# One Token per Highly Selective Frame: Towards Extreme Compression for Long Video Understanding

**Zheyu Zhang**
University of Illinois Urbana-Champaign
zheyuz5@illinois.edu

**Ziqi Pang**
University of Illinois Urbana-Champaign
ziqip2@illinois.edu

**Shixing Chen**
Amazon Prime Video
shixic@amazon.com

**Xiang Hao**
Amazon Prime Video
xianghao@amazon.com

**Vimal Bhat**
Amazon Prime Video
vimalb@amazon.com

**Yu-Xiong Wang**
University of Illinois Urbana-Champaign
yxw@illinois.edu

## Abstract

Long video understanding is inherently challenging for vision-language models (VLMs) because of the extensive number of frames. With each video frame typically expanding into tens or hundreds of tokens, the limited context length of large language models (LLMs) forces the VLMs to perceive the frames sparsely and lose temporal information. To address this, we explore extreme video token compression towards *one token per frame* at the final LLM layer. Our key insight is that heuristic-based compression, widely adopted by previous methods, is prone to information loss, and this necessitates supervising LLM layers into *learnable* and *progressive* modules for *token-level compression* (LP-Comp). Such compression enables our VLM to digest 2x-4x more frames with improved performance. To further increase the token efficiency, we investigate *frame-level compression*, which selects the frames most relevant to the queries via the internal attention scores of the LLM layers, named *question-conditioned compression* (QC-Comp). As a notable distinction from previous studies, we mitigate the position bias of LLM attention in long contexts, *i.e.*, the over-concentration on the beginning and end of a sequence, by splitting long videos into short segments and employing local attention. Collectively, our combined *token-level* and *frame-level* leads to an e**x**treme compression model for long video understanding, named **XComp**, achieving a significantly larger compression ratio and enabling denser frame sampling. Our XComp is finetuned from VideoChat-Flash with a data-efficient *supervised compression tuning* stage that only requires 2.5% of the supervised fine-tuning data, yet boosts the accuracy from 42.9% to 46.2% on LVBench and enhances multiple other long video benchmarks. Code and Model are available at https://github.com/ZheyuAqaZhang/XComp.

## 1 Introduction

Enabling vision-language models (VLMs) to comprehensively understand long videos remains a critical yet formidable challenge [12, 22, 23, 32, 35, 38, 57, 61, 62, 69, 70]. The sheer volume of visual information challenges current VLMs' inherent context length constraints: as each frame is

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

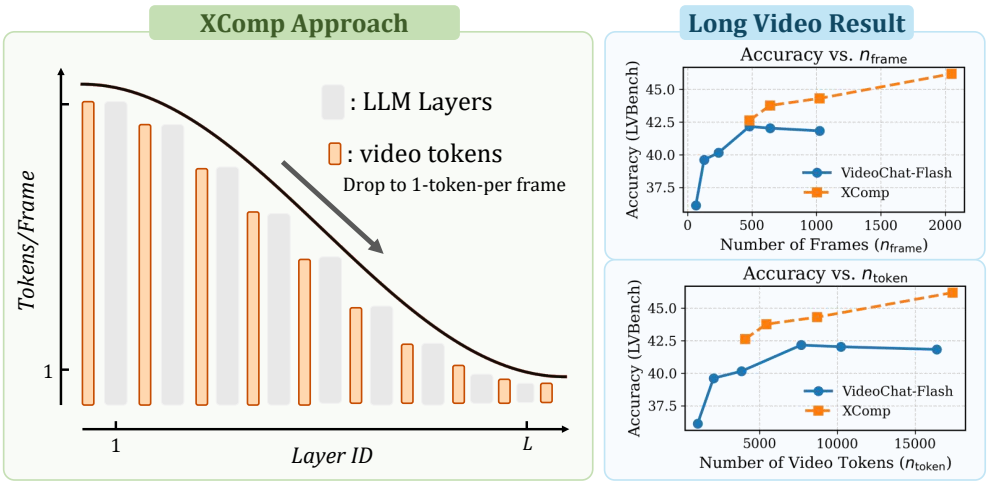


Figure 1: **Left:** We present XComp that explores using the LLM layers to progressively compress the video tokens towards the extreme of *one token per frame*. **Right:** Such capabilities enable the model to better improve itself with denser input frames without significantly increasing video tokens.

encoded into tens or hundreds of tokens, processing more than 1k frames usually exceeds the typical context lengths or computational budgets of the large language model (LLM) layers for the VLMs [11] during both training and inference. Yet, such a framework is still far from processing an hour-long video at more than one frame per second (FPS), easily losing critical temporal information and visual details. Therefore, how to effectively *compress the tokens of video frames while maintaining useful information* becomes the major bottleneck. In this paper, we explore the extreme of this direction and propose an approach to compress each video frame into *one single highly informative token* when reaching the final layer of the LLM in VLMs, thereby unlocking an unprecedented density of frames for VLMs to process.

From this aspect, we aim to achieve extreme compression at the *token level*: condensing the information from tens or hundreds of tokens per frame into a single token. Existing methods predominantly rely on heuristic-based or training-free strategies, such as special token selection [19, 35, 53] and pooling [31, 34]. Despite their reduced token counts, these methods risk discarding crucial visual tokens, as their underlying LLM layers of the VLMs were not supervised to encapsulate the capability of compression, *i.e.*, condense the contexts from the dropped tokens into the kept ones. As a result, our analysis (Table 2) reveals the limitations of such heuristic approaches when reaching high compression ratios. Therefore, our key insight is that the LLM layers should be supervised to compress extensive video data into remarkably fewer representative tokens via an additional *supervised compression tuning* stage, motivating *learnable* compression. To further avoid drastically losing information at large compression ratios, we let the LLM layers *progressively* compress the visual tokens with a smooth schedule. These two combined lead to our "*learnable* and *progressive* compression" (LP-Comp) as in Fig. 1, which marks clear distinctions with previous heuristic methods.

In addition to decreasing the number of tokens per frame, the token efficiency of long video understanding can be further enhanced by selecting the most relevant frames, corresponding to the compression at the *frame level*. Within the internal representation of VLMs, the distinctions of such relevant frames can be intuitively evaluated by the attention scores between the questions and the video tokens [19], where the frames having larger attention scores are preferred. However, a bottleneck under-explored by the previous methods is the internal *position bias* of LLM transformers in long context understanding: the attention heads might assign larger scores to the tokens at the beginning or the end of the sequence [42]. Inspired by local attention [3, 47, 76], under the context of long video understanding, this motivates a splitting of long videos into short video chunks for frame-level compression, where our model inspects the relevance of frames inside each video segment. Such a step effectively utilizes the question information and is referred to as *question-conditioned* compression (QC-Comp).

Combining both *learnable & progressive* compression (LP-Comp) and *question-conditioned* compression (QC-Comp), we propose a VLM named **XComp** that could achieve extreme token efficiency

towards *one-token-per-frame for highly selective frames*. Notably, the compression behavior of the LLM layers can be learned via a data-efficient *supervised compression tuning* stage requiring only 2.5% of the supervised fine-tuning (SFT) data of VideoChat-Flash [35]. More importantly, such token efficiency enables us to sample the frames densely and significantly enhance the performance as the LVBench [59] evaluation in Fig. 1: our method not only achieves better accuracy on a token-for-token efficiency basis, but also keeps improving with the number of input frames increases, while VideoChat-Flash experiences performance drops after a certain frame number.

In summary, we make the following contributions:

1. We introduce **LC-Comp**, a *learnable* and *progressive* framework that supervises LLM layers to compress at the token level, without purely relying on heuristics.
2. We propose **QC-Comp**, a frame-level *question-conditioned* selection method that effectively selects the relevant frames for specific questions.
3. We demonstrate the effectiveness of our token compression towards *each selected video frame as a single token*, drastically increasing the number of frames that can be processed and leading to improved long video understanding accuracy.

## 2 Related Work

**Vision-Language Models for Long Sequence Understanding.** Early Vision-Language Models (VLMs), such as GPT-4V and Gemini-1.5 [49, 58], showcased powerful multimodal reasoning by integrating visual encoders with large language models. Open-source efforts like Llama-Vid [36], IDEFICS [24], VideoChat [34], Video-LLaMA [12], and others [2, 32, 35, 38, 44, 61, 62] have further advanced capabilities, often matching or exceeding proprietary systems on various benchmarks. While effective on static images or short video clips, scaling VLMs to long videos (minutes to hours) introduces a significant challenge. The core difficulty lies in managing the immense volume of visual data, which generates an excessive number of tokens that quickly exceed the typical context windows and computational capacities of the LLM components during both training and inference. Current research addressing long-context VLMs primarily explores two directions: (i) methods that significantly extend the effective context length of the underlying transformer architectures, such as LongVA, LongVILA, LongViTA, and LLaVA-NeXT [11, 55, 75, 77], and (ii) strategies focused on reducing the number of visual tokens fed to the LLM via selection or compression modules [26, 28, 56, 57]. While context extension allows processing more tokens, it often incurs high computational costs. Conversely, existing visual compression techniques, while reducing token count, frequently rely on heuristics or fixed operations, or uniform sampling across the video [2, 12, 32, 34, 35, 75] that struggle to preserve critical temporal and fine-grained information, particularly when pushed to high compression ratios necessary for very long videos. Other orthogonal approaches tackle long videos by processing shorter segments independently and employing retrieval mechanisms [3].

