# OmniSparse: Training-Aware Fine-Grained Sparse Attention for Long-Video MLLMs

**Feng Chen[1], Yefei He[2], Shaoxuan He[2], Yuanyu He[2], Jing Liu[3], Lequan Lin[4], Akide Liu[3], Zhaoyang Li[5], Jiyuan Zhang[5], Zhenbang Sun[5], Bohan Zhuang[2], Qi Wu[1]**

[1]AIML, The University of Adelaide, Australia
[2]Zhejiang University, China
[3]Monash University, Australia
[4]The University of Sydney, Australia
[5]Tiktok, Australia

## Abstract

Existing sparse attention methods primarily target inference-time acceleration by selecting critical tokens under predefined sparsity patterns. However, they often fail to bridge the training–inference gap and lack the capacity for fine-grained token selection across multiple dimensions—such as queries, key-values (KV), and heads—leading to suboptimal performance and acceleration gains. In this paper, we introduce `OmniSparse`, a training-aware fine-grained sparse attention of long-video MLLMs, which is applied in both training and inference with dynamic token budget allocation. Specifically, OmniSparse contains three adaptive and complementary mechanisms: (1) query selection as lazy-active classification, aiming to retain active queries that capture broader semantic similarity, while discarding most of lazy ones that focus on limited local context and exhibit high functional redundancy with their neighbors, (2) KV selection with head-level dynamic budget allocation, where a shared budget is determined based on the flattest head and applied uniformly across all heads to ensure attention recall after selection, and (3) KV cache slimming to alleviate head-level redundancy, which selectively fetches visual KV cache according to the head-level decoding query pattern. Experimental results demonstrate that OmniSparse can achieve comparable performance with full attention, achieving 2.7× speedup during prefill and 2.4× memory reduction for decoding.

## Introduction

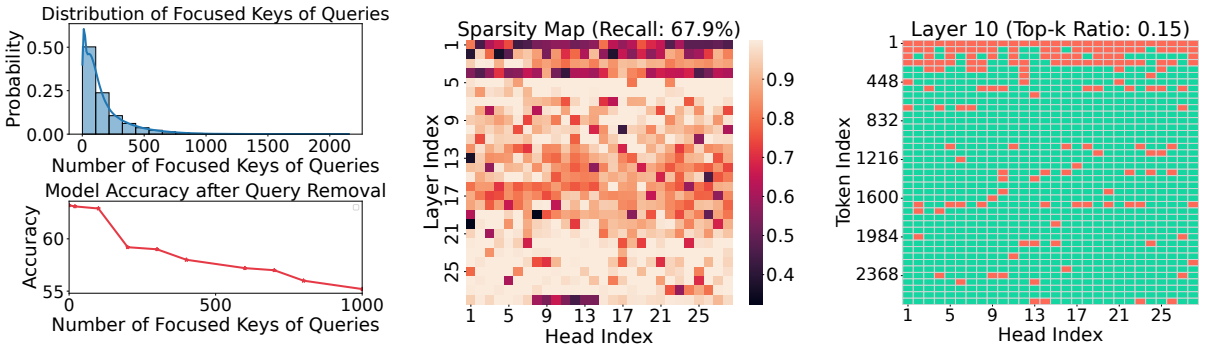
Long-video multimodal large language models (MLLMs) (Li et al. 2025a; Weng et al. 2024; Chen et al. 2024b) are crucial for understanding complex temporal interactions, but their application is limited by the quadratic computational cost of attention mechanism (Vaswani et al. 2017). Recent studies (Chen et al. 2024a; Li et al. 2025b) primarily focus on training-free sparse attention, leveraging the inherent sparsity of attention to predefine sparsity patterns, to significantly reduce computational overhead during inference. For instance, FastV (Chen et al. 2024a) reduces half of the visual tokens after layer 2 based on the attention scores. MMInference (Li et al. 2025b) introduces a modality-aware dynamic sparse attention mechanism that performs predefined pattern search to accelerate the prefill stage. These methods achieve significant hardware efficiency but still struggle with training-inference inconsistency and fine-grained token selection across multiple dimensions, resulting in suboptimal performance and limited acceleration gains.



The training-inference gap stems from the use of sparse attention exclusively at inference time, while models are trained under full attention (Chen et al. 2025a). This mismatch leads to inconsistent attention patterns between training and deployment, ultimately degrading generalization and performance. In addition, fine-grained token selection aims to eliminate redundant computations by identifying and removing token-level redundancies across multiple dimensions—namely queries, key-values, and attention heads. However, existing methods typically focus on only one or two of these dimensions, limiting their ability to fully exploit the potential for computational savings and model efficiency.

In this paper, we propose `OmniSparse`, a training-aware fine-grained sparse attention for long-video MLLMs. OmniSparse dynamically determines token selection budgets across queries, key-values, and attention heads, and applies the same sparse attention during both training and inference to ensure consistency. It adopts a Top-$p$ token-wise sparsification strategy and primarily consists of query selection to remove most functionality-overlapped queries, key-value (KV) selection to determine the minimum token budget across heads, and KV cache slimming to reduce head-wise redundancy. The first two components are designed to alleviate the computation cost of the prefill phase, while the last component focuses on eliminating the memory redundancy of the decoding phase.

**Query selection as binary lazy-active classification.** Although the query-key angle is designed to capture semantic similarity between tokens (Zhang et al. 2024b), not all queries contribute equally. As shown in Figure 1a, we observe that the majority of queries focus on fewer than 100 out of 2600 tokens (*i.e.*, less than 3%), typically concentrating on the attention sink (Xiao et al. 2023) or spatial-temporal adjacent tokens. These "lazy" queries capture limited semantic similarity and can be safely removed from attention with minimal performance degradation, as their functionality is often overlapped with nearby queries or those from other heads (we will demonstrate in Sec. ). To this end, we aim to retain "active" queries that capture broader context, while pruning most of the lazy ones that primarily focus on fewer tokens. We formu-




(a) Query redundancy in MLLMs. (b) Dynamic sparsity in attention. (c) Focus diversity in heads.

