A. Additional Ablation Studies

Due to space constraint, we provide the rest of the ablation studies here.

A.1. The Video-Centric Branch

Given that *VideoOrion* adopts a two-branch design, we investigate the effectiveness of both branches. In Table 4, we have demonstrated the efficacy of the proposed Object-Centric Branch. Here, in Table 7, we present the performance of the object-only baseline. Relying solely on object tokens results in a performance decline, as only a limited number of tokens—at most 64, or fewer depending on the video—are used to represent the video ($\leq$ 64 v.s 576 for video-branch). This underscores the importance of the video-centric branch in providing contextual information. However, on certain benchmarks (e.g., Perception), the object-only baseline achieves performance comparable to the video-only baseline (with 576 tokens for the video), suggesting that object tokens capture essential information.

Table 7. Ablation study on the Video-Centric Branch.

Model	MVBench	Egoschema	Perception	VideoMME	Avg.
object-only	37.0	34.3	43.3	40.8	38.9
VideoOrion	**44.2**	**44.5**	**46.3**	**46.1**	**45.3**

A.2. The Object-Centric Branch Pretraining Stage

We assess the impact of the additional Object-Centric Branch pretraining stage by comparing it to a randomly initialized Object Branch. In this experiment, the Video-Centric Branch is initialized with the pretrained STC-Connector from VideoLLaMA2. As shown in Table 8, pretraining consistently improves performance across tasks, except for the Perception Test, highlighting the necessity of pretraining object tokens—similar to standard visual tokens—for effective text alignment.

Table 8. Ablation on the Object-Centric Branch pretraining.

Object Pretrain	MVBench	Egoschema	Perception	VideoMME	ActNet
✗	51.2	43.5	**50.5**	46.8	44.3
✓	**52.5**	**51.3**	49.7	**47.6**	**46.3**

A.3. Number of Object Tokens

We also study the impact of different upper limits for the number of object tokens N_{o_i} in Table 9.

A.4. Comparison with Video Encoders

Video encoders share a similar function with our Object-Centric Branch, as they not only capture temporal dynamics but also excel at encoding finer details, leveraging emergent objectness within videos and potentially introducing implicit object modeling. Therefore, in Table 10, we compare *VideoOrion* with baseline models in which the Object-Centric Branch is replaced by two widely used video encoders: VideoMAE [63] and UMT-L [31]. The results demonstrate that *VideoOrion* outperforms both VideoMAE and UMT-L by more than 2% on average.

Table 9. Different upper limit numbers of object tokens.

Max N_{o_i}	MVBench	Egoschema	Perception	VideoMME	Avg.
16	43.7	41.1	44.7	45.3	43.7
32	43.9	40.9	45.8	45.2	44.0
64	**44.2**	**44.5**	**46.3**	**46.1**	**45.3**
80	44.0	40.2	45.4	45.3	43.7

Table 10. Comparison with video-encoder based models.

Model	MVBench	Egoschema	Perception	VideoMME	Avg.
VideoMAE	43.5	38.6	44.4	43.2	42.4
UMT-L	43.7	40.8	45.1	42.3	43.0
VideoOrion	**44.2**	**44.5**	**46.3**	**46.1**	**45.3**

B. Enhenced Temporal-Understanding.

The Object-Branch of *VideoOrion* takes in additional frames to capture the essential object dynamic information in the video. We hypothesize that this inclusion of extra temporal information improves the temporal reasoning capabilities.

In Table 11, we compare *VideoOrion* and *VideoOrion*+ against the baseline models VideoLLaMA2 and VideoLLaMA2.1 on video QA tasks from TemporalBench[7], a benchmark tailored to evaluate fine-grained temporal understanding capabilities. The results demonstrate that our models outperform the baselines, underscoring VideoOrion's superior ability to capture and utilize fine-grained temporal details in videos.

Table 11. Results of zero-shot performance on TemporalBench.

Model	Multi-Binary Accuracy (Acc.)	Binary Accuracy (Acc.)
VideoLLaMA2	15.9	57.4
VideoOrion	**18.2**	**59.0**
VideoLLaMA2.1	17.9	59.5
VideoOrion+	**20.3**	**61.8**

C. More Examples of the Detect-Segment-Track Pipeline

We show additional examples of the object mask lists extracted through the detect-segment-track pipeline in Figure 4. To povide a clearer illustration of the mask pooling mechanism in our model, we resize the masks and map them to the patch level. As can be seen in most instances, the pipeline effectively identifies the salient objects present in videos, ensuring that the resulting object tokens are enriched with clear and meaningful semantics.