**Video Token Compression in Multimodal LLMs.** Given the computational constraints of processing dense video streams with VLMs, compressing visual tokens is a vital approach. Various methods have been proposed to distill frame sequences into more manageable representations. These include temporal pooling strategies [17, 46], heuristic or saliency-based frame selection often guided by simple visual priors or attention mechanisms [56, 57], and lightweight learned modules or bottlenecks aimed at reducing the token sequence, such as Llama-Vid [36] which explores compressing frames potentially to a single token by leveraging cross-attention with a text query. Relatedly, techniques for dynamic token pruning or merging have been explored for both images and video [5, 13, 26, 29, 45, 52, 68, 70]. However, a critical limitation of many existing approaches, especially those relying purely on heuristics or fixed pooling, is that they are not trained to consolidate the information from dropped tokens into the retained ones within the LLM's representation space. Aggressively applying these methods to achieve the extreme compression ratios necessary for very long videos risks discarding vital visual details, thus hindering deep understanding. Furthermore, while frame selection is intuitive, applying it effectively across extremely long sequences without being hampered by issues like internal LLM position biases remains challenging. Therefore, developing methods that can perform learned and highly efficient visual compression, potentially towards representing each frame with an unprecedentedly low token count while preserving critical information, remains a significant research frontier.

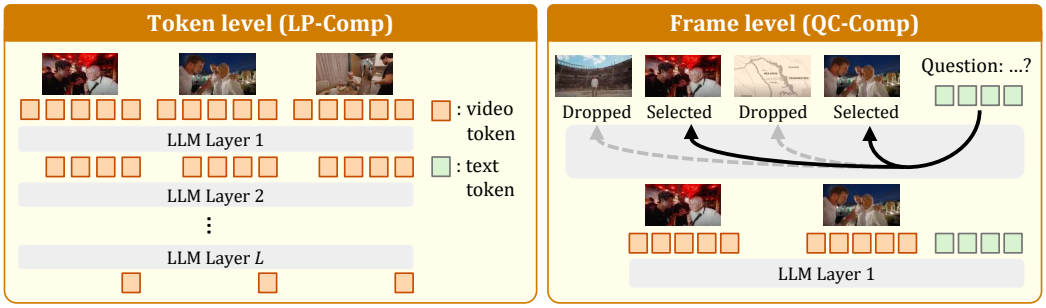


Figure 2: **Overview.** Our XComp comprises of two parts to achieve extreme compression in long video understanding. **(1)** At the *token level*, we propose the supervised compression tuning that enables the LLM to compress every video frame into one compact token in a *learnable and progressive* manner, namely, LP-Comp (Sec. 3.2). **(2)** At the *frame level*, we utilize the internal attention mechanism to select the frames relevant to the questions, which is *question-conditioned* compression, namely, QC-Comp (Sec. 3.3). Both aspects combined lead to our objective of extreme compression in long video understanding: one token per highly selective frame.

# 3 Methodology

Towards the objective of extreme compression, we introduce a novel framework that unifies the compression of long video tokens from two dimensions, as illustrated in Fig. 2:

1. *Token-level*: the **Learnable&Progressive Compression (LP-Comp)** represents each frame using as few as one token, which is learned via training (Sec. 3.2).
2. *Frame-level*: **Question-conditioned Compression (QC-Comp)** selects the frames that are most relevant to the questions based on the attention scores inside the LLM layers, which is employed as an inference-time enhancement (Sec. 3.3).

## 3.1 Preliminaries and Formulation

**VLMs for Video Understanding.** Current VLMs primarily follow the architecture of LLaVA [40] when handling visual inputs: (1) an encoder encodes the raw pixels into a set of visual tokens; (2) a projector, commonly several multi-layer perceptions (MLPs), projects the visual tokens into the dimensions the LLMs; (3) an LLM accepts both the text tokens and projected visual tokens and decodes the output in an auto-regressive manner. This paper concentrates on the compression between the LLM layers and keeps the visual encoder and projector unchanged.

More formally, we consider accepting videos as input, the standard practice is to sample $T$ frames for the VLM to process, denoted as $\{F_t\}_{t=1}^{T}$. Suppose the vision encoder converts each frame into $N^{(1)}$ visual tokens, we then let the LLM's input for the $i$-th frame be $V_i^{(1)} \in \mathbb{R}^{N^{(1)} \times d}$, where $d$ is the LLM's dimension. So the input video tokens for the LLM become a sequence of $T \times N^{(1)}$ tokens, namely, $V^{(1)} \leftarrow [V_1^{(1)}, ..., V_T^{(1)}]$. Denoting the input query as $Q^{(1)}$ with $N_q$ tokens in standard question-answering scenarios, we follow the practice of our baseline model VideoChat-Flash [35] of appending the query $Q^{(1)}$ after the visual tokens $V^{(1)}$, yielding the input sequence to LLM as $X^{(1)} \leftarrow [V^{(1)}, Q^{(1)}]$, a sequence with length $T \times N^{(1)} + N_q$. Then, the LLM begins the auto-regressive generation of new text tokens based on the input sequence $X^{(1)}$.

**Objective: Visual Token Compression.** The computation process of the LLM is commonly dominated by the large volume of visual tokens $T \times N^{(l)}$ in long video understanding. To reduce the cost of LLM layers and permit more input frames, we explore two directions to reduce video token counts: either from the aspect of $N^{(1)}$, the number of tokens per frame, or $T$, the number of frames. **(1)** *Token-level Compression*. Denoting $N^{(l)}$ as the number of tokens per frame at $l$-th transformer layer, our formal objective is to decrease $N^{(l)}$ so that the transformer layers have a gradually more concise set of visual tokens to process, shrinking from $T \times N^{(1)}$ to $T \times N^{(l)}$. We demonstrate such token-level compression in Sec. 3.2. **(2)** *Frame-level Compression*. Another way of decreasing the computation is to convert the frame number $T$ into a smaller $T'$. In the context of long video understanding, this means selecting a subset of relevant frames to process and ignoring the rest. Our

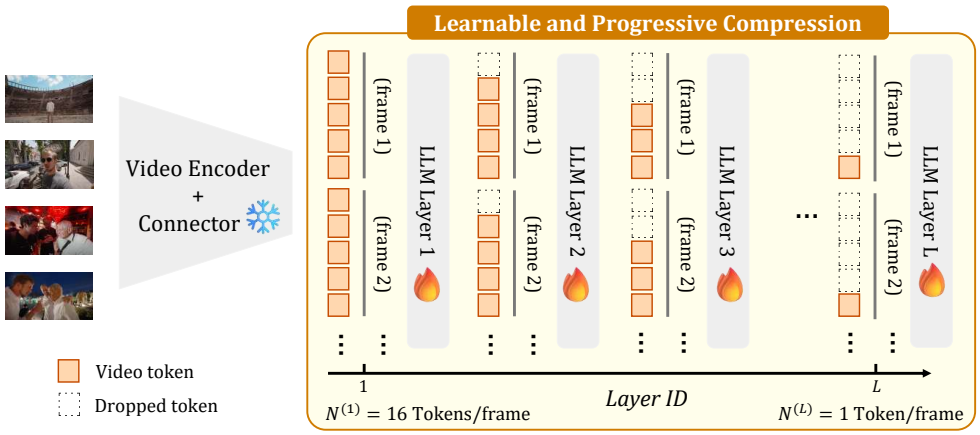


Figure 3: **Learnable and Progressive Compression (LP-Comp).** With *supervised compression tuning*, the LLM layers *learn* to condense the visual tokens *progressively* into a concise set of tokens until reaching the extreme of one token per frame.

Sec. 3.3 presents frame-level compression strategies. Finally, our framework unifies both types of compression to enhance long video understanding.

### 3.2 Learnable and Progressive Token Compression (LP-Comp)

**Objective.** As discussed in Sec. 3.1, the objective of *token-level compression* is to decrease the number of video tokens $N^{(l)}$ than the initial token number $N^{(1)}$. Suppose the LLM contains $L$ layers, we aim at enabling the LLM to compress the tokens so that *each frame is compressed to a single token* at the last layer, namely, $N^{(L)} = 1$. In the case of our baseline VideoChat-Flash [35], where $N^{(1)} = 16$, our target marks the compression ratio of 16.

Although previous studies have demonstrated the redundancy in visual modalities [5], especially for videos [37], such a level of compression is challenging, as modern VLMs like VideoChat-Flash have already utilized the projector for visual token compression in the form of downsampling. As a result, we discover that the heuristic compression approaches, *e.g.*, selecting tokens in a training-free manner, are prone to failure for such a large compression ratio (as in Sec. 4.3, Table 2), whose performance significantly falls behind the baseline without token compression. Therefore, our key insight is to let the LLM layers actively *learn* to condense the tokens into a concise set instead of purely relying on inference-time heuristics for extreme visual token compression.