Figure 1: (a) The majority of queries focus on fewer than 100 out of 2,600 tokens and can be pruned from attention with minimal performance degradation. (b) The dynamic sparsity across layers and heads suggests that head-wise budget allocation could improve efficiency, but determining an optimal budget for each head is computationally expensive. (c) Heterogeneous token focus across heads (selected keys are highlighted in red) necessitates the head-level KV selection. Data collected from LLaVA-Video-7b (Zhang et al. 2024c) with VideoMME (Fu et al. 2024).

late query selection as a binary lazy-active classification task. Specifically, we construct an active probe key by aggregating all visual keys, and take the key of attention sink as the lazy probe key, which is used to absorb unnecessary weights from other keys (Xiao et al. 2023). For each query, we compute its similarity to these probe keys, where queries exhibiting high similarity with the attention sink are identified as lazy and excluded from subsequent attention computation. To avoid over-pruning of lazy queries, we preserve the queries in the first attention head, which serves as a reliable channel to maintain information integrity.

**KV selection with head-level dynamic budget allocation.** As illustrated in Figure 1b and 1c, we observe heterogeneous attention patterns across heads: each head attends to distinct KV pairs and exhibits distinct sparsity due to varying attention distribution (Jiang et al. 2024). This disparity causes suboptimal key-value selection under the predefined Top-$k$ strategy (Chen et al. 2024a; Gao et al. 2024), leading to under-selection of flatter heads (Lin et al. 2025)—which have relatively uniform attention weights across tokens—and over-selection of sharp heads. A straightforward solution is to dynamically allocate KV budgets per head based on their individual attention distributions. However, this fine-grained allocation incurs substantial computational overhead. To address this, we first identify the flattest attention head using kurtosis, as it typically requires a larger token budget to preserve attention recall. Then we allocate the token budget in this head and then uniformly apply it across all heads, allowing each head to select its most salient KV pairs while ensuring the overall attention recall remains above a predefined threshold.

**KV cache slimming to alleviate head-wise redundancy.** Apart from head-wise selection of visual KV pairs, we further reduce memory overhead by conditionally fetching KV caches based on query classification during decoding. Specifically, when a decoding query is identified as lazy, we skip fetching the corresponding head's visual KV cache entirely. This selective fetching strategy significantly decreases the memory access cost during decoding, as it avoids unnecessary KV cache retrievals.

In summary, our contributions are as follows:

(1) We propose `OmniSparse`, a training-aware fine-grained sparse attention to reduce redundant computation along query, key-value, and head dimensions.

(2) OmniSparse dynamically adapts attention sparsity according to head-specific diversity for efficient prefill, and further reduces memory overhead during decoding by skipping visual KV fetching for heads with lazy decoding queries, enabling fine-grained sparsity across both computation and memory dimensions.

(3) Experimental results demonstrate that OmniSparse attains performance comparable to that of full attention mechanisms, while providing a 2.7× acceleration during the prefill phase and achieving a 2.4× reduction in memory consumption during decoding.

## Related Work

**Training-free sparse attention on MLLMs.** Training-free sparse attention is designed to alleviate the computational constraints during deployment, which arise from the extended visual sequence. FastV (Chen et al. 2024a) selects a set of critical tokens after the first layer, reducing the QKV computations of attention to 1/4. FlexPrefill (Lai et al. 2025) and MMinference (Li et al. 2025b) dynamically search each head using predefined patterns, accelerating computation with corresponding sparse kernels. VisionZip selects critical visual tokens and merges contextual tokens before the LLM to reduce the number of tokens. AIM (Zhong et al. 2024) gradually prunes and merges the redundancy tokens based on embedding similarity. However, these methods still struggle with the training-inference gap, leading to an inevitable performance drop and limiting the acceleration gains.

**Long-video MLLMs.** Long-video MLLMs (Chen et al. 2024b, 2025b; Shen et al. 2024; Wang et al. 2024) have been developed to tackle the challenges associated with processing prolonged video sequences. One approach focuses

