original versions (Charades-STA [11], ActivityNet Captions [25] and QVHighlights [27]).

B.1. Evaluation Metrics

We evaluate model performance on TimeLens-Bench using four standard metrics: Recall@1 at IoU thresholds of 0.3, 0.5, and 0.7 (denoted as R1@0.3, R1@0.5, R1@0.7), and mean Intersection over Union (mIoU).

- **mIoU** is defined as the average of the temporal Intersection over Union (IoU) scores between the predicted and ground-truth segments across all test samples.
- **R1@**m measures the percentage of samples for which the temporal IoU of the prediction exceeds a given threshold m.

While TimeLens-Bench can be treated as a single unified benchmark for computing the aforementioned metrics, we compute and report metrics **separately** on its three constituent benchmarks to enable a more fine-grained analysis of model performance across different domains. We encourage future works to adopt this evaluation protocol.

C. More Implementation Details.

C.1. Preliminaries for RL

Thinking-based vs. Thinking-free RLVR.. We formalize the distinction between thinking-based and thinking-free RLVR paradigms using GRPO [47] as the reinforcement learning algorithm.

In the task of video temporal grounding, given a video v and query q, the model generates a response y. For GRPO training, for each input pair (v, q) in the training set D, we sample a group of G responses $\{y^{(i)}\}_{i=1}^{G}$ from the policy π_θ, compute their rewards $\{r(y^{(i)})\}_{i=1}^{G}$, and optimize the policy

Training Data	Charades-TimeLens				ActivityNet-TimeLens				QVHighlights-TimeLens			
	R1@0.3	R1@0.5	R1@0.7	mIoU	R1@0.3	R1@0.5	R1@0.7	mIoU	R1@0.3	R1@0.5	R1@0.7	mIoU
Original Noisy Data	52.6	30.4	14.0	35.6	45.0	29.5	16.0	31.3	61.3	46.1	29.1	44.6
TimeLens-100K	**70.0**	**53.9**	**28.1**	**48.3**	**57.9**	**46.3**	**30.5**	**43.1**	**73.0**	**62.2**	**46.1**	**56.7**

Table 5. **Ablation on training data.** Our TimeLens-100K training set significantly improves model performance, validating its enhanced quality.

Model	Charades-TimeLens				ActivityNet-TimeLens				QVHighlights-TimeLens			
	R1@0.3	R1@0.5	R1@0.7	mIoU	R1@0.3	R1@0.5	R1@0.7	mIoU	R1@0.3	R1@0.5	R1@0.7	mIoU
Qwen2.5-VL-3B [3] (Baseline)	51.3	30.5	14.2	33.9	25.3	18.2	9.4	18.4	27.4	19.3	10.6	20.9
+TimeLens	<u>63.5</u>	<u>48.1</u>	<u>23.9</u>	<u>43.3</u>	<u>56.6</u>	<u>44.9</u>	<u>27.4</u>	<u>41.2</u>	<u>71.9</u>	<u>58.9</u>	<u>37.8</u>	<u>52.9</u>
Qwen2.5-VL-7B [3] (Baseline)	59.7	37.8	16.6	39.3	44.1	31.0	16.1	31.4	41.5	27.8	15.2	31.6
+TimeLens	**70.5**	**55.6**	**28.4**	**48.8**	**62.8**	**51.0**	**32.6**	**46.2**	**74.1**	**62.7**	**43.1**	**56.0**

Table 6. **Results across different model sizes.** The best and second-best results are highlighted in **bold** and <u>underlined</u>, respectively.

to maximize the relative advantage within the group:

$$\mathcal{L}_{\text{GRPO}} = -\mathbb{E}_{(v,q)\sim\mathcal{D}}\mathbb{E}_{y^{(i)}\sim\pi_\theta}\left[A^{(i)}\log\pi_\theta(y^{(i)}|v,q)\right], \tag{1}$$

where the advantage is computed as:

$$A^{(i)} = r(y^{(i)}) - \frac{1}{G}\sum_{j=1}^{G} r(y^{(j)}). \tag{2}$$

The key distinction between the two paradigms lies in the response structure and reward computation. In **thinking-based RLVR**, following the "think-then-answer" approach [16], the response consists of two parts:

$$y = [y_{\text{thinking}}, y_{\text{answer}}], \tag{3}$$

where y_{thinking} represents the explicit reasoning process and y_{answer} contains the predicted temporal segment $\hat{S} = (\hat{t}_{\text{start}}, \hat{t}_{\text{end}})$. The reward function combines accuracy and format compliance:

$$r(y) = r_{\text{acc}}(y_{\text{answer}}) + r_{\text{format}}(y), \tag{4}$$

where $r_{\text{acc}}(y_{\text{answer}}) = \text{IoU}(\hat{S}, S^*)$ measures the temporal intersection-over-union with the ground truth segment S^*, and $r_{\text{format}}(y)$ ensures proper output formatting following the "think-then-answer" structure.