**Learnable Token Compression.** As shown in Fig. 3, the LLM layers can be trained with the token compression by decreasing the number of tokens during the training stage. A critical design of such token compression is always to preserve the *suffix* tokens of a frame. Concretely, we assume a specific layer $k$ compresses visual tokens from $N^{(k)}$ to $N^{(k+1)}$, where $N^{(k+1)} < N^{(k)}$, then the first $N^{(k)} - N^{(k+1)}$ remaining tokens for a frame are determined to be removed (the dashed boxes in Fig. 3). Such a *suffix-preservation* behavior is compatible with the causal attention of decoder-only LLM layers, as the later tokens in a sequence can absorb the earlier token features instead of the reverse direction. By end-to-end supervising the LLM layers for compression, we observe a significant improvement over heuristics (Sec. 4.3, Table 2).

**Progressive Token Compression.** The effectiveness of learnable compression is determined by the capability of LLM layers. For instance, we could directly conduct extreme one-token-per-frame compression with the first LLM layer for maximum efficiency, *i.e.*, $N^{(2)} = 1$, but this would risk discarding critical visual information. To balance both token efficiency and preserving information, we propose to compress progressively across all the LLM layers. As illustrated in Fig. 3, each LLM layer only decreases a small amount of visual tokens according to a smooth schedule. We empirically find the cosine curve effective for token compression, where the visual token number evolves as,

$$N^{(\ell)} = \left\lceil \frac{N^{(1)} - 1}{2} \cos\left(\frac{\ell}{L}\pi\right) + \frac{N^{(1)} + 1}{2} \right\rceil, \quad \ell = 1, \ldots, L. \tag{1}$$

Our experiments (Sec. 4.3, Table 2) show that smoother compression indeed performs better.

**Data-efficient Supervised Compression Tuning.** We state such continued training of VLMs for the compression capability as *supervised compression tuning* (SCT). It can be learned in a data-efficient way: only using 2.5% of the data samples from VideoChat-Flash's SFT dataset. Besides, our learnable compression also improves the training efficiency by reducing the average number of tokens, accelerating the training speed, and enabling training on more frames.

**Discussion.** Despite its simplicity, we emphasize three critical insights leading to our effective token compression: (1) it is viable to conduct visual token compression between the LLM layers without only relying on the visual encoder and projector; (2) *learnable* LLM layers can grasp the condensing of visual information without only relying on training-free heuristics; (3) *progressive* schedule utilizes all the LLM layers for compression without only relying on a few selected layers.

### 3.3 Question-Conditioned Frame Compression (QC-Comp)

**Objective.** Recalling our objective in Sec. 3.1, this section addresses the compression at the frame level (from $T$ to $T'$). In its simplest form, our frame-level compression is achieved via frame selection, where only the frames relevant to the user's questions are used for the response. According to the prior works in VLMs, reducing such redundancy is beneficial to the performance and also enable us to digest more video frames. To complete such selection, we aim to assign a *relevance score* for each frame reflecting its usefulness to the user's question and then remove the ones with the lowest relevance; thus, this step is named *Question-Conditioned Compression*.

**QC-Comp Overview.** The QC-Comp is employed at the inference time for our model to select the most relevant frames. Similar to the previous works [6, 16, 50, 63], we also utilize the internal representation of the VLM for this objective. Concretely, we calculate the attention scores between the question tokens and the video tokens, which serve as the metric for the contributions of a frame: a larger attention score indicates better relevance, an intuition adopted by representative LLM [66] and video analysis [79]. However, previous approaches utilizing attention score for frame selection exhibit an under-explored issue: the bias of LLM attention for the tokens at the beginning and the end of a sequence, namely "*lost in the middle*" [42], which could overlook the critical information at the middle of videos. To address the above problem, we propose splitting a long video sequence into independent short clips.

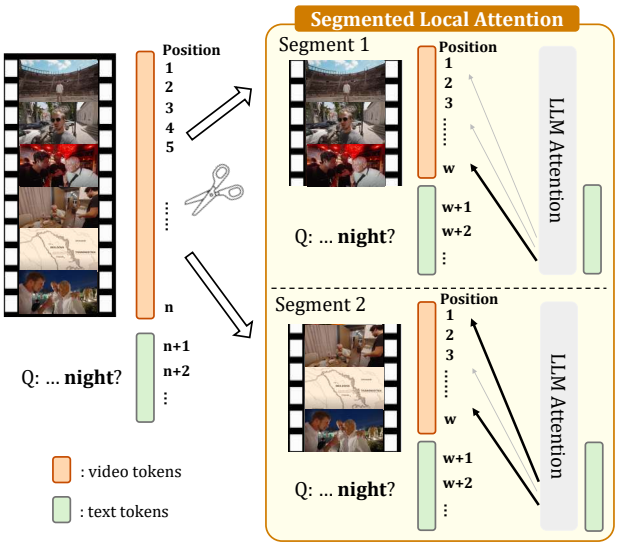


Figure 4: **Question-Conditioned Compression.** To reduce the visual redundancy and improve long video understanding performance, we split a video into individual segments and assign question-conditioned relevance scores to video frames. The frames with lower relevance scores are discarded so that the LLM only concentrates on the informative frames.

**Segmented Local Attention for Relevance Scoring.** The only change of our solution to conventional LLM layers is to split a video token sequence $V^{(l)}$ with $T$ frames into $N_S = \lceil (T-w)/\text{stride} \rceil$ segments, where $w$ is the window size of the segments. Denoting the $N_S$ segments as $[\tilde{V}_1^{(l)}, ..., \tilde{V}_{N_S}^{(l)}]$, we compute the question-conditioned attention within each segment locally by connecting the segment-level video tokens with question tokens, *i.e.*, $[\tilde{V}_i^{(l)}, Q^{(l)}]$. The calculation of these segments is independent and can be conducted in parallel, as shown in Fig. 4. Finally, we record the average attention scores across different attention heads. With these attention scores, we calculate the relevant score for each frame as the average of attention score from any question token to any video token that represents this frame in any segment. Top-k relevant frames among the $T$ frames are selected.

Table 1: **Long video understanding comparison.** Our method XComp built upon VideoChat-Flash-2B achieves state-of-the-art results among 2B-scale models, demonstrating the effectiveness of extreme compression in long video understanding. Note that the performance of proprietary models and 3∼9B-scale VLMs are provided for reference.

| Average Duration | Size | LongVideoBench 473s | MLVU 651s | VideoMME (Long) 2386s | LVBench 4101s |
|---|---|---|---|---|---|
| *Proprietary Models* | | | | | |
| GPT-4V [48] | - | 59.1 | 49.2 | 53.5 | – |
| GPT-4o [49] | - | **66.7** | **64.6** | 65.3 | 30.8 |
| Gemini-1.5-Pro [58] | - | 64.0 | – | **67.4** | **33.1** |
| *3∼9B-Scale VLMs* | | | | | |
| Qwen2.5VL-3B [4] | 3B | 43.3 | 68.2 | – | – |
| mPLUG-Owl3 [73] | 7B | 52.1 | – | 50.1 | 43.5 |
| VideoChat-Flash-7B [35] | 7B | 64.7 | 74.7 | **55.4** | **48.2** |
| Eagle2.5-8B [8] | 8B | **66.4** | **77.6** | – | – |
| Kangaroo [41] | 8B | 54.8 | 61.0 | 46.7 | 39.4 |
| TimeMarker [10] | 8B | 56.3 | – | 46.4 | 41.3 |
| InternVL3-9B [81] | 9B | 62.5 | 70.8 | – | – |
| *2B-Scale VLMs* | | | | | |
| InternVL3-2B [81] | 2B | 55.4 | 64.2 | – | – |
| VideoChat-Flash-2B [35] | 2B | 58.3 | 65.7 | 44.9 | 42.9 |
| XComp | 2B | **59.7** | **66.7** | **45.6** | **46.2** |

**Discussion.** Despite the complexity of duplicating the question token, it is viable to mitigate the bias in long-context LLM attention, resulting in an improved performance as shown in Table 4.

# 4 Experiments

## 4.1 Implementation Details

We fine-tune XComp from VideoChat-Flash-2B [35] for supervised compression tuning, integrating our proposed LP-Comp mechanism. The core components follow the original setup from VideoChat-Flash: we use UMT-L [43] as the visual encoder, token merging with an MLP-based connector, and Qwen2-1.5B [71] as the large language model (LLM). For long videos, we segment them into short clips of 8 frames each. Each clip is compressed into 128 visual tokens, leading to an average of 16 tokens per frame.

Our LP-Comp mechanism introduces a progressive token reduction strategy across LLM layers. Initially, each frame is represented by 16 tokens. After each LLM layer $l$, we compute the target token count $N^{(l)}$ as defined in Equation 1. If the token count from the previous layer $N^{(l)}$ differs from $N^{(l-1)}$, we uniformly drop $N^{(l-1)} - N^{(l)}$ tokens for each frame across the temporal dimension. This process continues across all layers, eventually reducing to a single token per frame at the final LLM layer, *i.e.*, $N^{(L)} = 1$. With LP-Comp reducing the token sequence length during training, our method enables faster training of VLMs on video sequences with more frames. Concretely, we use the mixture of 128–1024 frames per video with up to 8 frames per second in fine-tuning. The fine-tuning data follows the mixture of short-video and long-video instruction tuning formats used in VideoChat-Flash [35], but our supervised compression tuning only requires 2.5% of VideoChat-Flash's dataset. The training process costs around 24 hours on 8×NVIDIA H100 GPUs.