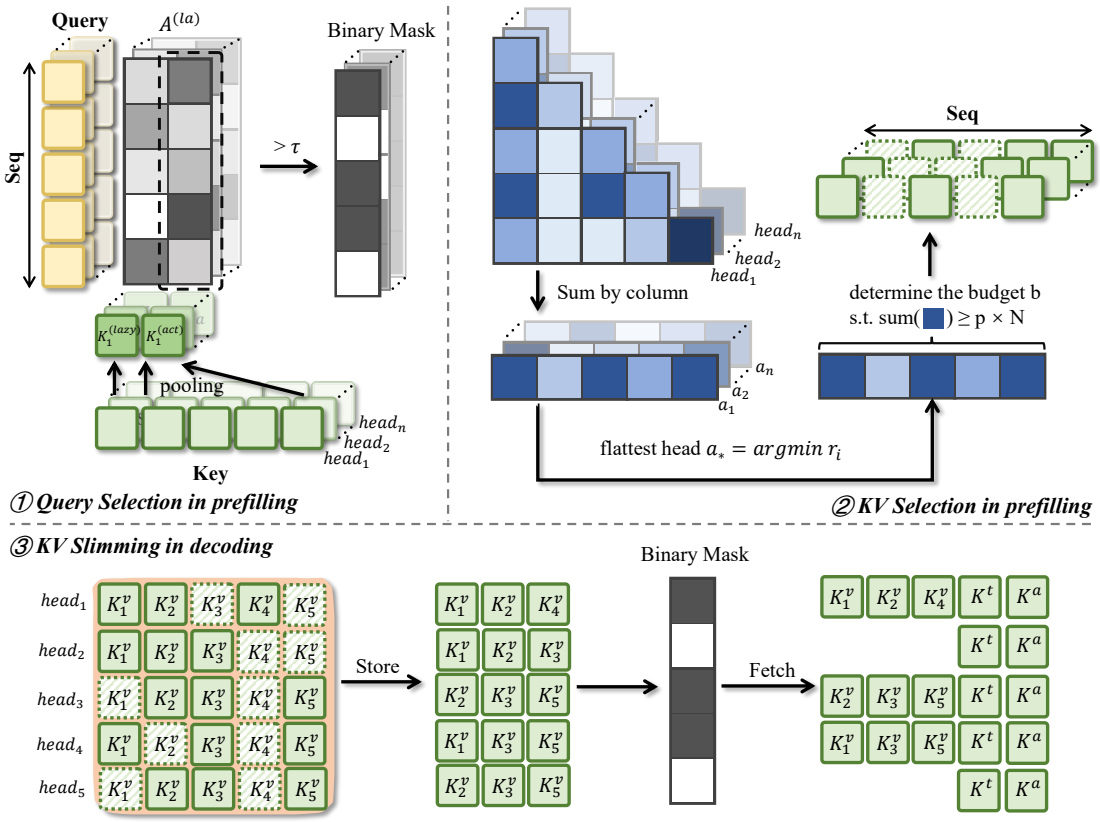



Figure 2: Overview of OmniSparse. During prefill, head-level queries are selected by probing query patterns, with a threshold $\tau$ used to filter out redundant queries. KV selection selects the top $b$ salient KV pairs for each head, with the budget $b$ determined by the flattest head to ensure attention recall exceeds the retention ratio $p$ for all heads. During decoding, only relevant visual KV pairs are fetched for active decoding queries.

on context compression (Weng et al. 2024; Shen et al. 2024). LongVLM (Weng et al. 2024) proposes a hierarchical method to merge local and global information from long-term and short-term video clips, while Maxinfo (Li et al. 2025a) selects key frames and eliminates redundant ones. Another approach extends the context length of LLMs directly (Chen et al. 2024b). For instance, LongVA (Zhang et al. 2024a) leverages the long-context capabilities of LLMs to process long-video sequences, while LongVITA (Shen et al. 2025) and LongVILA (Chen et al. 2024b) aim to train models directly on long-video data. However, these models are usually limited by the prolonged input sequence during inference.

## Method

### Preliminary

**Fine-grained token-level sparse attention.** The multi-head attention mechanism enables the model to capture diverse patterns of interactions across different subspaces via learnable linear projections $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{R}^{d \times d_i}$ at each head $i$, where $d_i = d/h$ with $h$ being the number of heads. Let $\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i$ be the queries, keys, and values at head $i$, respectively. Then, the multi-head attention is mathematically

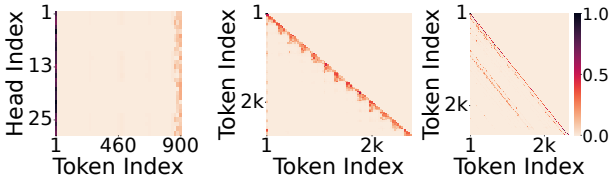



Figure 3: Query redundancy: Queries from different heads (left), spatially adjacent positions (middle), and temporally adjacent positions (right) focus on similar tokens.

defined as follows:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}^O, \tag{1}$$

$$\text{head}_i = \mathbf{O}_i = \text{Att}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) = \sigma\left(\frac{\mathbf{Q}_i \mathbf{K}_i^\top}{\sqrt{d_i}}\right)\mathbf{V}_i, \tag{2}$$

where Concat denotes horizontal concatenation of matrices, $\mathbf{W}^O \in \mathbb{R}^{d \times d}$ is a linear projection to drive final output, $\mathbf{O}_i$ is the output of each head, and $\sigma$ is the `Softmax` ac-

tivation function. In the decoder-only transformer, the self-attention mechanism enables each token in a sequence to attend to preceding tokens when constructing its representation. Fine-grained token-level sparse attention is to select a subset of most important queries, keys and values $\widetilde{\mathbf{Q}}_i = \mathbf{Q}_i \odot \mathbf{M}_i^Q, \widetilde{\mathbf{K}}_i = \mathbf{K}_i \odot \mathbf{M}_i^K, \widetilde{\mathbf{V}}_i = \mathbf{V}_i \odot \mathbf{M}_i^V$ for each head, where $\mathbf{M}_i^Q, \mathbf{M}_i^K, \mathbf{M}_i^V \in \{0,1\}^{N \times d_i}$ are masks to specify the selection, $N$ is the sequence length, and $\odot$ denotes the element-wise matrix multiplication.

## Overview

Given an input sequence $\mathbf{X} = [\mathbf{X}_v, \mathbf{X}_t]$ comprising language tokens $\mathbf{X}_t \in \mathbb{R}^{N_t \times d}$ and vision tokens $\mathbf{X}_v \in \mathbb{R}^{N_v \times d}$, where $N_t \ll N_v$ and $N = N_t + N_v$, our proposed `OmniSparse` module aims to accelerate attention by reducing the overhead introduced by the visual modality. As illustrated in Figure 2, OmniSparse consists of three components: query selection, KV selection, and KV cache slimming. The first two components are designed to accelerate the prefill phase, while the last component aims at speeding up the decoding phase.

## Query Selection as Binary Lazy-Active Classification

While extensive studies (Lu et al. 2025; He et al. 2024) have explored the sparsity of KV pairs, the redundancy among queries remains under-explored, despite their essential role in capturing token-level semantic similarity (Zhang et al. 2024b). As shown in Figure 1a, we observe that most queries attend to fewer than 100 tokens, and skipping these "lazy" queries during attention leads to minimal performance degradation. The underlying reason is that the functional role of queries exhibits substantial overlap. A showcase is illustrated in Figure 3, where many queries focus on similar tokens with queries from other heads or neighboring spatial-temporal positions. Motivated by this observation, we propose to identify and eliminate such lazy queries to reduce redundant computation. We formulate the query pattern as a lazy-active classification problem. Specifically, for each head $i$, we treat the key $\mathbf{K}_i^{(\text{lazy})}$ corresponding to the attention sink (Xiao et al. 2023) (the first token) as the lazy reference, which is a typically lazy pattern and recycles the unnecessary attention weight on broader context (Xiao et al. 2023), and compute the active reference key $\mathbf{K}_i^{(\text{act})}$ by conducting average pooling over all visual keys. We then construct a compact probe key matrix with these two references and compute binary classification logits as:

$$\mathbf{A}_i^{(\text{la})} = \sigma\left(\frac{\mathbf{Q}_i \mathbf{K}_i^{(\text{la})^\top}}{\sqrt{d_i}}\right), \quad \mathbf{K}_i^{(\text{la})} = \left[\mathbf{K}_i^{(\text{lazy})}, \mathbf{K}_i^{(\text{act})}\right], \tag{3}$$

where $\mathbf{A}_i^{(\text{la})} \in \mathbb{R}^{N_v \times 2}$ denotes the attention score over the lazy-active references and serves as classification logits. A query is classified as active if its attention score to the active category exceeds a predefined threshold $\tau$. In addition, to avoid over-pruning of queries focusing on local context, we preserve the queries in the first attention head as active queries, which serves as a reliable channel to maintain information integrity. Formally, for each head $i$, we construct an active query mask for each token $n$ as:

$$\left(\mathbf{M}_i^Q\right)_{n,:} = \begin{cases} \mathbf{1}, & \text{if } \left(\mathbf{A}_i^{(\text{la})}\right)_{n,2} > \tau \text{ or } i = 1 \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{4}$$

Then, queries associated with zeros in $\mathbf{M}_i^Q$ are considered lazy and can be skipped in subsequent self-attention layers to avoid redundant computation.