In contrast, our **thinking-free RLVR** directly generates the answer without explicit reasoning:

$$y = y_{\text{answer}}, \tag{5}$$

with a simplified reward based solely on grounding accuracy:

$$r(y) = r_{\text{acc}}(y) = \text{IoU}(\hat{S}, S^*). \tag{6}$$

As shown in Tab. 3, the thinking-free paradigm eliminates the need for explicit reasoning generation and format reward engineering, leading to simpler mplementation, faster training and inference, and superior performance.

Difficulty-aware Sampling. To formalize difficulty-aware sampling [21, 54, 57], we first perform offline inference with the model to be trained on the training dataset $\mathcal{D} = \{(v_i, q_i, S_i^*)\}_{i=1}^N$. For each sample, we obtain the predicted segment $\hat{S}_i$ and compute the difficulty estimate as:

$$d_i = 1 - \text{IoU}(\hat{S}_i, S_i^*), \tag{7}$$

where higher values indicate more challenging samples for the current model.

We then compute sampling weights for each sample based on its difficulty. Following [54, 57], we employ Gaussian sampling to construct a training subset where samples with difficulty around a target mean μ are more likely to be selected. Let $g(d; \mu, \sigma^2)$ denote the target Gaussian distribution:

$$g(d;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(d-\mu)^2}{2\sigma^2}\right). \tag{8}$$

To ensure that samples with difficulty d are selected with probability proportional to $g(d)$, we compute the sampling weight for each sample i as:

$$w_i = \frac{g(d_i;\mu,\sigma^2)}{\hat{p}(d_i)}, \tag{9}$$

where $\hat{p}(d_i)$ is the empirical density of samples with difficulty d_i in the original dataset. This density correction ensures that the difficulty distribution of the sampled subset follows the target Gaussian distribution, rather than being biased by the original difficulty distribution in $\mathcal{D}$.

By varying the mean μ of the Gaussian distribution, we obtain training sets with different average difficulty levels and conduct RLVR training on each independently to evaluate the impact of sample difficulty on final model performance in Sec. 5.3.

Model	Charades-TimeLens				Charades-STA			
	R1@0.3	R1@0.5	R1@0.7	mIoU	R1@0.3	R1@0.5	R1@0.7	mIoU
Proprietary Models								
GPT-4o [23]	60.6	44.5	23.5	41.8	51.6	27.9	11.7	34.7
GPT-5 [42]	59.3	42.0	22.0	40.5	39.7	18.3	6.2	28.4
Gemini-2.0-Flash [8]	66.4	53.5	27.1	46.7	55.6	29.0	9.5	35.1
Gemini-2.5-Flash [8]	68.7	56.1	30.6	48.6	47.0	21.8	7.1	31.1
Gemini-2.5-Pro [8]	74.1	61.1	34.0	52.8	53.9	25.5	8.8	34.6
Open-Source Models								
VideoChat-Flash-7B [31]	60.2	37.9	17.8	39.7	72.5*	51.4*	26.4*	45.2*
VideoChat-R1-7B [32]	51.9	30.8	11.7	33.7	-	71.7*	50.2*	60.8*
Time-R1-7B [54]	57.9	32.0	16.9	36.6	78.1*	60.8*	35.3*	58.1*
MiMo-VL-7B [9]	57.9	42.6	20.5	39.6	-	-	-	50.0*
Qwen2.5-VL-7B [3] (Baseline)	59.7	37.8	16.6	39.3	59.4	38.2	18.1	43.6*
TimeLens-7B	**70.5**	**55.6**	**28.4**	**48.8**	70.7	39.8	14.5	42.3

Table 7. **Results on our refined Charades-TimeLens and the original Charades-STA benchmark.** * indicates results from the original paper, other results are from our evaluation.