D. More Qualitative Results

Additional qualitative examples of *VideoOrion+*, *VideoOrion-Ref* and *VideoOrion-Ref-FT+* are presented in Figure 5. These examples highlight our model's capabilities of capturing interaction details and object dynamics, as well as its enhanced video-based referring capabilities after being trained on this task.

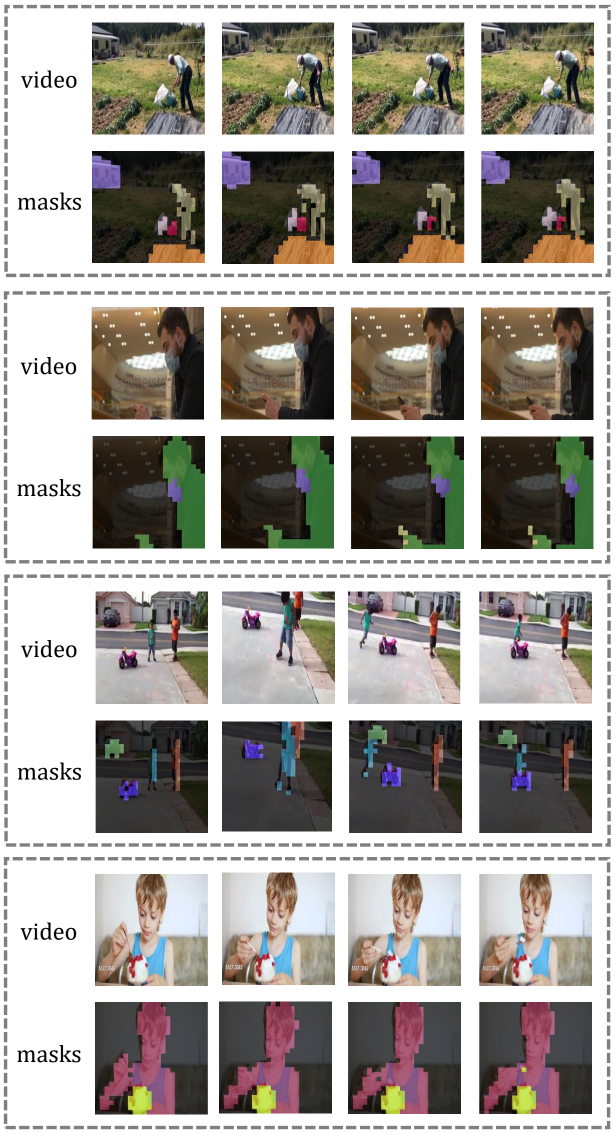
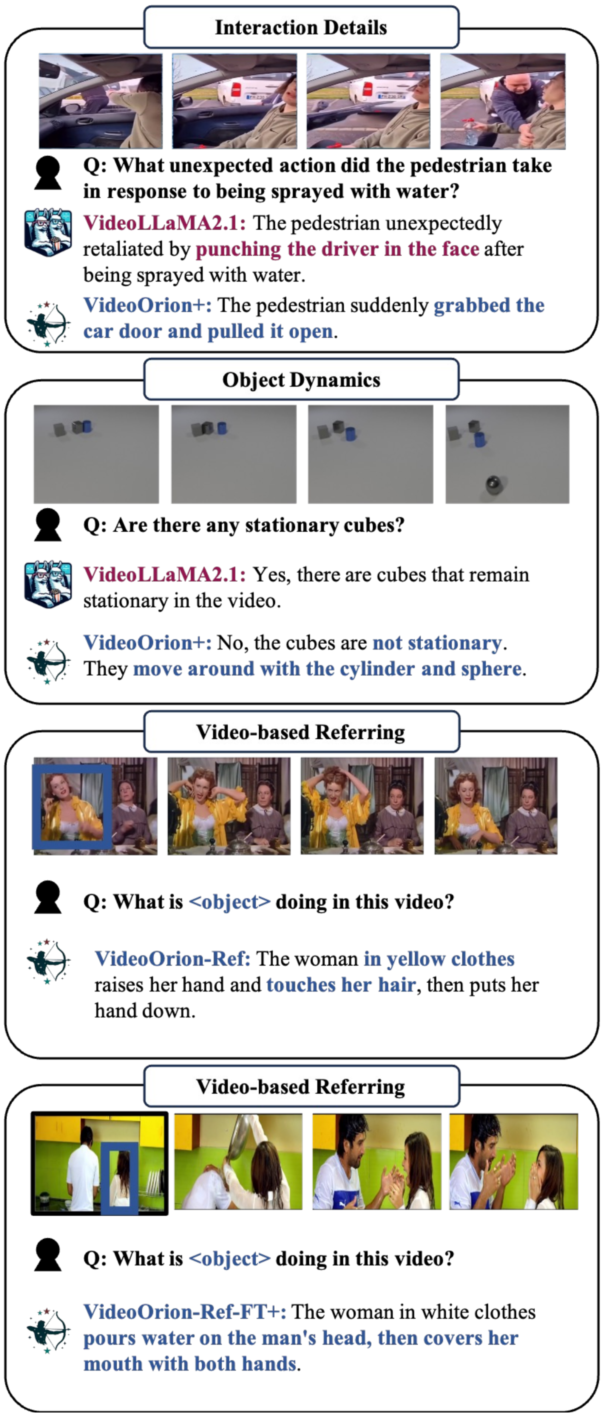


