MSJoE: Jointly Evolving MLLM and Sampler for Efficient Long-Form Video Understanding

Supplementary Material

7. More Implementation Details

7.1. Sampler Architecture

We implement the key-frame sampler with a U-Net with 1×3 convolution layers. This U-Net consists of four down-sampling layers with 32, 64, 128, 256 channels, and four up-sampling layers with reversed channels. To enlarge perception field, the convolution layers are dilated by 1, 2, 4, 8 for the encoders. For the input similarity matrix, query axis is padded to four, denoting a maximum number of four queries (input channels) is processed by the sampler.

7.2. Training Details

For **sampler pre-training**, we use a mini-batch size of 32 to train the sampler for one epoch. We use Adam optimizer with a fixed learning rate of 1e-5. Unlike regular single-label classification tasks, which could directly obtain $\log \mathbf{p}(x_{idx})$, our task is a subset selection without replacement. If we simply do $\mathbf{p} = \text{softmax}(\mathbf{s})$ ($\mathbf{s}$ is the output scores of the sampler) and sample K frames, the log probability would be biased, as $\mathbf{p}$ converges to K equal peaks with $(N_f - K)$ zeros. This leads to suboptimal loss $K \times \log(1/K)$ while treating all frames *equally*. Due to the *unordered* nature of frame selection, we employ iterative probability calculation where each step conditions on previous selections. Algorithm 1 illustrates our approach for proper probability estimation.

Algorithm 1 Probabilistic Sampling Without Replacement

Require: Scores scores, sample size K
Ensure: Selected indices selected, total log probability $\log p_{\text{total}}$

$N_f \leftarrow$ length(scores)
remaining $\leftarrow$ all true vector of size N_f
selected $\leftarrow []$, log_probs $\leftarrow []$
for $k = 1$ **to** K **do**
    Mask unavailable frames in scores with $-\infty$
    probs $\leftarrow$ softmax(masked scores)
    Sample idx from Categorical(probs)
    Append $\log$ probs(idx) to log_probs
    Append idx to selected
    Set remaining[idx] $\leftarrow$ false
end for
return selected, sum(log_probs)

For **joint RL**, we adopt VERL [21] as our training framework and vllm [8] as the inference backend. Learning rate is set to 1e-6 and 1e-5 for the MLLM and sampler, respectively. The training batch size is 32 and group size $G = 8$. Following the latest RL training paradigms [32], we discard KL Divergency regularization, *i.e.*, $kl\_coef = 0$. All the experiments are conducted on a single machine with eight H20 GPUs. The following code block includes a minimal RL training script with verl [21]:

```
python3 -m verl.trainer.main_ppo \
data.train_files=hard.parquet \
data.train_batch_size=32 \
data.nframe_init=16 \
data.ntoken_init=32 \
actor_rollout_ref.rollout.agent.
    nframe_tool=32 \
actor_rollout_ref.rollout.agent.
    ntoken_tool=256 \
actor_rollout_ref.rollout.agent.
    sample_method=UNet \
algorithm.adv_estimator=grpo \
algorithm.kl_ctrl.kl_coef=0.0 \
actor_rollout_ref.model.path=Qwen2.5-
    VL-7B-Instruct \
actor_rollout_ref.actor.optim.lr=1e-6
    \
actor_rollout_ref.actor.
    ppo_mini_batch_size=8 \
actor_rollout_ref.actor.use_kl_loss=
    False \
actor_rollout_ref.actor.kl_loss_coef
    =0.0 \
actor_rollout_ref.actor.kl_loss_type=
    low_var_kl \
actor_rollout_ref.actor.entropy_coeff
    =0.0 \
actor_rollout_ref.rollout.name=vllm \
actor_rollout_ref.rollout.n=8 \
actor_rollout_ref.rollout.agent.
    single_response_max_tokens=512 \
trainer.n_gpus_per_node=8 \
trainer.nnodes=1 \
trainer.total_epochs=2
```

7.3. Inference Details

During training (rollout) phase, the inference temperature is set to 1.0 and top-p is set to 0.9, and we use greedy decoding/sampling during evaluation to ensure reproduction. These temperature is set for both the MLLM and sampler.

8. Dataset Collection Details

Step 1: Dense Captioning We first collect 2.8k long videos from the Internet, covering diverse domains including movies, documentaries, and sports. To obtain fine-grained textual descriptions, we employ the Gemini-2.5-Flash model to segment each video into semantically coherent scenes and generate captions for each segment. On average, a video is divided into about 20 segments. Gemini is also asked to produce an overall caption summarizing the global context of the entire video. These captions serve as detailed and structured textual annotations that facilitate subsequent question generation.