Model	ActivityNet-TimeLens				ActivityNet-Captions			
	R1@0.3	R1@0.5	R1@0.7	mIoU	R1@0.3	R1@0.5	R1@0.7	mIoU
Proprietary Models								
Gemini-2.0-Flash [8]	62.9	54.0	37.7	49.3	50.4	33.2	19.9	36.5
Gemini-2.5-Flash [8]	66.8	57.5	41.3	52.5	51.2	34.7	21.0	37.4
Open-Source Models								
VideoChat-R1-7B [32]	35.0	23.9	11.3	25.0	-	33.3*	16.7*	35.5*
Time-R1-7B [54]	44.8	31.0	19.0	33.1	58.1*	39.0*	21.4*	40.5*
MiMo-VL-7B [9]	49.3	38.7	22.4	35.5	39.3	24.3	12.9	28.1
Qwen2.5-VL-7B [3] (Baseline)	44.1	31.0	16.1	31.4	34.5	20.8	11.2	26.4
TimeLens-7B	**62.8**	**51.0**	**32.6**	**46.2**	53.5	35.2	19.7	37.7

Table 8. **Results on our refined ActivityNet-TimeLens and the original ActivityNet-Captions benchmark.** * indicates results from the original paper, other results are from our evaluation.

C.2. Experimental Setup

Unless otherwise specified, all experiments are conducted using Qwen2.5-VL-7B [3] as the base model. Under the Qwen2.5-VL architecture, every two consecutive video frames are merged during the patch-embedding stage of the vision encoder.

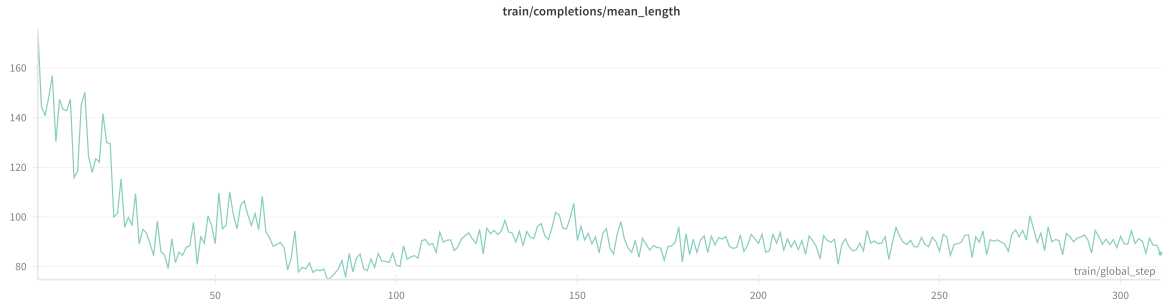
Model Configuration. We sample video frames at 2 FPS. For all ablation experiments, we set the minimum number of tokens per frame (i.e., every two merged frames) to `min_tokens` $= 16$, and the maximum number of tokens for the entire video to `total_tokens` $= 3584$. Under this configuration, the model adaptively adjusts the spatial resolution based on the video's duration and raw resolution. For the final TimeLens models presented in Tab. 1, we scale the resolution budget to `min_tokens` $= 64$ and `total_tokens` $= 14336$. Under this setting, for a 110-second video, the maximum resolution per frame is about 320×320 pixels.

Timestamp Encoding. In Tab. 2 and Sec. 5.1, we experiment with different timestamp encoding methods. For *position-embedding* methods, we directly adopt the original MRoPE implementation from Qwen2.5-VL [3]. For *Visual Overlay*, we render timestamps in red with a font size of 40 pt, overlaid at the bottom-left corner of each frame. For *Non-interleaved Textual Encoding*, an instruction like "This video samples N frames of a T-second video at $t_1, t_2, \ldots$ seconds" is prepended to the prompt.

For *Interleaved Textual Encoding*, timestamps are converted to text with one decimal place retained (*e.g.*, 10.2 seconds), then tokenized and prepended to the tokens of each frame. As described above, when processing videos, every two consecutive frames are merged during the patch-embedding stage of the vision encoder, while for images, each image is duplicated into two identical copies for merging. Since we insert textual timestamp tokens into the original video tokens to form an interleaved visual-text sequence, we treat each video frame as an independent image and duplicate it for processing. This approach allows us to bypass the original MRoPE mechanism entirely, enabling an *isolated* comparison between Interleaved Textual Encoding and MRoPE. Meanwhile, we adopt 1 FPS sampling to ensure the

Model	QVHighlights-TimeLens				QVHighlights			
	R1@0.3	R1@0.5	R1@0.7	mIoU	R1@0.3	R1@0.5	R1@0.7	mIoU
Proprietary								
Gemini-2.0-Flash [8]	76.2	66.4	48.3	60.8	72.1	58.1	41.9	54.9
Gemini-2.5-Flash [8]	78.2	69.4	55.0	64.3	76.9	62.6	46.7	59.1
Open-Source								
MiMo-VL-7B [9]	57.1	42.6	28.4	41.5	59.6	43.3	24.7	41.8
Qwen2.5-VL-7B [3]	41.5	27.8	15.2	31.6	29.2	19.2	11.9	27.4
TimeLens-7B	**74.1**	**62.7**	**43.1**	**56.0**	78.4	63.6	43.7	57.4

Table 9. **Results on our refined QVHighlights-TimeLens and the original QVHighlights benchmark.** All results are from our evaluation.