## KV Selection with Head-level Dynamic Budget Allocation

While using a predefined global budget for KV selection across all attention heads (Lu et al. 2025) enables efficient batch processing and budget management, it struggles to accommodate the varying sparsity levels among different heads (He et al. 2024; Jiang et al. 2024). To mitigate this limitation, we first identify the flattest attention head, characterized by low token sparsity and typically requiring more tokens to preserve attention recall. We then assign a minimal budget based on this head and use it as a uniform baseline budget for all heads, allowing each head to independently select its most important KV pairs. This approach ensures that, after KV selection, all heads maintain an attention recall above the predefined threshold. Moreover, the method is amenable to efficient implementation via GPU batch processing. Specifically, for each head $i$, the accumulated attention score for each key $k$ is computed by summing its corresponding row in the attention matrix $\mathbf{A}$: $a_{i,k} = \sum_m (\mathbf{A}_i)_{k,m}$.

To identify the flattest head, we simply evaluate the attention sparsity for each head by computing the kurtosis (Joanes and Gill 1998) of $a_i$ as $r_i$. The head with the smallest $r$ is regarded as the flattest, and the KV selection budget is determined based on this head. Let $a_*$ be the accumulated attention scores at the flattest head. Then we sort the scores in descending order and use $a_*^{\text{sorted}(j)}$ to denote the $j$-th highest attention score. We determine the budget $b$ of important KV pairs by retaining the minimal number of tokens that collectively preserve the majority of the attention weight:

$$b = \min\left\{ b \in \mathbb{Z} \;\middle|\; \sum_{j=1}^{b} a_*^{\text{sorted}(j)} \geq p \times N \right\}, \tag{5}$$

where $N$ is the number of queries, and $p$ is a threshold that controls the proportion of total attention to retain, which determines a theoretical upper bound of error by $(1-p)\cdot||\mathbf{V}||$ (Lin et al. 2025). Note that the total sum of attention scores in $\mathbf{A}_i$ equals $N$, due to the row-wise `Softmax` activation. Finally, we select the top $b$ most critical keys in each head $i$ with the following key mask for each token $n$ as:

$$\left(\mathbf{M}_i^K\right)_{n,:} = \begin{cases} \mathbf{1}, & \text{if } a_{i,n} \geq a_i^{\text{sorted}(b)} \\ \mathbf{0}, & \text{otherwise} \end{cases}. \tag{6}$$

Since key and value pairs are inherently coupled, we set $\mathbf{M}_i^V = \mathbf{M}_i^K$. Overall, our KV selection strikes a balance between the predefined Top-$k$ strategy across heads (Lu et al. 2025), which is fixed but efficient, and per-head dynamic allocation, which is adaptive but computationally slower.

| Model | Inference Method | Atten FLOPs Reduction | KV Cache Reduction | ActNet-QA | VideoDC | Next-QA | VideoMME |
|---|---|---|---|---|---|---|---|
| baseline-256k | Full | 0% | 0% | 57.4 | 3.72 | 79.0 | 63.6 |
| | FastV (Chen et al. 2024a) | 71.7% | 46.4% | 55.8 | 3.68 | 78.4 | 63.3 |
| | MInference (Jiang et al. 2024) | 35.3% | 0% | 56.3 | 3.70 | 78.3 | 63.5 |
| | ZipVL (He et al. 2024) | 63.5% | 40.2% | 56.9 | 3.69 | 78.3 | 63.5 |
| | OmniSparse($\tau = 0.08, p = 0.82$) | **72.6%** | **53.4%** | 57.4 | 3.71 | 79.1 | 63.5 |
| MOBA-256k | MOBA (Lu et al. 2025) | 84% | 0% | 55.4 | 3.62 | 78.8 | 63.4 |
| OmniSparse-256k | OmniSparse($\tau = 0.08, p = 0.82$) | 85.7% | 66.8% | 57.6 | 3.72 | 78.9 | 63.9 |
| | OmniSparse($\tau = 0.12, p = 0.75$) | 87.1% | 70.2% | 57.4 | 3.70 | 78.9 | 63.8 |
| OmniSparse-1M | OmniSparse($\tau = 0.08, p = 0.82$) | 86.1% | 67.1% | 58.2 | 3.74 | 79.5 | 64.0 |
| | OmniSparse($\tau = 0.12, p = 0.75$) | 87.4% | 68.6% | 58.1 | 3.73 | 79.5 | 64.0 |

Table 1: Demonstration of training-aware sparse attention methods across four benchmarks. Models are evaluated with 256 frames and 65,600 tokens. "OmniSparse-256k" denotes the model trained with OmniSparse and a context length of 256k.

**Block-wise probing and sparse flash attention kernel.** To efficiently probe the attention map, we adopt the block-wise probe strategy (Yuan et al. 2025; Lu et al. 2025; Gao et al. 2024) via pooling the queries and keys in the sequence dimension to approximate the full attention. To avoid the overhead of customizing a probe attention mask for the selected queries from Sec. , we instead apply pooling over all queries. Following SeerAttention, we similarly set the block size to 256 and use a customized block-wise sparse Flash-Attention (Dao 2024) kernel for efficient attention processing.

## KV Cache Slimming to Alleviate Head-wise Redundancy

Fetching visual KV caches for all attention heads during decoding introduces redundancy. Since lazy decoding queries contribute little to the generation process, their associated visual KV caches can be omitted to reduce memory access without affecting performance.