During the inference stage, we utilize QC-Comp to conduct frame selection. When splitting the long video into short segments, each segment contains 64 frames, and we select 512 frames for videos with a duration of less than an hour, and 2048 frames for videos longer than an hour. More details on implementation are in the supplementary materials.

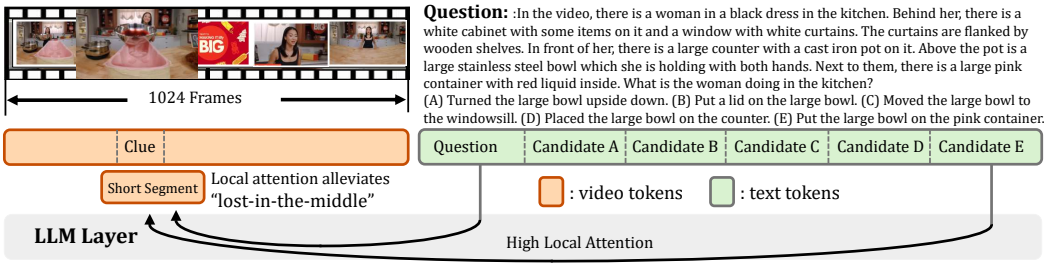


Figure 5: **LongVideoBench case analysis.** XComp leverages QC-Comp to divide a long video into short segments, mitigating attention bias. Local attention highlights key frames relevant to the question and correct answer, enabling effective filtering of irrelevant content.

### 4.2 Main Results on Long Video Understanding

**Benchmarks.** We assess our model on four widely used long video understanding benchmarks: LongVideoBench [64], MLVU [80], LVBench [59], and VideoMME [18] (w/o subtitles). Unlike short-video datasets, they feature extended video durations ranging from several minutes to over an hour, posing greater temporal reasoning and memory challenges. These benchmarks adopt a question-answering format, primarily using accuracy as the evaluation metric.

**Comparison.** We choose the baselines with similar model size of VideoChat-Flash-2B [35] and InternVL3-2B [81], as shown in Table 1. Importantly, our XComp is trained from VideoChat-Flash using only 2.5% of the data, the improvement over VideoChat-Flash suggests that our *one token per highly selective frame* preserves the critical information after compression.

**Leading Performance.** As shown in Table 1, our XComp consistently improves the baseline VideoChat-Flash and outperforms the other VLMs at similar scales, even exhibiting comparable performance to some models at 7B scale. As also demonstrated in Fig. 1, our XComp demonstrates the unique trend of improving performance with more input frames, demonstrating the effectiveness of our extreme compression.

**Qualitative Analysis.** Figure 5 presents an example from LongVideoBench illustrating how XComp leverages QC-Comp to effectively identify frames relevant to the question. In this long video question answering scenario, the question provides a detailed description of the background and inquires about the character's actions. Five answer candidates are also included. To answer the question, QC-Comp divides the video into shorter segments of 64 frames each, mitigating attention biases such as the "lost-in-the-middle" issue. This segmentation allows the model to focus local attention on key frames, which receive high relevance scores with respect to both the question and the correct answer candidate. As a result, XComp is able to discard 512 irrelevant frames and accurately identify the correct answer.

### 4.3 Ablation Study

**Learnable Compression.** The key insight of our XComp is to train the LLM layers to condense the visual tokens. In Table 2, we compare using learnable with training-free token compression. Enabling the LLM layers to learn the compression behavior consistently outperforms the training-free model, supporting the necessity of our learnable compression. Another critical comparison is that when using the heuristic token compression from VideoChat-Flash, which utilizes attention scores to select the tokens, it is worse than the original baseline under our extreme token

Table 2: **Ablation study using LVBench:** progressive v.s. step-wise non-heuristic token compression v.s. heuristic compression from VideoChat-Flash, and learnable v.s. training-free token compression. All the evaluations are conducted with 1024 frames. Both progressive compression and staged compression would gradually compress tokens to 1 token per frame with the same amount of average tokens per frame.

| Method | Training-free | Learnable |
|---|---|---|
| Baseline (w/o compression) | 41.8 | - |
| + Heuristic | 41.1 | - |
| + Step-wise | 38.4 | 42.3 |
| + Progressive | 39.7 | **44.3** |

compression ratio. This is precisely the observation that motivates us to propose the learnable token compression (Sec. 3.2).

**Progressive Compression.** To investigate the importance of progressive token compression, *i.e.*, utilizing all the LLM layers to gradually decreasing the token number, we conduct ablation experiments comparing it with the conventional *step-wise* approach, which performs abrupt token reduction at pre-defined layers as in VideoChat-Flash [35]. As shown in Table 2, progressive token compression consistently outperforms the step-wise compression, indicating that smooth compression schedule is essential for preserving the critical visual information.

Table 3: **Comparison of token compression strategies on LVBench and MLVU.** Both Uniform Drop and Suffix-Preservation follow the same Cosine compression schedule and efficient continual training.

| Strategy | LVBench | MLVU |
|---|---|---|
| Uniform | 43.6 | 64.1 |
| Suffix | 44.3 | 65.2 |

**Compressing to Suffix Tokens.** A critical design choice in our LP-Comp is that the LLM layers should gradually preserve the information to the last token of each frame, namely, the suffix tokens. Such a design is compatible with the causal attention in LLM layers, and we analyze its necessity by comparing it with another intuitive strategy: keeping the tokens uniformly by their positions (implementation details in the supplementary materials). As shown in Table 3, preserving the suffix tokens outperforms its uniform counterpart strategy, supporting the importance of designing the compression strategy aligning with the attention of LLM layers.

**Segmented Local Attention.** To understand the impact of segment-wise attention in QC-Comp, which is proposed to mitigate the position bias of attention in long contexts, we compare it with the conventional way of computing the attention between the question and all frames globally. As shown in Table 4, segment-wise attention consistently leads to better performance across all three long video QA benchmarks LongVideoBench, MLVU, and VideoMME (long). This validates our design choice of computing frame scores locally within segments, effectively reducing positional bias and improving the relevance of retained frames in the early filtering stage.

Table 4: **Comparison of attention strategies in QC-drop.** Segment-wise attention consistently outperforms global attention across benchmarks.

| Attention | LongVideoBench | MLVU | VideoMME (long) |
|---|---|---|---|
| Global | 59.5 | 66.3 | 44.8 |
| Segment Local (Ours) | 59.7 | 66.7 | 45.7 |

**Effect of LP-Comp and QC-Comp.** We perform an ablation study to analyze the impact of the two components in XComp: LP-Comp and QC-Comp. As shown in Table 5, the baseline without any compression achieves a score of 58.3 on LongVideoBench. Adding LP-Comp alone (XComp without QC-Comp) yields a slight improvement to 58.8. When both LP-Comp and QC-Comp are used (full XComp), the performance reaches 59.7, suggesting that selectively removing less relevant content may help retain useful information. These results indicate that both components contribute to the final performance.

Table 5: **Ablation results showing the individual and combined impact of LP-Comp and QC-Comp on LongVideoBench.** Both components contribute to the overall performance gains of XComp.

| | LongVideoBench |
|---|---|
| Baseline (w/o compression) | 58.3 |
| XComp w/o QC-Comp | 58.8 |
| XComp | 59.7 |

Table 6: **The complete table of all ablation studies on LongVideoBench and LVBench.**

| Model Name | LP-Comp | Variants | QC-Comp | Variants | LongVideoBench | LVBench |
|---|---|---|---|---|---|---|
| VideoChat-Flash | No | - | No | - | 58.3 | 42.9 |
| XComp | Yes | Uniform | No | - | 58.2 | 43.6 |
| XComp | Yes | Suffix | No | - | 58.8 | 44.3 |
| XComp | Yes | Suffix | Yes | Global | 59.5 | 45.6 |
| XComp | Yes | Suffix | Yes | Segment Local | **59.7** | **46.2** |

**Compare All Variants Together.** Table 6 shows the complete results for all ablation studies of XComp design, on two benchmarks, LongVideoBench and LVBench. The results are consistent with the studies above.

### 4.4 XComp for LLaVA-Next-Video

We apply XComp to LLaVA-Next-Video [78] to demonstrate the generalizability of our design. Similar to the experiments reported in the paper, XComp effectively enhances model efficiency, allowing a larger maximum number of frames to be processed under limited GPU memory. Following the paper's setup, we use only 2.5% of the training data of LLaVA-Next-Video (i.e., LLaVA-Video-178k).

For a fair comparison, we also evaluate LLaVA-Next-Video with the same amount of fine-tuning data while keeping the model architecture unchanged. As shown in Table 7, XComp improves performance across both benchmarks.