Figure 4. Examples of the detect-segment-track pipeline.



Interaction Details

Q: What unexpected action did the pedestrian take in response to being sprayed with water?

VideoLLaMA2.1: The pedestrian unexpectedly retaliated by **punching the driver in the face** after being sprayed with water.

VideoOrion+: The pedestrian suddenly **grabbed the car door and pulled it open**.

Object Dynamics

Q: Are there any stationary cubes?

VideoLLaMA2.1: Yes, there are cubes that remain stationary in the video.

VideoOrion+: No, the cubes are **not stationary**. They **move around with the cylinder and sphere**.

Video-based Referring

Q: What is <object> doing in this video?

VideoOrion-Ref: The woman **in yellow clothes** raises her hand and **touches her hair**, then puts her hand down.

Video-based Referring

Q: What is <object> doing in this video?

VideoOrion-Ref-FT+: The woman in white clothes **pours water on the man's head, then covers her mouth with both hands**.

Figure 5. Qualitative examples of *VideoOrion+*, *VideoOrion-Ref* and *VideoOrion-Ref-FT+*.

E. Analysis of Failure Cases

E.1. Failures in Detect-Segment-Track Pipeline

One potential limitation is that inaccuracies in the pipeline may hinder the model's understanding and perception abilities. However, the dual-branch design of *VideoOrion* helps alleviate this issue by leveraging context tokens as complementary. As shown in Figure 6, even when the pipeline fails to detect and track the box in a person's hand, *VideoOrion* can still correctly infer the action based on contextual cues.

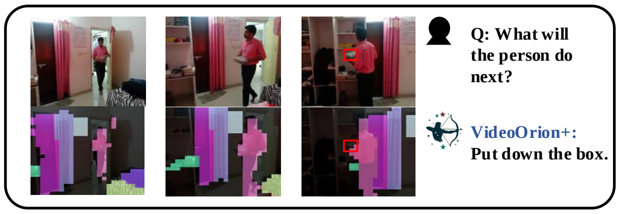


Figure 6. Failure of the detect-segment-track pipeline.

E.2. Failures on Object Attention

Although *VideoOrion* successfully detects and tracks critical objects, it can still occasionally make errors. However, its explicit and disentangled object representation allows for better diagnosis and interpretation of these mistakes.

We analyze a case presented in Figure 7, where a green cylinder in the bottom right corner moves, yet *VideoOrion* incorrectly predicts that no cylinders are moving. By visualizing the attention weights assigned to the object tokens, we observe that the model assigns relatively low attention to the moving cylinder ($O5$) while focusing more on the static grey cylinder ($O4$). This likely explains the misclassification in this instance. This case highlights that while object tokens generally enhance *VideoOrion*'s understanding, misplaced attention on irrelevant objects can still lead to errors.

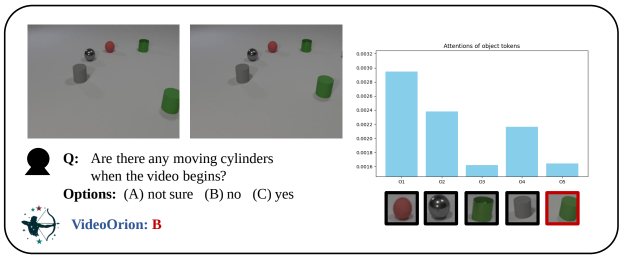


Figure 7. Failure case due to the object attention.