Specifically, at each head $i$, let $\mathbf{q}_i \in \mathbb{R}^{d_i}$ be the decoding query, and $\mathbf{K}_i = [\mathbf{K}_i^v, \mathbf{K}_i^t, \mathbf{K}_i^a], \mathbf{V}_i = [\mathbf{V}_i^v, \mathbf{V}_i^t, \mathbf{V}_i^a]$ be the KV cache, where $v, t, a$ denote the KV from vision tokens, text tokens, and the decoded answer, respectively. First, we store the selected KV pair at the token level. Since each head applies an equal token budget $b$, we can easily index head-wise salient KV pairs in parallel. Then, we probe the decoding query pattern in each head using Eq. 3 to identify the lazy decoding query focusing on fewer visual information, then construct the decoding query mask $\mathbf{M}_i^q \in \mathbb{R}^{d_i}$ following Eq. 4. We selectively fetch visual KV cache $\mathbf{K}_i^v, \mathbf{V}_i^v$ only for the heads whose query is active (*i.e.*, $\mathbf{M}_i^q = \mathbf{1}$), while skipping visual KV fetches in the heads where $\mathbf{M}_i^q = \mathbf{0}$. Finally, the attention at each head for the decoding token is:

$$\mathbf{O}_i = \sigma\left(\frac{\mathbf{q}_i \left[\mathbf{K}_i^v \mathrm{diag}(\mathbf{M}_i^q), \mathbf{K}_i^t, \mathbf{K}_i^a\right]^\top}{\sqrt{d_i}}\right)\left[\mathbf{V}_i^v \mathrm{diag}(\mathbf{M}_i^q), \mathbf{V}_i^t, \mathbf{V}_i^a\right]. \quad (7)$$

Notably, the head corresponding to a lazy decoding query is not pruned entirely, as it still attends to the KV caches from textual tokens and previously decoded answers.

# Experiment

**Models.** For training-aware sparse attention comparison, we implement our OmniSparse on LLaVA-Video (Zhang et al. 2024c) to ensure training-inference consistency, supporting 256k and 1M token length for long video training. We use Long-VITA (Shen et al. 2025) training data to gradually extend the context from 32k to 256k and then 1M. We adopt Qwen2.5-7b-Instruct (Yang et al. 2024a) as LLM backbone, SigLip-400M (Zhai et al. 2023) as visual encoder, and a 2-layer MLP as the adapter, where each frame is encoded to 256 tokens. The whole training is conducted using 256 H100 GPUs. For training-free comparison, we use LongVA-7b (Zhang et al. 2024a), LLaVA-Video-7b-Qwen2 (Zhang et al. 2024c) and LongVILA-7b-Qwen2-1M (Chen et al. 2024b) for their proficiency in handling long-context tasks and compare our method with state-of-the-art sparse attention, including FastV (Chen et al. 2024a), MInference (Jiang et al. 2024), ZipVL (He et al. 2024), and VisionZip (Yang et al. 2024b).

**Benchmarks.** To verify the effectiveness, we use ActivityNet-QA (Yu et al. 2019), EgoSchema (Mangalam, Akshulakov, and Malik 2023), EventBench (Du et al. 2024), VideoMME (Fu et al. 2024), PerceptionTest (Patraucean et al. 2023), NExT-QA (Xiao et al. 2021), LongVideoBench (Wu et al. 2024), MVBench (Li et al. 2024), VNBench (Zhao et al. 2024), and VideoDC (Lab 2024) for video evaluation.

**Baseline.** For both training-aware and training-free comparison, we treat the model trained with full attention as the baseline and empirically set the hyperparameters of OmniSparse as $\tau = 0.08$ and $p = 0.82$ for training and inference. Besides, we apply MOBA (Lu et al. 2025) to the baseline as a training-aware sparse attention mechanism for comparison, using a block size of 2048 and selecting the top 20 most important blocks for attention.

## Performance Comparison

**Training-aware comparison.** We first demonstrate the advantages of training-aware sparse attention, as shown in Table 1. Results across four video benchmarks indicate that our OmniSparse achieves performance comparable to full-attention

| Model | Method | Atten FLOPs Reduction | KV Cache Reduction | ActNet-QA | VideoDC | Next-QA | VideoMME | Average |
|---|---|---|---|---|---|---|---|---|
| LongVA-7b (Zhang et al. 2024a) 110 frames 15840 tokens | Full | 0% | 0% | **50.5** | **3.14** | 67.5 | 52.9 | 50.6 |
| | FastV | 71.7% | 46.4% | 49.7 | 3.06 | 66.9 | 52.0 | 49.8 |
| | MInference | 71.5% | 0% | 49.4 | 3.06 | 67.0 | 52.2 | 49.8 |
| | ZipVL | 79.4% | 55.3% | 50.2 | 3.03 | 67.1 | 52.3 | 50.0 |
| | VisionZip | 78% | 47.0% | 39.6 | 1.06 | 52.4 | 35.2 | 34.5 |
| | OmniSparse(ours) | **82.2%** | **64.9%** | 50.4 | 3.13 | **68.1** | **52.9** | **50.7** |
| LLaVA-Video-7b (Zhang et al. 2024c) 110 frames 20240 tokens | Full | 0% | 0% | 59.6 | 3.66 | 81.2 | 64.7 | 60.5 |
| | FastV | 71.7% | 46.4% | 59.2 | 3.60 | 80.2 | 64.1 | 59.9 |
| | MInference | 22.8% | 0% | 59.6 | 3.64 | 80.6 | 64.6 | 60.3 |
| | ZipVL | 75.9% | 50.7% | 59.4 | **3.66** | 80.6 | 64.5 | 60.2 |
| | VisionZip | 64.8% | 40.6% | 42.1 | 1.35 | 69.5 | 44.9 | 42.5 |
| | OmniSparse(ours) | **75.9%** | **63.7%** | **60.4** | 3.65 | **81.3** | **64.7** | **60.8** |
| LongVILA-7b (Chen et al. 2024b) 256 frames 65800 tokens | Full | 0% | 0% | 59.5 | 2.76 | 80.7 | **60.1** | 56.9 |
| | FastV | 71.7% | 46.4% | 59.1 | 2.72 | 80.1 | 57.8 | 56.1 |
| | MInference | 53% | 0% | **59.7** | 2.77 | 79.1 | 60.0 | 56.6 |
| | ZipVL | 82.1% | 57.7% | 57.8 | 2.75 | 79.5 | 59.4 | 56.0 |
| | VisionZip | | | | OOM | | | |
| | OmniSparse(ours) | **82.3%** | **68.4%** | 59.6 | **2.78** | **80.7** | 60.0 | **57.0** |