(a)

<think>
The user is navigating through the settings menu oftheir LG smartphone during the time frame from57.3s to 99.7s. This is evident from the navigationthrough various settings pages and selections onthe screen.
</think>

<think>
The blue tractor driving through the vineyard is akey element of the scene, indicating the harvestingprocess. This vehicle's presence is consistentthroughout the relevant timestamp, clearly showingits action within the vineyard context.
</think>

(b)

Figure 8. **Observations during thinking-based RLVR training.** (a) The model's thinking length gradually decreases during training. (b) The model generates simple thinking content, primarily perception-related, without exhibiting any complex reasoning.

computational cost matches that of other temporal encoding methods using 2 FPS sampling.

Training Configuration. For all training procedures, we freeze the vision encoder while updating all other parameters, and train for one epoch. For supervised fine-tuning (SFT) experiments, we use a batch size of 128 and a learning rate of 1×10^{-5}. For reinforcement learning (RL) experiments, we perform difficulty-aware data sampling with a Gaussian distribution where $\mu = 0.05$ and $\sigma = 0.2$. The training batch size is 8, each prompt samples 8 roll-outs, the learning rate is 1×10^{-6}, and the KL coefficient β is set to 0. We train until we observe that the reward plateaus and then perform early stopping. In practice, this corresponds to approximately 310 training steps ($\sim$2.5K training examples) for Qwen2.5-VL models.

Throughout the development of this work, our experiments were conducted based on Qwen2.5-VL. More recently, the more powerful Qwen3-VL [2] models have been released, so we also validated the effectiveness of our data and recipe on Qwen3-VL. We observed that directly applying RL training on Qwen3-VL fails to yield improvements, likely because Qwen3-VL has undergone large-scale multi-task RL training that includes VTG data, preventing the model from generating rollouts with sufficient diversity on VTG task during our continual RL. Therefore, we first perform a small SFT stage to, in a sense, revert the model back to the "base model" state before RL. This is merely a workaround specific to Qwen3-VL, a model that has already acquired

strong VTG capabilities through an RL stage similar to that proposed in this paper. In the common scenario, our recipes are designed to enhance the VTG capabilities of a "base MLLM", where this trick is not required.

D. More Experimental Results

Results across different model sizes. In Tab. 6, we demonstrate the effectiveness of our proposed design principles across various model sizes. Across base models of varying sizes, TimeLens consistently delivers significant performance gains. Remarkably, despite having fewer parameters, TimeLens-3B substantially surpasses even the larger Qwen-2.5-VL-7B model.

Generalization to an external benchmark. To further validate that our recipe generalizes beyond TimeLens-Bench, we additionally evaluate on VUE-TR, a high-quality human-annotated VTG benchmark introduced concurrently in Vidi [49]. As shown in Tab. 10, TimeLens-7B achieves the best result among the compared models, indicating that the gains from our data curation and algorithmic recipe are not limited to our own refined benchmark suite.

Model	IoU @ VUE-TR (0-200s)
Qwen2.5-VL-7B [3]	36.0
GPT-4o [23]	34.5
Gemini-2.5-Pro [8]	41.6
TimeLens-7B	**45.1**

Table 10. **Results on the VUE-TR benchmark from Vidi [49].** TimeLens-7B achieves the best IoU among the compared models on this external benchmark.

Comparison of results on TimeLens-Bench and original benchmarks. In Tab. 7, Tab. 8, and Tab. 9, we compare the evaluation results of various models on TimeLens-Bench and the original benchmarks. On the original benchmarks, due to data quality issues, open-source models *deceptively* surpass state-of-the-art proprietary models like Gemini-2.5-Pro. On our refined benchmarks, model capabilities are more reliably evaluated, with proprietary models maintaining a significant advantage over open-source models. Remarkably, our TimeLens model substantially narrows the performance gap between open-source and proprietary models.