Table 7: **Generalize to LLaVA-Next-Video, a different model from VideoChat-Flash.** XComp consistently improves the efficiency and accuracy of LLaVA-Next-Video on the two long video benchmarks, outperforming both the baseline and fine-tuning.

| Model Version | VideoMME (Long) | LVBench |
|---|---|---|
| LLaVA-Next-Video | 62.7 | 40.6 |
| LLaVA-Next-Video + FT | 61.4 | 41.4 |
| LLaVA-Next-Video + XComp | **63.2** | **43.1** |

## 5 Conclusion

In this work, we propose extreme token compression, named **XComp**, to address the long video understanding problem by alleviating the large number of tokens. At the token level, our XComp framework targets on compressing each frame to *one token*. Observing the failures of heuristic compression adopted by previous methods, we propose the key insight of letting the LLM layers *learn* the *progressive* compression of video tokens. This compression strategy, called LP-Comp, effectively improves the token efficiency and enables denser frame sampling. Our compression at the frame level further enhances the information contained in the tokens by selecting the frames relevant to the user queries. Based on the internal attention scores of LLM layers, we further propose to split the whole video into shorter segments to mitigate the position bias issues. The resulting *question-conditioned* compression, QC-Comp, curates a selective set of frames for the LLM to process. Collectively, our proposed strategies enable XComp to explore the extreme of video token compression via data-efficient fine-tuning and significantly improve the base VideoChat-Flash on multiple long video understanding benchmarks.

**Limitations and Future Work.** Under a limited budget, we primarily explore the extreme token compression with VideoChat-Flash-2B and LLaVA-Next-Video via data-efficient fine-tuning. Therefore, potential future work includes adapting our XComp framework to larger VLMs and potentially integrating the learnable compression into the pre-training stage of the VLM. In addition to the scale and diversity of experiments, another future work lies in training with the question-conditioned frame selection, which could be jointly optimized with the LLM layers to achieve even larger compression ratios than presented in the paper.

**Broader Impacts.** This work presents advancements in vision-language modeling, with a focus on improving long video understanding. While our contributions are primarily foundational, we acknowledge potential societal risks associated with vision-language models, including misuse of disinformation, privacy concerns, and the amplification of biases present in training data. We encourage future work to include fairness assessments and responsible release strategies. Where applicable, safeguards should be considered to mitigate unintended harms.

## Acknowledgments

This work was supported in part by Amazon, NSF under Grants 2106825 and 2519216, the ONR Grant N00014-26-1-2099, and the DARPA Young Faculty Award. This work used computational resources, including Amazon Web Services (AWS), and the NCSA Delta and DeltaAI supercomputers through allocation CIS230012 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program.

## References

[1] Triantafyllos Afouras, Effrosyni Mavroudi, Tushar Nagarajan, Huiyu Wang, and Lorenzo Torresani. Ht-step: Aligning instructional articles with how-to videos. *NeurIPS*, 2023. 25

[2] Kirolos Ataallah, Xiaoqian Shen, Eslam Abdelrahman, Essam Sleiman, Deyao Zhu, Jian Ding, and Mohamed Elhoseiny. Minigpt4-video: Advancing multimodal llms for video understanding with interleaved visual-textual tokens. *arXiv preprint arXiv:2404.03413*, 2024. 3

[3] Kirolos Ataallah, Xiaoqian Shen, Eslam Abdelrahman, Essam Sleiman, Mingchen Zhuge, Jian Ding, Deyao Zhu, Jürgen Schmidhuber, and Mohamed Elhoseiny. Goldfish: Vision-language understanding of arbitrarily long videos. In *ECCV*, 2024. 2, 3

[4] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025. 7, 23

[5] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022. 3, 5, 23

[6] Shyamal Buch, Cristóbal Eyzaguirre, Adrien Gaidon, Jiajun Wu, Li Fei-Fei, and Juan Carlos Niebles. Revisiting the" video" in video-language understanding. In *CVPR*, 2022. 6

[7] Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Huichi Zhou, Qihui Zhang, Zhigang He, Yilin Bai, Chujie Gao, Liuyi Chen, et al. Gui-world: A video benchmark and dataset for multimodal gui-oriented understanding. In *The Thirteenth International Conference on Learning Representations*, 2024. 25

[8] Guo Chen, Zhiqi Li, Shihao Wang, Jindong Jiang, Yicheng Liu, Lidong Lu, De-An Huang, Wonmin Byeon, Matthieu Le, Tuomas Rintamaki, et al. Eagle 2.5: Boosting long-context post-training for frontier vision-language models. *arXiv preprint arXiv:2504.15271*, 2025. 7

[9] Lin Chen, Xilin Wei, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Zhenyu Tang, Li Yuan, et al. Sharegpt4video: Improving video understanding and generation with better captions. *NeurIPS*, 2024. 25

[10] Shimin Chen, Xiaohan Lan, Yitian Yuan, Zequn Jie, and Lin Ma. Timemarker: A versatile video-llm for long and short video understanding with superior temporal localization ability. *arXiv preprint arXiv:2411.18211*, 2024. 7

[11] Yukang Chen, Fuzhao Xue, Dacheng Li, Qinghao Hu, Ligeng Zhu, Xiuyu Li, Yunhao Fang, Haotian Tang, Shang Yang, Zhijian Liu, et al. Longvila: Scaling long-context visual language models for long videos. *arXiv preprint arXiv:2408.10188*, 2024. 2, 3

[12] Zesen Cheng, Sicong Leng, Hang Zhang, Yifei Xin, Xin Li, Guanzheng Chen, Yongxin Zhu, Wenqi Zhang, Ziyang Luo, Deli Zhao, et al. Videollama 2: Advancing spatial-temporal modeling and audio understanding in video-llms. *arXiv preprint arXiv:2406.07476*, 2024. 1, 3

[13] Joonmyung Choi, Sanghyeok Lee, Jaewon Chu, Minhyuk Choi, and Hyunwoo J Kim. vid-tldr: Training free token merging for light-weight video transformer. In *CVPR*, 2024. 3

[14] E Cui, Y He, Z Ma, Z Chen, H Tian, W Wang, K Li, Y Wang, W Wang, X Zhu, et al. Sharegpt-4o: Comprehensive multimodal annotations with gpt-4o, 2024. 25

[15] Dave Epstein, Boyuan Chen, and Carl. Vondrick. Oops! predicting unintentional action in video. *arXiv preprint arXiv:1911.11206*, 2019. 25

[16] Yue Fan, Xiaojian Ma, Rujie Wu, Yuntao Du, Jiaqi Li, Zhi Gao, and Qing Li. Videoagent: A memory-augmented multimodal agent for video understanding. In *ECCV*, 2024. 6

[17] Jiajun Fei, Dian Li, Zhidong Deng, Zekun Wang, Gang Liu, and Hui Wang. Video-ccam: Enhancing video-language understanding with causal cross-attention masks for short and long videos. *arXiv preprint arXiv:2408.14023*, 2024. 3

[18] Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. *arXiv preprint arXiv:2405.21075*, 2024. 8

[19] Jun Gao, Yongqi Li, Ziqiang Cao, and Wenjie Li. Interleaved-modal chain-of-thought. *arXiv preprint arXiv:2411.19488*, 2024. 2

[20] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017. 25

[21] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *CVPR*, 2022. 25

[22] Bo He, Hengduo Li, Young Kyun Jang, Menglin Jia, Xuefei Cao, Ashish Shah, Abhinav Shrivastava, and Ser-Nam Lim. Ma-lmm: Memory-augmented large multimodal model for long-term video understanding. In *CVPR*, 2024. 1

[23] Suyuan Huang, Haoxin Zhang, Yan Gao, Yao Hu, and Zengchang Qin. From image to video, what do we need in multimodal llms? *arXiv preprint arXiv:2404.11865*, 2024. 1

[24] IDEFICS Team. Introducing idefics: An open reproduction of state-of-the-art visual language model. https://huggingface.co/blog/idefics, 2023. Accessed 2025-05-12. 3

[25] Yunseok Jang, Yale Song, Youngjae Yu, Youngjin Kim, and Gunhee Kim. Tgif-qa: Toward spatio-temporal reasoning in visual question answering. In *ICCV*, 2017. 25

[26] Peng Jin, Ryuichi Takanobu, Wancai Zhang, Xiaochun Cao, and Li Yuan. Chat-univi: Unified visual representation empowers large language models with image and video understanding. In *CVPR*, 2024. 3

[27] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 25

[28] Bruno Korbar, Yongqin Xian, Alessio Tonioni, Andrew Zisserman, and Federico Tombari. Text-conditioned resampler for long form video understanding. In *ECCV*, 2024. 3

[29] Sanghyeok Lee, Joonmyung Choi, and Hyunwoo J Kim. Multi-criteria token fusion with one-step-ahead attention for efficient vision transformers. In *CVPR*, 2024. 3

[30] Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L Berg. Tvqa: Localized, compositional video question answering. *arXiv preprint arXiv:1809.01696*, 2018. 25

[31] Bo Li, Kaichen Zhang, Hao Zhang, Dong Guo, Renrui Zhang, Feng Li, Yuanhan Zhang, Ziwei Liu, and Chunyuan Li. Llava-next: Stronger llms supercharge multimodal capabilities in the wild. 2024. 2