F. Additional Computation Time Induced

As noted in Limitations, our method will increase computation cost due to the additional Detect-Segment-Track pipeline. We report an average computation time for 50 samples in Table 12 and observe a 38.5% increase. We believe this computation cost is acceptable in trade-off of the benefits brought by *VideoOrion*. We also hypothesize that with computation optimization and faster tracking model, these extra time will be negliable in the future.

G. Scaling Effect Observed in *VideoOrion*

In this section, we demonstrate how our model can benefit from data scaling. To evaluate this, we randomly divide the instruction tuning dataset from VideoChat2 into three parts, and we begin by fine-tuning *VideoOrion* using only the Video-LLaVA dataset. Subsequently, we progressively incorporate each of the three parts from VideoChat2 into the training data and demonstrate how performance evolve with scaling of the dataset. As per Figure 8, the performances of *VideoOrion* consistently improve across all four benchmarks, i.e. MVBench, Perception-Test, Video-MME and ActivityNet-QA, with more training data. These results demonstrate *VideoOrion*'s capacity to effectively harness larger datasets, enabling consistent improvements in performance.

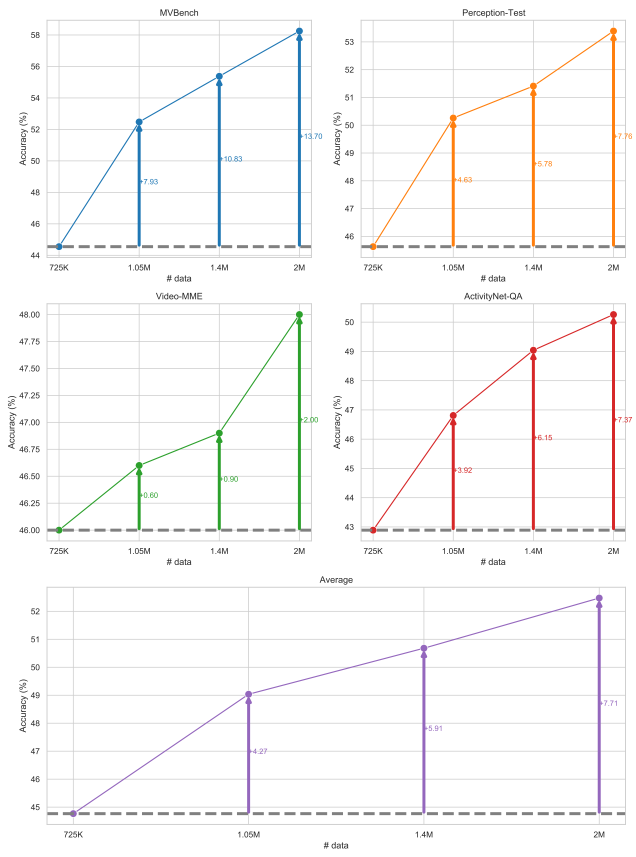


Figure 8. An illustration showcasing how *VideoOrion* benefits from increased training data.

H. Hyperparameters

We report in Table 13 the detailed hyperparameters for *VideoOrion* and *VideoOrion+* used in different training stages.

Table 12. Additional computation time for *VideoOrion*.

	no pipeline	with pipeline	extra (%)
Time/sample	8.27s	11.46s	+38.5%

Table 13. Training hyperparameters for *VideoOrion* and *VideoOrion*+.

Config	**VideoOrion**			**VideoOrion**+		
	Stage1	Stage2	Stage3	Stage1	Stage2	Stage3
Vision Encoder	CLIP(ViT-L/14)			SigLIP(so400m-patch14-384)		
LLM Backbone	Mistral-Instruct-7B			Qwen2-7B		
Frame Number	8	8	8	16	16	16
Input Resolution	336	336	336	384	384	384
Learning Rate	$1e-3$	$1e-4$	$5e-6$	$1e-3$	$1e-4$	$5e-6$
Weight Decay	0	0	0	0	0	0
Warmup Ratio	0.03	0.03	0.03	0.03	0.03	0.03
Learning Rate Schedule	cosine	cosine	cosine	cosine	cosine	cosine
Numerical Precision	bfloat16	bfloat16	bfloat16	bfloat16	bfloat16	bfloat16
Batch Size	256	256	128	256	256	128
LLM Sequence Length	2048	2048	2048	2048	2048	2048
Epoch Number	1	1	1	1	1	1
Max Object Token Number	-	-	64	-	-	64