Results on general video understanding. In Tab. 11, we evaluate TimeLens-7B's general video understanding capabilities on VideoMME [10], the most comprehensive and widely-adopted video understanding benchmark. The results demonstrate that TimeLens-7B maintains the strong general video understanding capability of its base model. This validates that our proposed design principles can effectively enhance video temporal grounding capabilities without sacrificing general-purpose video understanding abilities.

Model	Video-MME			
	Short	Medium	Long	All
Qwen2.5-VL-7B [3] (Baseline)	64.3*	75.2*	**55.1***	65.1†
TimeLens-7B	**66.4**	**76.7**	54.1	**65.7**

Table 11. **Results on the general video understanding benchmark Video-MME [10].** The results show that TimeLens-Qwen2.5-VL-7B maintains strong general video understanding capability while achieving substantial improvements in video temporal grounding. * Results reproduced by us. † Results reported in original paper.

E. Discussion on Thinking-free vs. Thinking-based RLVR

We analyze the possible reasons why thinking-based RLVR underperforms thinking-free methods in our experiments, from both intuitive and empirical perspectives.

Intuitive Analysis.. When manually examining and refining existing grounding datasets, we observe that queries in the grounding task are relatively straightforward, primarily testing the model's perception capability: whether the model can accurately localize the corresponding event in a long video containing massive information. From a human perspective, completing existing grounding tasks indeed relies mainly on intuition and instinct, rather than complex reasoning.

Empirical Observations.. In our experiments, when training with thinking-based RLVR, the model's thinking length gradually decreases and converges to simple, non-reasoning thinking processes, as shown in Fig. 8. This suggests that the model learns to bypass explicit reasoning when it provides no benefit to the task.

Implications and Future Work.. We believe that most samples in existing video temporal grounding data do not require complex reasoning capabilities, but rather demand sufficiently robust long-video perception and localization abilities. Due to the high cost and corresponding low quality of existing data annotation, as well as limitations in current algorithmic designs, existing MLLMs cannot yet achieve this perfectly. Therefore, in this work, we focus on addressing these two fundamental issues. Meanwhile, we believe that certain grounding tasks do require reasoning capabilities, and we leave the exploration of reasoning-intensive VTG scenarios to future work.

F. Additional Discussions

Why do VideoChat-R1 and Time-R1 behave differently across benchmarks?. VideoChat-R1 [32] is trained on single-domain data with relatively limited quality control, and it does not incorporate the training recipes that we find most effective for VTG, such as difficulty-aware sampling and early stopping. As a result, it underperforms even the

Qwen2.5-VL baseline on our refined benchmarks. Time-R1 [54] by contrast, benefits from a stronger RL-oriented training recipe and therefore transfers better to ActivityNet-TimeLens and QVHighlights-TimeLens, where long-video localization is a major challenge. However, its training data is still noisier than ours, which likely limits temporal precision and makes it less competitive on Charades-TimeLens, where videos are shorter but localization accuracy must be much higher.

Training setting for timestamp encoding.. All methods in Tab. 2 are compared after RLVR training rather than in a training-free setting. This controlled setup is important because the temporal encoding method interacts with the optimization paradigm: the final comparison reflects how well each representation supports post-training for VTG, instead of only measuring zero-shot prompting behavior.

Early stopping.. In our single-task VTG setting, reward plateauing is a reliable signal that continued RLVR is unlikely to improve grounding quality and may even degrade it, so early stopping saves computation while preventing over-training. We regard this as a practical recipe rather than a universal rule. In more general multi-task training, a single stopping criterion may be harder to define because different capabilities can peak at different stages, requiring more sophisticated recipes (e.g., remove training data for a single task when reward on this task peaks, while continuing training for other tasks).

G. Annotation Interface and Manual

We present our annotation interface in Fig. 14 and our annotation manual in Fig. 15.

H. Details of Curating TimeLens-100K

As described in Sec. H, we perform automated re-annotation on existing training datasets, resulting in TimeLens-100K, a large-scale, high-quality VTG training set comprising approximately 20K videos and 100K VTG annotations.

We begin by sampling videos from numerous existing VTG datasets, including CosMo-Cap [50], InternVid-VTime [22], DiDeMo [1], QuerYD [41], HiREST [60], etc. These datasets already cover sufficiently diverse video domains. Additionally, we perform uniform sampling based on video duration to ensure sampled videos are approximately uniformly distributed within 0–240 seconds, with a small portion of longer videos included.