[32] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024. 1, 3, 25

[33] Feng Li, Renrui Zhang, Hao Zhang, Yuanhan Zhang, Bo Li, Wei Li, Zejun Ma, and Chunyuan Li. Llava-next-interleave: Tackling multi-image, video, and 3d in large multimodal models. *arXiv preprint arXiv:2407.07895*, 2024. 25

[34] KunChang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*, 2023. 2, 3

[35] Xinhao Li, Yi Wang, Jiashuo Yu, Xiangyu Zeng, Yuhan Zhu, Haian Huang, Jianfei Gao, Kunchang Li, Yinan He, Chenting Wang, et al. Videochat-flash: Hierarchical compression for long-context video modeling. *arXiv preprint arXiv:2501.00574*, 2024. 1, 2, 3, 4, 5, 7, 8, 9, 23, 24, 25, 26

[36] Yanwei Li, Chengyao Wang, and Jiaya Jia. Llama-vid: An image is worth 2 tokens in large language models. In *ECCV*, 2024. 3

[37] Yucheng Li, Huiqiang Jiang, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Amir H Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, et al. Mapsparse: Accelerating pre-filling for long-context visual language models via modality-aware permutation sparse attention. In *ICLR 2025 Workshop on Foundation Models in the Wild*. 5

[38] Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*, 2023. 1, 3

[39] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024. 25

[40] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023. 4

[41] Jiajun Liu, Yibing Wang, Hanghang Ma, Xiaoping Wu, Xiaoqi Ma, Xiaoming Wei, Jianbin Jiao, Enhua Wu, and Jie Hu. Kangaroo: A powerful video-language model supporting long-context video input. *arXiv preprint arXiv:2408.15542*, 2024. 7

[42] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023. 2, 6

[43] Ye Liu, Siyuan Li, Yang Wu, Chang-Wen Chen, Ying Shan, and Xiaohu Qie. Umt: Unified multi-modal transformers for joint video moment retrieval and highlight detection. In *CVPR*, 2022. 7, 23

[44] Ruipu Luo, Ziwang Zhao, Min Yang, Junwei Dong, Da Li, Pengcheng Lu, Tao Wang, Linmei Hu, Minghui Qiu, and Zhongyu Wei. Valley: Video assistant with large language model enhanced ability. *arXiv preprint arXiv:2306.07207*, 2023. 3

[45] Xu Ma, Yuqian Zhou, Huan Wang, Can Qin, Bin Sun, Chang Liu, and Yun Fu. Image as set of points. *arXiv preprint arXiv:2303.01494*, 2023. 3

[46] Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. *arXiv preprint arXiv:2306.05424*, 2023. 3

[47] Xupeng Miao, Shenhan Zhu, Fangcheng Fu, Ziyu Guo, Zhi Yang, Yaofeng Tu, Zhihao Jia, and Bin Cui. X-former elucidator: reviving efficient attention for long context language modeling. In *IJCAI*, 2024. 2

[48] OpenAI. Gpt-4 technical report, 2024. 7

[49] OpenAI. Gpt-4o system card, 2024. 3, 7

[50] Jongwoo Park, Kanchana Ranasinghe, Kumara Kahatapitiya, Wonjeong Ryu, Donghyun Kim, and Michael S Ryoo. Too many frames, not all useful: Efficient strategies for long-form video qa. *arXiv preprint arXiv:2406.09396*, 2024. 6

[51] Jiyang Qi, Yan Gao, Yao Hu, Xinggang Wang, Xiaoyu Liu, Xiang Bai, Serge Belongie, Alan Yuille, Philip HS Torr, and Song Bai. Occluded video instance segmentation: A benchmark. *IJCV*, 2022. 25

[52] Shuhuai Ren, Sishuo Chen, Shicheng Li, Xu Sun, and Lu Hou. Testa: Temporal-spatial token aggregation for long-form video-language understanding. *arXiv preprint arXiv:2310.19060*, 2023. 3

[53] Xiaoqian Shen, Yunyang Xiong, Changsheng Zhao, Lemeng Wu, Jun Chen, Chenchen Zhu, Zechun Liu, Fanyi Xiao, Balakrishnan Varadarajan, Florian Bordes, et al. Longvu: Spatiotemporal adaptive compression for long video-language understanding. *arXiv preprint arXiv:2410.17434*, 2024. 2

[54] Xuyang Shen, Dong Li, Jinxing Zhou, Zhen Qin, Bowen He, Xiaodong Han, Aixuan Li, Yuchao Dai, Lingpeng Kong, Meng Wang, et al. Fine-grained audible video description. In *CVPR*, 2023. 25

[55] Yunhang Shen, Chaoyou Fu, Shaoqi Dong, Xiong Wang, Peixian Chen, Mengdan Zhang, Haoyu Cao, Ke Li, Xiawu Zheng, Yan Zhang, et al. Long-vita: Scaling large multi-modal models to 1 million tokens with leading short-context accuray. *arXiv preprint arXiv:2502.05177*, 2025. 3

[56] Yan Shu, Zheng Liu, Peitian Zhang, Minghao Qin, Junjie Zhou, Zhengyang Liang, Tiejun Huang, and Bo Zhao. Video-xl: Extra-long vision language model for hour-scale video understanding. *arXiv preprint arXiv:2409.14485*, 2024. 3

[57] Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Haozhe Chi, Xun Guo, Tian Ye, Yanting Zhang, et al. Moviechat: From dense token to sparse memory for long video understanding. In *CVPR*, 2024. 1, 3, 25

[58] Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024. 3, 7

[59] Weihan Wang, Zehai He, Wenyi Hong, Yean Cheng, Xiaohan Zhang, Ji Qi, Xiaotao Gu, Shiyu Huang, Bin Xu, Yuxiao Dong, et al. Lvbench: An extreme long video understanding benchmark. *arXiv preprint arXiv:2406.08035*, 2024. 3, 8

[60] Weiyao Wang, Matt Feiszli, Heng Wang, and Du Tran. Unidentified video objects: A benchmark for dense, open-world segmentation. In *ICCV*, 2021. 25

[61] Yi Wang, Kunchang Li, Xinhao Li, Jiashuo Yu, Yinan He, Guo Chen, Baoqi Pei, Rongkun Zheng, Zun Wang, Yansong Shi, et al. Internvideo2: Scaling foundation models for multimodal video understanding. In *ECCV*, 2024. 1, 3

[62] Yi Wang, Kunchang Li, Yizhuo Li, Yinan He, Bingkun Huang, Zhiyu Zhao, Hongjie Zhang, Jilan Xu, Yi Liu, Zun Wang, et al. Internvideo: General video foundation models via generative and discriminative learning. *arXiv preprint arXiv:2212.03191*, 2022. 1, 3

[63] Ziyang Wang, Shoubin Yu, Elias Stengel-Eskin, Jaehong Yoon, Feng Cheng, Gedas Bertasius, and Mohit Bansal. Videotree: Adaptive tree-based video representation for llm reasoning on long videos. *arXiv preprint arXiv:2405.19209*, 2024. 6

[64] Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. Longvideobench: A benchmark for long-context interleaved video-language understanding. *NeurIPS*, 2024. 8

[65] Weijia Wu, Yuzhong Zhao, Zhuang Li, Jiahong Li, Hong Zhou, Mike Zheng Shou, and Xiang Bai. A large cross-modal video retrieval dataset with reading comprehension. *Pattern Recognition*, 157:110818, 2025. 25

[66] Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. Retrieval head mechanistically explains long-context factuality. *arXiv preprint arXiv:2404.15574*, 2024. 6

[67] Junbin Xiao, Xindi Shang, Angela Yao, and Tat-Seng Chua. Next-qa: Next phase of question-answering to explaining temporal actions. In *CVPR*, 2021. 25

[68] Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and Xiaolong Wang. Groupvit: Semantic segmentation emerges from text supervision. In *CVPR*, 2022. 3

[69] Lin Xu, Yilin Zhao, Daquan Zhou, Zhijie Lin, See Kiong Ng, and Jiashi Feng. Pllava: Parameter-free llava extension from images to videos for video dense captioning. *arXiv preprint arXiv:2404.16994*, 2024. 1

[70] Mingze Xu, Mingfei Gao, Zhe Gan, Hong-You Chen, Zhengfeng Lai, Haiming Gang, Kai Kang, and Afshin Dehghan. Slowfast-llava: A strong training-free baseline for video large language models. *arXiv preprint arXiv:2407.15841*, 2024. 1, 3

[71] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. 7

[72] Dongjie Yang, Suyuan Huang, Chengqiang Lu, Xiaodong Han, Haoxin Zhang, Yan Gao, Yao Hu, and Hai Zhao. Vript: A video is worth thousands of words. *NeurIPS*, 2024. 25

[73] Jiabo Ye, Haiyang Xu, Haowei Liu, Anwen Hu, Ming Yan, Qi Qian, Ji Zhang, Fei Huang, and Jingren Zhou. mplug-owl3: Towards long image-sequence understanding in multi-modal large language models. *arXiv preprint arXiv:2408.04840*, 2024. 7

[74] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B Tenenbaum. Clevrer: Collision events for video representation and reasoning. *arXiv preprint arXiv:1910.01442*, 2019. 25