Table 2: Inference efficiency comparison of sparse attention methods on four benchmarks.

training while easily scaling to 1 million contexts. Besides, our method significantly outperforms MOBA (Lu et al. 2025), a training-aware sparse attention method with head-wise Top-$k$ KV selection strategy. For example, OmniSparse surpasses MOBA by 2.2% on ActivityNet-QA and provides additional KV cache compression to accelerate the decoding phase. Furthermore, we compare training-aware sparse attention with its training-free counterpart. For our OminiSparse, maintaining consistency between training and inference results in an additional 13.1% reduction in FLOPs and 13.4% reduction in memory usage, while still delivering better performance than the training-free approach.

**Training-free comparison.** We also compare our method with existing sparse attention approaches on mainstream MLLMs for inference acceleration. As shown in Table 2, our method, OmniSparse, not only achieves performance comparable to full attention but also surpasses other sparse attention methods in both FLOPs reduction and KV cache efficiency. For example, compared to the Top-$k$-based FastV (Chen et al. 2024a), OmniSparse further reduces computation by 4.2% in FLOPs and memory usage by 17.3% within the LLaVA-Video (Zhang et al. 2024c) framework. Relative to the Top-$p$-based ZipVL (He et al. 2024), our method enables finer-grained token selection across both query and head dimensions, yielding an additional 16% reduction in KV cache size on LongVA (Zhang et al. 2024a). These results suggest that dynamic budget allocation across multiple dimensions offers greater potential than coarse-grained or fixed-budget approaches. We also present a decoding speed comparison in Table 3, demonstrating that our method outperforms existing approaches in deployment efficiency.

**Runtime deployment speed.** We present the time-to-first-token (TTFT) and decoding throughput in Table 4, comparing the performance of our method against FlashAttention (Dao 2024) on different input lengths using LLaVA-Video-7b. Our

| | FastV | Minference | ZipVL | VisionZip | OmniSparse |
|---|---|---|---|---|---|
| TTFT (s) | 13.6 | 12.9 | 13.3 | 11.6 | 10.1 |
| Throughput (tokens/s) | 8.5 | 4.3 | 8.4 | 9.7 | 11.1 |

Table 3: Decoding speed comparison with existing sparse attention with a context length of 64k. "TTFT" denotes time-to-first-token and is measured with a batch size of 1 on an Nvidia H100 GPU.

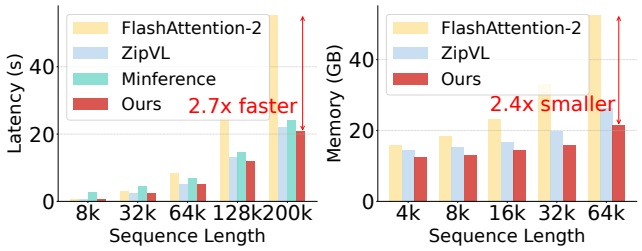



Figure 4: Prefill latency (left) and decoding memory usage (right) under varying sequence lengths on LLaVA-Video-7b.

method consistently demonstrates lower latency and higher throughput across all tested input lengths, highlighting its efficiency over FlashAttention, particularly for smaller input sizes. At larger input lengths of 64k, FlashAttention encounters out-of-memory (OOM) errors, while our method remains scalable. Furthermore, we evaluate OmniSparse under varying sequence lengths in terms of latency and memory, as shown in Figure 4. Our method achieves a 2.7× speedup during the prefill stage and a 2.4× reduction in memory usage during decoding.

| Input Length | Method | TTFT(s) | Throughput (tokens/s) |
|---|---|---|---|
| 16k | FlashAttention | 3.52 | 15.50 |
| | Ours | 3.02 | 40.61 |
| 32k | FlashAttention | 6.40 | OOM |
| | Ours | 5.37 | 16.32 |
| 64k | FlashAttention | 15.45 | OOM |
| | Ours | 10.06 | 11.10 |
| 128k | FlashAttention | 44.82 | OOM |
| | Ours | 20.05 | OOM |

Table 4: Latency and throughput under different input lengths on LLaVA-Video-7b. "TTFT" denotes time-to-first-token and is measured with a batch size of 1 on an Nvidia H100 GPU.

| Sparse Attention | | | | |
|---|---|---|---|---|
| Query Selection | KV Seletion | Ratio | FLOPs Reduction | VideoMME |
| | | 100% | 0% | 64.7 |
| ✓ | | 72.8% | 54.4% | 64.7 |
| | ✓ | 74.2% | 51.5% | 64.7 |
| ✓ | ✓ | 47.1% | 77.9% | 64.7 |
| **KV Cache Compression** | | | | |
| KV Selection | KV Pruning | KV Regrouping | Cache Reduction | VideoMME |
| | | | 0% | 64.7 |
| ✓ | | | 15.5% | 64.7 |
| ✓ | ✓ | | 29.7% | 64.7 |
| ✓ | | ✓ | 51.5% | 64.7 |
| ✓ | ✓ | ✓ | 64.1% | 64.7 |

Table 5: Ablation of sparse attention and KV cache compression on LLaVA-Video-7b. "Ratio" denotes the proportion of tokens participating in attention computation.