Given that most queries in the original annotations either lack clarity and specificity, or describe events that do not exist in the video, we directly use MLLMs for re-annotation. First, we prompt the MLLM to identify distinct events in the video and ensure these events are distributed across different time periods rather than being concentrated in a particular segment. Then, we have the model describe each event to generate queries and output the corresponding timestamps. Finally, we prompt the model to verify the quality of the queries and annotations.

Specifically, we use Gemini-2.5-Pro [8], currently the best-performing VTG model, for re-annotation. The annotation prompt is provided in Fig. 10. Notably, although this prompt appears simple, it is the result of extensive prompt engineering and optimization. We find that a concise and intuitive prompt is sufficient, as the model possesses adequate common sense and reasoning capabilities to understand the task. Overly complex and detailed prompts are unnecessary and can actually degrade annotation quality. During its reasoning process, the model can automatically verify and ensure the uniqueness and uniform distribution of events throughout the video.

As shown in Tab. 5, our training data substantially improves model performance, validating the enhanced quality of our training set. Notably, the construction of our training data is independent of our manual benchmark refinement process, ensuring a fair comparison.

I. Implementation Details for Benchmarking Existing MLLMs

In this section, we present the implementation details for evaluating existing MLLMs on our TimeLens evaluation suite, yielding the results reported in Fig. 2a and Tab. 1.

TimeLens Models. The prompt for training and evaluating TimeLens models is illustrated in Fig. 9. Implementation details are provided in Sec. C.

GPT-5 [42] and GPT-4o [23]. Since GPT models only support multi-image sequences as input, we sample frames from videos at 1 FPS and prepend textual timestamps (*i.e.*, "Frame at 2.5s:") to each frame. As the Azure OpenAI API we use does not support more than 50 images for a single request, we adopt different strategies for videos longer than 50 seconds: for videos lasting 50-80 seconds, we uniformly sample 50 frames; for videos longer than 80 seconds, we sample at 1 FPS and arrange every 4 consecutive frames into a 2×2 grid within a single image, following previous works [24, 63, 64]. For GPT-5, which is a thinking model, we use the default value for the `reasoning.effort` parameter. The evaluation prompt is shown in Fig. 11.

Gemini models [8]. Although Gemini models support audio input, we remove the audio from videos to ensure fair comparison with other vision-only models and maintain consistency with our benchmarks, which features vision-only, audio-free annotations. Following the best practices outlined in the official Gemini API documentation, we prompt the models to output timestamps in "MM:SS" format. For other hyperparameters, we use the default settings: 1 FPS sampling and default `mediaResolution`, which tokenizes each frame into 258 tokens. For thinking models, we do

Prompt for TimeLens Models

You are given a video with multiple frames. The numbers before each video frame indicate its sampling timestamp (in seconds). Please find the visual event described by the sentence '**{query}**', determining its starting and ending times. The format should be: 'The event happens in <start time> - <end time> seconds'.

Figure 9. **Prompt for training and evaluating TimeLens.**

Prompt for Annotating TimeLens-100K

First, design five queries related to video grounding, for example: "When can we observe 'a woman cooking in the kitchen'? Please specify the time range." The queries can be flexible and diverse in form, but the answer to each query must correspond to exactly one time range.

Then provide the answers. The answers should be time ranges (timestamps).

Please provide the queries and answers in JSON format, with 'query' for the question and 'timestamps' for the answer, in the format ["MM:SS", "MM:SS"], for example ["00:59", "01:02"].

Figure 10. **Prompt for annotating TimeLens-100K.**

not impose any limit on the `thinkingBudget` parameter. The evaluation prompt is shown in Fig. 12.

Qwen3-VL [2], Qwen2.5-VL [3] and MiMo-VL [9]. TimeLens models, Qwen2.5-VL-7B, and MiMo-VL share approximately the same model architecture and hyperparameter configurations. Therefore, when evaluating these models, we adopt the same settings to ensure fair comparisons in Tab. 1. Specifically, consistent with Sec. C.2, we sample video frames at 2 FPS and set the resolution budget to $\texttt{min\_tokens} = 64$ and $\texttt{total\_tokens} = 14,336$. For MiMo-VL, we evaluate their best-performing model, MiMo-VL-7B-RL. The evaluation prompt is shown in Fig. 13.

Other Open-source Models. When evaluating Time-R1 [54], VideoChat-Flash [31] and VideoChat-R1 [32], we directly use their original codebases. Please refer to their papers and code repositories for details.