[75] Peiyuan Zhang, Kaichen Zhang, Bo Li, Guangtao Zeng, Jingkang Yang, Yuanhan Zhang, Ziyue Wang, Haoran Tan, Chunyuan Li, and Ziwei Liu. Long context transfer from language to vision. *arXiv preprint arXiv:2406.16852*, 2024. 3

[76] Qingru Zhang, Dhananjay Ram, Cole Hawkins, Sheng Zha, and Tuo Zhao. Efficient long-range transformers: You need to attend more, but not necessarily at every layer. *arXiv preprint arXiv:2310.12442*, 2023. 2

[77] Y Zhang, B Li, H Liu, Y Lee, L Gui, D Fu, J Feng, Z Liu, and C Li. Llava-next: A strong zero-shot video understanding model. 2024. 3

[78] Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. Video instruction tuning with synthetic data. *arXiv preprint arXiv:2410.02713*, 2024. 10, 25

[79] Junbao Zhou, Ziqi Pang, and Yu-Xiong Wang. Rmem: Restricted memory banks improve video object segmentation. In *CVPR*, 2024. 6

[80] Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Shitao Xiao, Xi Yang, Yongping Xiong, Bo Zhang, Tiejun Huang, and Zheng Liu. Mlvu: A comprehensive benchmark for multi-task long video understanding. *arXiv preprint arXiv:2406.04264*, 2024. 8

[81] Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Yuchen Duan, Hao Tian, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025. 7, 8

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See the Methods and Experiments sections, where we provide a detailed description of our proposed method and the corresponding experiments.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See the Conclusion section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: There is no theoretical result.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We show the methodology details with exact numbers with well-defined formulas.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We mentioned the data we used in the paper. These datasets are public. As for the code, we are planning to release the code.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provided the details with specific numbers in appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Restricting to computation budget, we did not run experiments in multiple seeds as well as apply statistical analysis.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provides GPU hours.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: This research adheres to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We mentioned the social impact in appendix.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [No]

    Justification: The paper does not describe safeguards.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the assets used in the paper are either cited in the main paper (models, benchmarks, ...) or in the appendix (data, ...).

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLM is a part of the method in this paper.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.

# A Implementation Details

## A.1 XComp Architecture

**XComp** is based on VideoChat-Flash [35] and uniquely integrates our learnable progressive compression (**LP-Comp**)(Sec. 3.2) and question-conditioned compression (**QC-Comp**)(Sec. 3.3). XComp comes from fine-tuning the VideoChat-Flash-2B model [35] with a small amount of 2.5% data, the fine-tuning details are shown in Appendix A.2.

Algorithm A shows the neural network architecture of XComp, which is largely consistent with VideoChat-Flash. It begins with the UMT-L visual encoder [43], which encodes short video clips consisting of 8 frames into visual tokens. Token merging [5] is subsequently applied to reduce the token count to 128 tokens per clip and 16 tokens per frame. A two-layer MLP connector then maps these visual tokens from the visual encoder space into the representation space of the large language model. Finally, Qwen2-1.5B [4] serves as the large language model (LLM) in our framework.

During inference with QC-Comp, multiple forward passes through the model are performed. First, XComp conducts a forward pass to obtain scores for the frames. Note that for videos longer than 512 frames, the video is divided into shorter segments, each processed separately to compute frame scores. These individual scores are then aggregated. Based on the aggregated scores, XComp selects the top-ranked frames and passes them to the model to generate responses.

**Details of LP-Comp** Algorithm B shows the details of LP-Comp (Sec. 3.2). It compresses video tokens layer-by-layer in a suffix-preserving manner. Given input tokens $V^{(\ell)} \in \mathbb{R}^{K \times N^{(\ell)} \times d}$ at layer $\ell$, the algorithm computes the target token count $N_{\text{next}}$ for the next layer using a cosine-based schedule. If no reduction is needed, tokens are returned unchanged. Otherwise, for each video clip and each frame within it, only the last $N_{\text{next}}$ tokens are kept from the current $N_{\text{prev}}$, preserving the temporal suffix. The retained tokens across all frames and clips are concatenated to form the output $V^{(\ell+1)}$. This design ensures progressive compression while retaining semantically rich information.

**Details of QC-Comp** Algorithm C shows the details of Segmented Local Attention, which is the main part of QC-Comp (Sec. 3.3) and computes scores from attention maps in LLM layers. For a given video $V$ and text query $Q$, the algorithm slides a local window (64 frames with a stride of 32) over the video sequence. Within each segment, it computes the full attention map from the LLM's $\ell$-th layer. For each frame in the segment, the attention weights over the query tokens are averaged and accumulated into corresponding 8-frame clip buckets. Although Segmented Local Attention inherently produces clip-level scores, QC-Comp assigns the same score to all frames within a clip, thereby converting clip-level scores to frame-level scores. For long videos exceeding 512 frames, the

**Algorithm A:** MODEL_FORWARD: forward pass with LP-Comp and the score for QC-Comp

**Input:** $\mathcal{M}$ (model), $V$ (video), $Q$ (text), $returnScore \in \{0, 1\}$
**Output:** *logits*; *score* (per-clip) if $returnScore = 1$
**/* Step 1: video encoding */**
$\{V_1^{\text{clip}}, \ldots, V_K^{\text{clip}}\} \leftarrow$ `Partition(`$V$, 8`)` `// separate each 8 frames → 1 clip`
**foreach** $V_k^{\text{clip}}$ **do**
  $T_k \leftarrow \mathcal{M}.\text{umt\_visual\_enc}(V_k^{\text{clip}})$
  $T_k \leftarrow$ `TokenMerge(`$T_k$, 128`)` `// to 128 tokens/clip, 16 tokens/frame`
  $T_k \leftarrow \mathcal{M}.\text{mlp}(T_k)$ `// project to LLM dim`
$V^{(0)} \leftarrow \text{concat}(T_1, \ldots, T_K)$
$Q^{(0)} \leftarrow \mathcal{M}.\text{text\_embed}(Q)$
**/* Step 2: reasoning in LLM layers */**
**for** $\ell = 0$ **to** $\mathcal{M}.L - 1$ **do**
  $[V^{(\ell+1)}, Q^{(\ell+1)}] \leftarrow \mathcal{M}.\text{llm\_layer}_\ell(V^{(\ell)}, Q^{(\ell)})$
  $V^{(\ell+1)} \leftarrow$ `LP_Comp(`$\ell$, $V^{(\ell+1)}$`)` `// token-level compression`
  $score[\ell] \leftarrow$ `SegmentedLocalAttention(`$\ell$, $V^{(\ell)}, Q^{(\ell)}$`)` `// score for QC_Comp`
*logits* $\leftarrow$ `ComputeLogits(`$Q^{(\mathcal{M}.L)}$`)`
**if** *returnScore* **then return** (*logits*, *score*)
**else return** *logits*

method processes multiple overlapping 512-frame chunks. The final frame-level scores are obtained by aggregating the results from these overlapping chunks. To ensure diverse frame-level scores, each frame would be included in $n_{\text{repeat}}$ overlapping and shifted chunks that encourage score variation across neighboring frames. With these frame-level scores, $n\_\text{selected\_frames}$ among total frames are selected. Table B shows the hyperparameters.

### A.2 Supervised Fine-tuning Details

**Hyperparameters** Table A shows the hyperparameters used in fine-tuning. We follow the same training configuration as VideoChat-Flash [35], including the learning rate, weight decay, warmup ratio, and learning rate scheduler. The only difference lies in the frame sampling parameters: `frames_upbound`, `frames_lowbound`, and the default frames per second (FPS). This change is due to our use of LP-Comp, which enables more efficient frame representation. As a result, we double the values of `frames_lowbound` and `frames_upbound` compared to VideoChat-Flash.

**Datasets** We used the 2.5% supervised fine-tuning data collected or released by VideoChat-Flash [35]. During data curation, we disregarded a few datasets that were exceptionally large on disk or whose licenses made automatic download impractical. After this filtering, the final training set contains 71,927 instances are drawn from publicly available datasets, listed in Table C.