Step 2: QA Generation For each video, we aim to generate cross-event questions that require reasoning over multiple segments. We randomly select pairs of segment captions and merge them with the video's overall caption. The combined text is then fed to Gemini-2.5-Pro to automatically generate at least one multiple-choice question and its corresponding answer. This process is repeated for different segment combinations, resulting in approximately 20 to 40 QA pairs per video. The questions are designed to require temporal reasoning, comparison, or cause–effect understanding across events rather than within a single segment.

Step 3: QA Filtering The automatically generated QA pairs may contain trivial, ambiguous, or incorrect cases. We therefore design a two-stage filtering mechanism to retain only informative and solvable questions.

Easy QA filter. We uniformly sample 16 frames from each video and prompt Qwen2.5-VL-7B-Instruct to answer the question eight times with randomized decoding. If the model answers correctly in at least seven of eight trials, we label the question as too easy and discard it. This step removes questions that can be answered from superficial visual clues or common sense alone.

Hard or bad QA filter. We then sample 256 frames and again query Qwen2.5-VL-7B-Instruct four times. If the model fails to answer correctly even once, we label the question as too difficult or mismatched with the provided answer and discard it. After this stage, about four high-quality QA pairs remain per video, yielding a total of roughly 9.4k QA pairs over 2.8k videos. We denote this dataset as **LongVideoQA-ALL**.

Step 4: Difficulty Labeling and Subset Selection For reinforcement learning and curriculum evaluation, we further construct a more challenging subset. For each video, we provide all segment captions and the remaining QA pairs to Gemini-2.5-Pro, asking it to rank the relative difficulty of questions within the same video. We then select the top-ranked question as the hardest one. The resulting subset, containing one question per video, is denoted as **LongVideoQA-HARD**. This subset emphasizes reasoning-intensive, cross-event questions that best evaluate long-term understanding.

In summary, our dataset construction pipeline combines large-scale automatic generation with fine-grained filtering and difficulty calibration. The resulting dataset provides both scale and diversity, supporting supervised and reinforcement learning of models like MLLM-Sampler Joint Evolution that require adaptive reasoning over long-form videos.

9. More Experiments

9.1. Ablation Study

Reward Design. As shown in Table 3, we quantitatively evaluate the effect of the informativeness reward used during joint reinforcement learning (RL) and the difficulty-aware reward employed in the sampler pre-training phase.

A comparison between the first two rows reveals that removing the informativeness reward from RL training results in a slight performance degradation. This suggests that the reward encourages the MLLM to produce more informative and specific queries, as opposed to overly generic ones, thereby enabling the sampler to retrieve more relevant key frames for subsequent video understanding tasks.

The last two rows exhibit a more pronounced performance gap: when the difficulty-aware reward is removed, *i.e.*, when the sampler is trained using a simple binary reward scheme (-1/1), the average accuracy decreases by 3.9 points. This substantial drop underscores the importance of a well-calibrated reward signal in the training process.

Table 3. Ablation studies on the Informativeness Reward (IR) and Difficulty-aware Reward (DR) with a fixed input budget of 32 frames. LoVi, VLong, and LVB denote the benchmarks LongVideoBench, VideoMME-Long, and LVBench, respectively.

IR	DR	MLVU	LoVi	VLong	LVB
✓	✓	69.3	60.1	54.1	46.4
✗	✓	69.1	59.8	54.1	46.2
-	✓	68.2	56.8	52.2	44.3
-	✗	62.5	55.4	49.2	38.9

Sparse Preview of Query Reasoning To evaluate the effectiveness of *preview frames* in the first step of MSJoE inference (query generation), we vary the number of input preview frames and measure their corresponding accuracy. These experiments follow the same configuration as all other experiments, with $K = 32$.

The results in Figure 7 show that without any visual cues (#Frames $= 0$), the average accuracy drops significantly

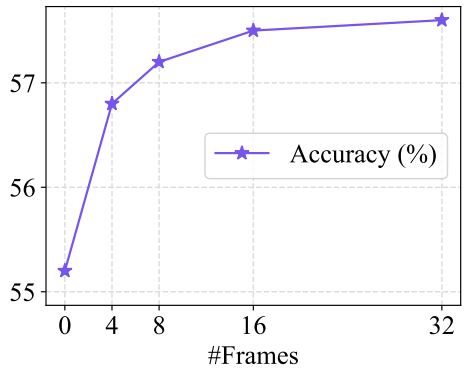

Figure 7. Average accuracy on the four benchmarks under different number of input frames as a preview.

to approximately 55.2%. In contrast, even a sparse preview with only four frames achieves substantially higher overall accuracy. This demonstrates the necessity of providing visual context to generate effective queries.

As the number of preview frames increases, a broader trend emerges: doubling the frame budget from 16 to 32 yields only a marginal accuracy gain of 0.1%, while doubling the computational cost. Therefore, we adopt $K//2 = 16$ frames at the minimum resolution (32 tokens per frame) as the final configuration.