Prompt for Evaluating GPT-5/GPT-4o Models

Prompt for videos shorter than 80 seconds:

You are given multiple frames from a video. Each frame is preceded by its timestamp (e.g., 'Frame at 2.5s:'). Please find the visual event described by the sentence '**{query}**', determining its starting and ending times. Answer in the format 'The event happens in x - y seconds', where x is the start time and y is the end time (x < y).

Prompt for videos longer than 80 seconds:

You are given multiple frames from a video. Every four adjacent frames are stacked into a 2x2 grid (top-left, top-right, bottom-left, bottom-right in timestamp order). Each stacked frame is preceded by its timestamps (e.g., 'Stacked frames at 1.0s, 2.0s, 3.0s, 4.0s:'). Please find the visual event described by the sentence '**{query}**', determining its starting and ending times. Answer in the format 'The event happens in x - y seconds', where x is the start time and y is the end time (x < y).

Figure 11. **Prompts for evaluating GPT-5 and GPT-4o.**

Prompt for Evaluating Gemini Models

Please find the visual event described by the sentence '**{query}**', determining its starting and ending times. Answer in the format 'The event happens in MM:SS - MM:SS'.

Figure 12. **Prompt for evaluating Gemini models.**

Prompt for Evaluating Qwen/MiMo Models

Please find the visual event described by the sentence '**{query}**', determining its starting and ending times. The format should be: 'The event happens in <start time> - <end time> seconds'.

Figure 13. **Prompt for evaluating Qwen3-VL, Qwen2.5-VL and MiMo-VL models.**

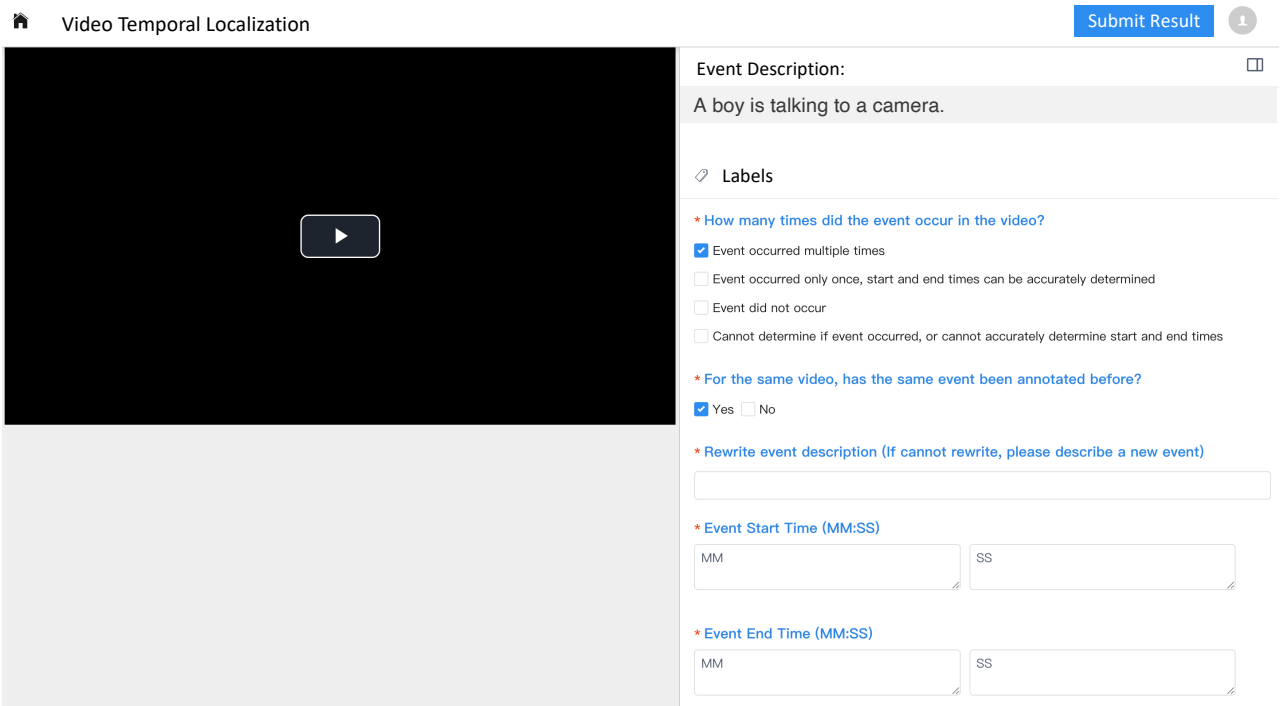


Figure 14. **Illustration of our annotation interface.**

Video Temporal Localization Annotation Guidelines

Task Overview

The annotation page contains a video and a description of a specific event. You need to watch the video to confirm whether the event occurred, how many times it occurred, select the corresponding option, and fill in the subsequent information (rewriting the description, filling in the start and end times of the event, etc.).