**Algorithm B:** LP_COMP: suffix-preserving layer-wise video-token compression

**Input:** $\ell$ (layer index), $V^{(\ell)}$ (video tokens, shape $K \times N^{(\ell)} \times d$)
**Output:** $V^{(\ell+1)}$ (compressed video tokens)

$N_{\text{prev}} \leftarrow \left\lceil \frac{N^{(1)}-1}{2} \cos\left(\frac{\ell}{L}\pi\right) + \frac{N^{(1)}+1}{2} \right\rceil$
$N_{\text{next}} \leftarrow \left\lceil \frac{N^{(1)}-1}{2} \cos\left(\frac{\ell+1}{L}\pi\right) + \frac{N^{(1)}+1}{2} \right\rceil$ `// Eq.(1)`
**if** $N_{\text{prev}} = N_{\text{next}}$ **then return** $V^{(\ell)}$ `// nothing to compress`
**foreach** $T \in V^{(\ell)}$ **do** `// iterate over` $K$ `clips`
  **foreach** $f \leftarrow 1$ **to** $F$ **do** `// iterate over` $F$ `frames in the clip`
    idx_keep $\leftarrow \left[ (f-1)N_{\text{prev}} + N_{\text{prev}} - N_{\text{next}}, \ \ldots, \ fN_{\text{prev}} - 1 \right]$
    $T'_f \leftarrow T[\text{idx\_keep}]$ `// suffix-preservation`
  $T' \leftarrow \text{concat}(T'_1, \ldots, T'_F)$
$V^{(\ell+1)} \leftarrow \text{concat}(T'\text{\_clip1}, \ldots, T'\text{\_clipK})$
**return** $V^{(\ell+1)}$

**Algorithm C:** SEGMENTEDLOCALATTENTION: compute clip-level saliency scores

**Input:** $\ell$ (layer index), $V$ (video tokens), $Q$ (text tokens)
**Output:** *score* (list of length $K$, one mean score per clip)

$L_{\text{seg}} \leftarrow 64$ `// frames per segment (8 clips)`
stride $\leftarrow 32$ `// stride in frames (4 clips)`
*bucket* $\leftarrow [\,]\_k = 1^K$
**for** *start = 0* **to** $F - L_{seg}$ ***step*** *stride* **do** `// slide local window over the video`
  $V_{\text{seg}} \leftarrow V[\text{start} : \text{start} + L_{\text{seg}}]$
  $A \leftarrow \mathcal{M}.\text{llm\_layer}_\ell.\text{attn}([V_{\text{seg}}, Q])$ `// full attention map`
  **for** $i = 0$ **to** $L_{seg} - 1$ **do**
    $c \leftarrow \left\lfloor \frac{\text{start}+i}{8} \right\rfloor$ `// clip index`
    $w \leftarrow \text{mean}(A[i, Q\text{\_start} : Q\text{\_end}])$
    $bucket[c].\text{append}(w)$
**for** $k = 1$ **to** $K$ **do**
  $score[k] \leftarrow \text{mean}(bucket[k])$ `// final clip score`
**return** *score*

Table A: **The hyperparameters used in fine-tuning.**

| Hyperparameter | Value / Description |
|---|---|
| `tunable_parts` | Large Langauge Model |
| `learning_rate` | $1 \times 10^{-5}$ |
| `weight_decay` | 0.0 |
| `warmup_ratio` | 0.03 |
| `lr_scheduler_type` | Cosine |
| `dataloader_num_workers` | 1 |
| `frames_upbound` | 1024 |
| `frames_lowbound` | 128 |
| `local_num_frames` | 8 |
| `sample_type` | Dynamic FPS (8 fps by default) |

Table B: **The hyperparameters used in evaluation.**

| Hyperparameter | Value / Description |
|---|---|
| `temperature` | 0.0 |
| `do_sample` | False |
| `num_beams` | 1 |
| `n_repeat` | 2 (each frame in 2 chunks) |
| `n_selected_frames` | 256 (Long), 512 (MME), 1,024 (MLVU), 2,048 (LVB) |

Table C: **Datasets used for supervised fine-tuning (71,927 instances).**

| Image datasets | Video datasets |
|---|---|
| LLaVA-OneVision [32], LLaVA-NeXT [39], M4-Instruct [33] | Kinetics-400 [27], Something-Something [20], TGIF-QA [25], TVQA [30], CLEVRER [74], NExT-QA [67], FAVD [54], MovieChat-1K [57], TextVR [65], ShareGPT-Video [9], ShareGPT-4o [14], Oops [15], OVIS [51], UVO [60], GUI-World [7], Vript [72], HT-Step [1], Ego4D [21], LLaVA-Video-178K [78], VideoChat-Flash [35] |

# B Additional Experiments

## B.1 Ablation Study on Fine-tuning

Table D presents an ablation study to isolate the effect of our proposed design from that of fine-tuning. Specifically, we compare XComp with a baseline that applies the same fine-tuning procedure (on 2.5% of the data as mentioned in Appendix A.2) to the original VideoChat-Flash-2B model, without introducing our LP-Comp and QC-Comp. VideoChat-Flash-2B+FT achieves comparable performance to the original model, suggesting a limited benefit from fine-tuning. This indicates that the performance gains of XComp stem from our method's enhancements, rather than from fine-tuning alone.

Table D: **Ablation study of fine-tuning.**

| Average Duration | Size | LongVideoBench 473s | MLVU 651s | VideoMME (Long) 2386s | LVBench 4101s |
|---|---|---|---|---|---|
| VideoChat-Flash-2B [35] | 2B | 58.3 | 65.7 | 44.9 | 42.9 |
| VideoChat-Flash-2B+FT | 2B | 57.4 | 65.6 | 44.7 | 43.2 |
| XComp | 2B | **59.7** | **66.7** | **45.6** | **46.2** |

### B.2 Efficiency Analysis

Table E shows the efficiency comparison on a LVBench query (863 text tokens), measured over 10 runs on a single NVIDIA H200. Across 1k–4k frames, XComp reduces LLM TFLOPs by ∼53–58% and latency by ∼45–58%.

Table E: **Efficiency Comparison on LVBench Queries at Inference Stage.**

| Frames | Model | TFLOPs | | Latency | |
|---|---|---|---|---|---|
| 1,024 | VideoChat-Flash | 43 | | 0.22 s | |
| 1,024 | Ours | 20 | (**53%**↓) | 0.12 s | (**45%**↓) |
| 2,048 | VideoChat-Flash | 132 | | 0.54 s | |
| 2,048 | Ours | 58 | (**56%**↓) | 0.25 s | (**54%**↓) |
| 4,096 | VideoChat-Flash | 448 | | 1.56 s | |
| 4,096 | Ours | 187 | (**58%**↓) | 0.66 s | (**58%**↓) |

### B.3 More Evaluations

While XComp primarily targets multiple-choice tasks on long videos, it is also capable of handling other fundamental video understanding tasks. We claim that the extreme compression is not harmful to the fundamental capability for short videos. In particular, we include evaluations on (1) reasoning benchmarks such as CLEVRER, which test causal and counterfactual reasoning, and (2) dense-output tasks such as video captioning (VDC benchmark), which assess the ability to generate descriptive text for videos. As shown in Table F, XComp achieves results comparable to its backbone model (VideoChat-Flash-2B), demonstrating that the compression does not significantly harm performance on these tasks.

We additionally evaluate on MME-VideoOCR, a benchmark designed to measure fine-grained text perception from videos. Table G reports results on both the full benchmark and the subset of long videos (>30s). XComp shows a moderate performance drop compared to the original VideoChat-Flash-2B, consistent with our main paper observations that this is largely due to data differences rather than architecture changes. When compared to fine-tuned with the same backbone, the performance gap is minimal, confirming that the degradation primarily comes from training data rather than compression.

Table F: **CLEVRER and VDC results.**

| | CLEVRER | | | | | | VDC | |
|---|---|---|---|---|---|---|---|---|
| | Expl (Opt) | Expl (Q) | Pred (Opt) | Pred (Q) | Cntrf (Opt) | Cntrf (Q) | BLEU@1 | BLEU@4 |
| VideoChat-Flash-2B | 0.9463 | 0.8497 | 0.7789 | 0.5738 | 0.7775 | 0.4007 | 7.1 | 1.6 |
| XComp | 0.9398 | 0.8406 | 0.7697 | 0.5600 | 0.7625 | 0.3652 | 7.0 | 1.6 |

Table G: **MME-VideoOCR results.**

| Model | MME-VideoOCR (Overall) | MME-VideoOCR (>30s) |
|---|---|---|
| VideoChat-Flash-2B | 37.1 | 49.1 |
| VideoChat-Flash-2B + FT | 34.9 | 46.0 |
| XComp | 35.4 | 46.4 |

### B.4 Multi-hop NIAH

Figure A shows the results of the Multi-Hop Needle-in-a-Haystack QA task [35] which is designed to evaluate extreme long-context reasoning abilities. This benchmark embeds a reasoning path of images within long video sequences, where each image contains clues guiding the model to the next. Given a starting point, the model must trace the correct path, identify the target image (needle), and answer a related question. In this experiment, we use QC-Comp with `n_selected_frames` $= 1$

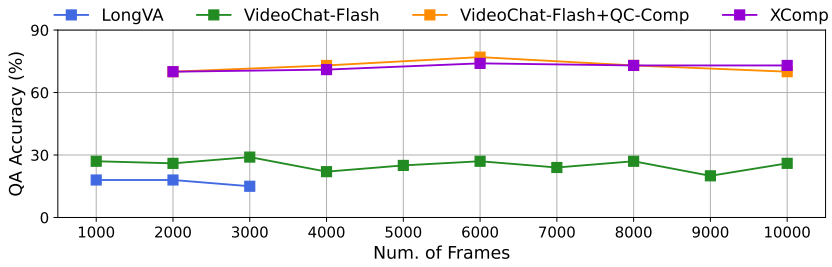


Figure A: **Multi-Hop Needle-in-a-Haystack QA Performance.**

and `n_repeat` $= 8$. It accurately selects the keyframe with 65% accuracy at a sequence length of 6,144, leading to QA average accuracies of 72.6 and 72.2 over 2,000 to 10,000 total frames for VideoChat-Flash+QC-Comp and XComp, respectively.