9.2. Efficiency Study

Compared to conventional uniform sampling, our proposed framework introduces several additional inference stages: (i) query reasoning, (ii) similarity matrix computation (FP16), and (iii) frame sampling via U-Net (FP32). To quantify the resulting overhead, we measure the time consumption by iterating over 1,000 samples through these steps. The results in Table 4 show that both similarity matrix generation and the U-Net forward pass incur negligible time with each accounting for less than 1% of the total cost.

Although the overall time increases by 30% compared to vanilla uniform sampling, our question-answering (QA) step is 10% faster, despite using the same number of input tokens as in uniform sampling. This speedup can be attributed to prefix caching in the vLLM backend during the query reasoning phase, which reduces computation in the subsequent answer generation phase due to cache hits. Quantitatively, the prefix cache hit rates for our pipeline and uniform sampling are 17.2% and 11.8%, respectively.

Table 4. Average time (s) cost by the four inference steps on 1,000 samples.

Query	CLIP	UNet	QA	ALL	Uniform
2.809	0.029	0.038	3.353	6.226	3.723

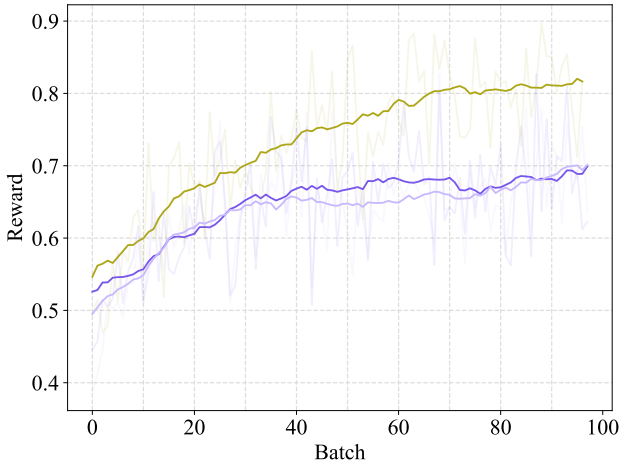

Figure 8. Smoothed reward curves across different training configurations.

9.3. Reward Visualization

We present the training reward curves in Figure 8, which compares three experimental settings: training on the full LongVideoQA-ALL dataset (yellow), training on the more challenging LongVideoQA-HARD dataset with a pre-trained sampler (violet), and training on LongVideoQA-HARD without pre-training (light violet).

As shown in the figure, the reward on the full dataset increases rapidly during the initial training phase. In contrast, progress on the harder dataset is more gradual, reflecting its greater difficulty. While the simpler dataset saturates quickly, the increased complexity of the harder dataset appears to better leverage the model's capacity, ultimately yielding a 2.5% overall performance improvement on our benchmarks.

Comparing the two LongVideoQA-HARD conditions reveals that initializing with a pre-trained sampler significantly stabilizes the early stages of training. This improved stability contributes to enhanced final performance, demonstrating the value of a well-initialized sampler for challenging learning tasks.

10. Prompts

This section details the prompt templates used for query generation and question answering in our framework. The prompt design is motivated by several key considerations: (1) *Explicit role definition* through system prompts establishes clear model behavior; (2) *Structured constraints* ensure outputs conform to downstream processing requirements; (3) *Visual grounding* prioritizes observable content over abstract reasoning; and (4) *Format standardization* enables reliable parsing and integration with our retrieval pipeline. The prompts below implement these principles through precise instruction phrasing, response formatting rules, and example demonstrations.

System Prompt

You are an expert video analyst. Strictly follow the user's instructions.

Query Generation Prompt

Task: Visual Descriptive Queries Generation
Generate **<=4** distinct and visually grounded descriptive queries based on the question and video. These queries will be sent to a CLIP model to retrieve relevant frames for answering the question.
Response Format: List queries as an array within `<queries></queries>` tags.
Strict Guidelines:

- **Visually grounded:** Describe only observable elements
- **Question grounded:** Focus on visual cues that help answer the question
- **Diverse:** Each query must describe a distinct visual perspective
- **Length:** <=4 queries, each under 20 words
- **Format:** Start each query with "An image of ..."
- **Priority:** Sort queries from most to least important for answering

Example Response:

```
<queries>
query 1: An image of ...
query 2: An image of ...
...
</queries>
```

Input:

```
Initial Video: <video>
Question and Options: ...
```

Question Answering Prompt

Task: Question Answering
Focus on the **Current Video**, directly answer the multiple-choice question without any extra text (a single upper-case letter of the option).
Response Format: Place final answer within `<answer></answer>` tags.
Example Response:

```
<answer>A</answer>
```

Input:

```
Current Video: <video>
Question and Options: ...
```