Annotation Page Description

The annotation page is shown below.

- **Left side:** The video page where you can play the video.
- **Right side (Red Box):** The "Event Description" area. This provides a sentence describing the event that needs to be annotated.
- **Right side (Green Box):** The options that need to be selected.

Image Here

Instructions: Watch the video, select the corresponding option, and the information required to be filled in will appear below. Fields marked with a red asterisk (*) are mandatory.

Image Here

Important Note on Time Entry

When filling in the "Event Start Time" and "Event End Time" in the image above, you need to move your mouse over the video timeline. You will see the specific time of the current moment in the **white text box** (indicated by the **Red Box** in the diagram below; e.g., 0 min 29 sec).

> **Please Note:** The time displayed on the right side of the video timeline (indicated by the **Green Box** area below) is **NOT** the time of the current moment! **Do not enter this time!**

(a)

Image Here

Scenario A: You selected "Event occurs only once..."

If you selected the second option in the first question ("Event occurs only once..."), and selected "No" in the second question:

1. Please perform appropriate **Polishing** on the event description. Make the description clearer and smoother without changing the semantic meaning of the sentence. You must ensure that after polishing, the start and end times of the event remain unchanged, and the event still occurs only once in the video. (If the original description already meets the requirements, you do not need to fill this in).
2. Then, near the start and end times of the event, pause the video and drag the progress bar to determine the start and end times as accurately as possible. Fill in the **minutes and seconds** for the start and end.

> **Note:** For detailed instructions on "Polishing," please refer to the "Precautions" section of this document.

Scenario B: You selected other options

If you selected any other option in the first question (Multiple times, Did not occur, etc.):

1. You need to **Rewrite** the description of the event (if it is truly impossible to rewrite, choose a **new event** to describe). Ensure the modified event occurs **only once** in the video and that you can accurately determine its start and end times.
   - **Example 1:** "A person walking" occurs multiple times. It can be rewritten as "A man wearing blue clothes is walking," "A person walking on the crosswalk," "A person walking while holding an ice cream," or "A man walking with two women."
   - **Example 2:** "A man walking" did not occur, but there is a woman walking. It can be rewritten as "A woman is walking."
2. Then, near the start and end times of the **rewritten** event, pause the video and drag the progress bar to determine the start and end times as accurately as possible. Fill in the **minutes and seconds** for the start and end.

(c)

Image Here

Detailed Step-by-Step Instructions

First, you need to watch the **entire video** in **mute mode**. Pay attention to observe in which time segments the "Event Description" occurred, and complete the first multiple-choice question on the page:

Image Here

1. If the event occurs multiple times in the video (i.e., occurs in several non-continuous time segments):

- a) You need to select the **"Event occurs multiple times"** option.

2. If the event described by the sentence occurs in only one continuous time segment, and you can clearly determine the start and end times of the event:

- a) You need to select the **"Event occurs only once, and start and end times can be accurately determined"** option.

3. If the event described by the sentence does not occur at all in the entire video:

- a) You need to select the **"Event did not occur"** option.

4. If for various reasons (description is unclear, video is too blurry, etc.) you cannot judge whether the event occurred, or cannot accurately determine the start and end times:

- a) You need to select the **"Unable to determine if event occurred, or unable to accurately determine start and end times"** option.
- b) Then, you need to fill in the specific reason in the "Why unable to determine" field.

Image Here

Next, complete the second multiple-choice question:
If you have already annotated the **same event** for the **same video**, select **"Yes"**; otherwise, select **"No"**.

(b)

> **Note:** For detailed instructions on "Rewriting," please refer to the "Precautions" section of this document.

Final Check

Afterwards, please **watch the entire video again** to check the filled content:

1. Ensure the event described by the modified sentence occurs **only once**.
2. Ensure the start and end times of the event are **correct and accurate**.

After checking, you can proceed to the next annotation.

Annotation Examples

.....

Precautions / Important Notes

I. Instructions regarding "Rewriting"

When rewriting the original sentence, please follow these steps:
.....

II. Instructions regarding "Polishing"

.....

FAQ (Frequently Asked Questions)

Question 1:
.....

Question 2:
.....

(d)

Figure 15. **Illustration of our annotation manual.** Some figures and details are removed for confidentiality and safety reasons.