## Ablation Study and Discussion

**Ablation on fine-grained selection.** As shown in Table 5, we conduct a comprehensive ablation study on the effects of query selection, KV selection, and KV cache compression strategies on LLaVA-Video-7b. In the top section, we observe that applying query selection or KV selection individually yields a significant reduction in attention FLOPs (54.4% and 51.5%, respectively) without any loss in VideoMME accuracy. When both are applied simultaneously, the computation cost is further reduced (77.9% reduction), demonstrating the complementary nature of the two strategies. In the bottom section, we evaluate the impact of KV cache compression. While individual techniques such as KV pruning or regrouping provide moderate cache savings (29.7% and 51.5%, respectively), combining them leads to substantial memory reduction (64.1%) with no drop in accuracy. These results confirm that our multi-dimensional sparse attention and KV compression techniques can significantly improve efficiency while preserving performance. The results suggest that multi-dimensional sparsification—spanning both query and KV token selection as well as KV cache compression—enables substantial savings in computation and memory without compromising model performance. Moreover, regarding the first-head query, removing it leads to a noticeable 0.4% drop in VideoMME accuracy while offering only a marginal additional FLOPs reduction of 3%. We therefore retain this head to avoid over-pruning queries that capture complementary contextual cues.

**Over-selecting KV pairs for sharp heads.** In KV selection of our method, applying the token budget determined by the flattest head to all other heads inevitably leads to over-selection for sharp heads. As shown in Figure 5, we report the layer-wise sparsity difference between the flattest and sharpest heads. We observe that this over-selection redundancy largely depends on the target attention recall. When the attention recall is set to 0.2, only an additional 5% of queries are selected, whereas at 0.8, the over-selection increases to approximately 20%. This sparsity gap tends to grow as the attention recall increases. Therefore, we set $p = 0.82$, a moderate attentional recall to balance the performance and efficiency. Moreover, compared to assigning separate token budgets for each head, such redundancy remains within an acceptable range for computation latency.

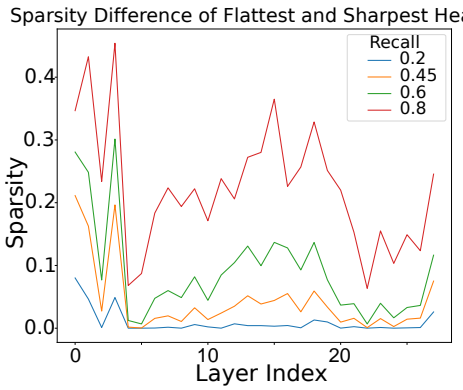



Figure 5: Layer-wise sparsity difference between flattest and sharpest heads on LLaVA-Video-7b.

## Conclusion

In this paper, we have proposed OmniSparse, a training-aware, fine-grained sparse attention framework designed to accelerate inference for long-video MLLMs. OmniSparse keeps training-inference consistency by applying in both training and inference, enabling it to not only approximate the results of full attention but also achieve high acceleration gains. It features multi-dimensional token selection across queries, key-values, and attention heads, and supports optimization for both the prefill and decoding stages. Experimental results have demonstrated that OmniSparse achieves performance comparable to full attention, while delivering a 2.7× speedup during prefill and a 2.4× reduction in memory usage during decoding.

**Limitations and future work.** While our OmniSparse framework demonstrates promising results in terms of reducing computational overhead and memory usage, several limitations remain. The threshold for lazy-active query classification may vary across layers. This variation could affect the balance between computational efficiency and model performance, particularly in layers where the attention patterns are more dynamic or complex. To address this issue, we plan to further investigate the role of attention layers in video perception and understanding.

## References


Chen, F.; He, Y.; Lin, L.; Liu, J.; Zhuang, B.; and Wu, Q. 2025a. Zipr1: Reinforcing token sparsity in mllms. *arXiv preprint arXiv:2504.18579*.

Chen, G.; Li, Z.; Wang, S.; Jiang, J.; Liu, Y.; Lu, L.; Huang, D.-A.; Byeon, W.; Le, M.; Rintamaki, T.; et al. 2025b. Eagle 2.5: Boosting Long-Context Post-Training for Frontier Vision-Language Models. *arXiv preprint arXiv:2504.15271*.

Chen, L.; Zhao, H.; Liu, T.; Bai, S.; Lin, J.; Zhou, C.; and Chang, B. 2024a. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, 19–35. Springer.

Chen, Y.; Xue, F.; Li, D.; Hu, Q.; Zhu, L.; Li, X.; Fang, Y.; Tang, H.; Yang, S.; Liu, Z.; et al. 2024b. Longvila: Scaling long-context visual language models for long videos. *arXiv preprint arXiv:2408.10188*.

Dao, T. 2024. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *International Conference on Learning Representations*.

Du, Y.; Zhou, K.; Huo, Y.; Li, Y.; Zhao, W. X.; Lu, H.; Zhao, Z.; Wang, B.; Chen, W.; and Wen, J.-R. 2024. Towards event-oriented long video understanding. *arXiv preprint arXiv:2406.14129*.

Fu, C.; Dai, Y.; Luo, Y.; Li, L.; Ren, S.; Zhang, R.; Wang, Z.; Zhou, C.; Shen, Y.; Zhang, M.; et al. 2024. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. *arXiv preprint arXiv:2405.21075*.

Gao, Y.; Zeng, Z.; Du, D.; Cao, S.; So, H. K.-H.; Cao, T.; Yang, F.; and Yang, M. 2024. Seerattention: Learning intrinsic sparse attention in your llms. *arXiv preprint arXiv:2410.13276*.

He, Y.; Chen, F.; Liu, J.; Shao, W.; Zhou, H.; Zhang, K.; and Zhuang, B. 2024. Zipvl: Efficient large vision-language models with dynamic token sparsification and kv cache compression. *arXiv preprint arXiv:2410.08584*.

Jiang, H.; Li, Y.; Zhang, C.; Wu, Q.; Luo, X.; Ahn, S.; Han, Z.; Abdi, A. H.; Li, D.; Lin, C.-Y.; et al. 2024. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *arXiv preprint arXiv:2407.02490*.

Joanes, D. N.; and Gill, C. A. 1998. Comparing measures of sample skewness and kurtosis. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 47(1): 183–189.

Lab, L. 2024. Video Detail Caption. https://huggingface.co/datasets/lmms-lab/VideoDetailCaption. Accessed: 2025-05-06.

Lai, X.; Lu, J.; Luo, Y.; Ma, Y.; and Zhou, X. 2025. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference. *arXiv preprint arXiv:2502.20766*.

Li, K.; Wang, Y.; He, Y.; Li, Y.; Wang, Y.; Liu, Y.; Wang, Z.; Xu, J.; Chen, G.; Luo, P.; et al. 2024. Mvbench: A comprehensive multi-modal video understanding benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 22195–22206.

Li, P.; Abdullaeva, I.; Gambashidze, A.; Kuznetsov, A.; and Oseledets, I. 2025a. MaxInfo: A Training-Free Key-Frame Selection Method Using Maximum Volume for Enhanced Video Understanding. *arXiv preprint arXiv:2502.03183*.

Li, Y.; Jiang, H.; Zhang, C.; Wu, Q.; Luo, X.; Ahn, S.; Abdi, A. H.; Li, D.; Gao, J.; Yang, Y.; and Qiu, L. 2025b. MMInference: Accelerating Pre-filling for Long-Context VLMs via Modality-Aware Permutation Sparse Attention. *arXiv preprint arXiv:2504.16083*.

Lin, C.; Tang, J.; Yang, S.; Wang, H.; Tang, T.; Tian, B.; Stoica, I.; Han, S.; and Gao, M. 2025. Twilight: Adaptive Attention Sparsity with Hierarchical Top-$p$ Pruning. *arXiv preprint arXiv:2502.02770*.

Lu, E.; Jiang, Z.; Liu, J.; Du, Y.; Jiang, T.; Hong, C.; Liu, S.; He, W.; Yuan, E.; Wang, Y.; Huang, Z.; Yuan, H.; Xu, S.; Xu, X.; Lai, G.; Chen, Y.; Zheng, H.; Yan, J.; Su, J.; Wu, Y.; Zhang, Y.; Yang, Z.; Zhou, X.; Zhang, M.; and Qiu, J. 2025. MoBA: Mixture of Block Attention for Long-Context LLMs. *arXiv preprint arXiv:2502.13189*.

Mangalam, K.; Akshulakov, R.; and Malik, J. 2023. Egoschema: A diagnostic benchmark for very long-form video language understanding. *Advances in Neural Information Processing Systems*, 36: 46212–46244.

Patraucean, V.; Smaira, L.; Gupta, A.; Recasens, A.; Markeeva, L.; Banarse, D.; Koppula, S.; Malinowski, M.; Yang, Y.; Doersch, C.; et al. 2023. Perception test: A diagnostic benchmark for multimodal video models. *Advances in Neural Information Processing Systems*, 36: 42748–42761.

Shen, X.; Xiong, Y.; Zhao, C.; Wu, L.; Chen, J.; Zhu, C.; Liu, Z.; Xiao, F.; Varadarajan, B.; Bordes, F.; et al. 2024. Longvu: Spatiotemporal adaptive compression for long video-language understanding. *arXiv preprint arXiv:2410.17434*.

Shen, Y.; Fu, C.; Dong, S.; Wang, X.; Chen, P.; Zhang, M.; Cao, H.; Li, K.; Zheng, X.; Zhang, Y.; et al. 2025. Long-VITA: Scaling Large Multi-modal Models to 1 Million Tokens with Leading Short-Context Accuray. *arXiv preprint arXiv:2502.05177*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, X.; Song, D.; Chen, S.; Zhang, C.; and Wang, B. 2024. LongLLaVA: Scaling Multi-modal LLMs to 1000 Images Efficiently via a Hybrid Architecture. *arXiv preprint arXiv:2409.02889*.

Weng, Y.; Han, M.; He, H.; Chang, X.; and Zhuang, B. 2024. Longvlm: Efficient long video understanding via large language models. In *European Conference on Computer Vision*, 453–470.

Wu, H.; Li, D.; Chen, B.; and Li, J. 2024. Longvideobench: A benchmark for long-context interleaved video-language understanding. *Advances in Neural Information Processing Systems*, 37: 28828–28857.

Xiao, G.; Tian, Y.; Chen, B.; Han, S.; and Lewis, M. 2023. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*.

Xiao, J.; Shang, X.; Yao, A.; and Chua, T.-S. 2021. Next-qa: Next phase of question-answering to explaining temporal actions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9777–9786.

Yang, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Li, C.; Liu, D.; Huang, F.; Wei, H.; et al. 2024a. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Yang, S.; Chen, Y.; Tian, Z.; Wang, C.; Li, J.; Yu, B.; and Jia, J. 2024b. VisionZip: Longer is Better but Not Necessary in Vision Language Models. *arXiv preprint arXiv:2412.04467*.

Yu, Z.; Xu, D.; Yu, J.; Yu, T.; Zhao, Z.; Zhuang, Y.; and Tao, D. 2019. Activitynet-qa: A dataset for understanding complex web videos via question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 9127–9134.

Yuan, J.; Gao, H.; Dai, D.; Luo, J.; Zhao, L.; Zhang, Z.; Xie, Z.; Wei, Y.; Wang, L.; Xiao, Z.; et al. 2025. Native Sparse Attention: Hardware-Aligned and Natively Trainable Sparse Attention. *arXiv preprint arXiv:2502.11089*.

Zhai, X.; Mustafa, B.; Kolesnikov, A.; and Beyer, L. 2023. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, 11975–11986.

Zhang, P.; Zhang, K.; Li, B.; Zeng, G.; Yang, J.; Zhang, Y.; Wang, Z.; Tan, H.; Li, C.; and Liu, Z. 2024a. Long Context Transfer from Language to Vision. *arXiv preprint arXiv:2406.16852*.

Zhang, X.; Chang, X.; Li, M.; Roy-Chowdhury, A.; Chen, J.; and Oymak, S. 2024b. Selective Attention: Enhancing Transformer through Principled Context Control. *Advances in Neural Information Processing Systems*, 37: 11061–11086.

Zhang, Y.; Wu, J.; Li, W.; Li, B.; Ma, Z.; Liu, Z.; and Li, C. 2024c. Video Instruction Tuning With Synthetic Data. arXiv:2410.02713.

Zhao, Z.; Lu, H.; Huo, Y.; Du, Y.; Yue, T.; Guo, L.; Wang, B.; Chen, W.; and Liu, J. 2024. Needle In A Video Haystack: A Scalable Synthetic Evaluator for Video MLLMs. *arXiv preprint arXiv:2406.09367*.

Zhong, Y.; Liu, Z.; Li, Y.; and Wang, L. 2024. Aim: Adaptive inference of multi-modal llms via token merging and pruning. *arXiv preprint arXiv:2412.03248*.