TASK-RELATED TOKEN COMPRESSION IN MULTIMODAL LARGE LANGUAGE MODELS FROM AN EXPLAINABILITY PERSPECTIVE

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing Multimodal Large Language Models (MLLMs) process a large number of visual tokens, leading to significant computational costs and inefficiency. Instruction-related visual token compression demonstrates strong task relevance, which aligns well with MLLMs' ultimate goal of instruction following. Previous works generally assume that visual tokens achieve better vision–language alignment in the shallow layers of LLMs, which have led to task-related token compression being primarily applied in intermediate LLM layers. In contrast, our study reveals that with proper selection, task-related token compression is feasible at the input stage of LLM with negligible performance loss. This new paradigm significantly reduces task-irrelevant visual tokens and its model-agnostic design enables application without modifying the LLM architecture. Specifically, we suggest that explainability methods for transformer-based architechtures can evaluate the global importance of each visual token with respect to the given instruction, which can effectively guide the task-related token compression for MLLMs. Furthermore, we propose to learn a mapping from the attention map of the first LLM layer to the explanation results, thereby avoiding the need for a full inference pass. Interestingly, this mapping can be learned using a simple and lightweight convolutional network, whose training is efficient and independent of MLLMs. Extensive experiments on 11 image and video benchmarks across three leading MLLMs (Qwen2-VL, LLaVA-OneVision, and VILA1.5) demonstrate the remarkable effectiveness and strong generalization of our approach. Additionally, our new compression paradigm achieves faster inference with reductions in both prefilling time and KV-cache memory.

1 INTRODUCTION

With large language models (LLMs) providing a strong foundation Brown et al. (2020); OpenAI (2023); Touvron et al. (2023); Bi et al. (2024), research on multimodal large language models (MLLMs) has gained significant momentum Liu et al. (2023); Chen et al. (2023); Zhu et al. (2024); Bai et al. (2023). Considerable progress has been achieved in various image- and video-related tasks Chen et al. (2024d); Anil et al. (2023). A common paradigm among existing MLLMs is to jointly feed visual tokens (generated by a vision encoder) and textual tokens into the LLM for cross-modal alignment and integration Liu et al. (2023); Zhu et al. (2024); Li et al. (2023b). This paradigm introduces substantial memory and computational overhead due to the high volume of visual tokens, which grows rapidly with higher resolutions or frame rates Wang et al. (2024b); Zhang et al. (2024a). Consequently, there is a pressing need for effective token compression techniques.

Previous exploration of visual token compression methods can be roughly divided into two categories. The first aims to obtain more compact and fewer visual representations (especially for videos) in a task/instruction-agnostic manner (independent of LLM) Bolya et al. (2023); Yang et al. (2024); Shen et al. (2025b); Wang et al. (2025); Shen et al. (2025a). We argue that visual representations are an integral part of MLLMs and serve as the foundation for achieving strong performance and generalization. Many state-of-the-art(SOTA) MLLMs already incorporate built-in, task-agnostic compression mechanisms (e.g., spatial and temporal pooling) instead of relying on separate compression techniques applied afterward Wang et al. (2024b;a); Li et al. (2024c). The second category

focuses on selecting visual tokens that are most relevant to the given instruction. FastV Chen et al. (2024b) is a pioneering work that highlights the importance of retaining all shallow-layer visual tokens in LLMs for lossless compression. While this assumption has been adopted by many subsequent studies Zhang et al. (2024b); Zhao et al. (2024); Xing et al. (2024); Tan et al. (2025); Huang et al. (2024); Wen et al. (2025), we believe it remains open to question: *Are all visual tokens in the shallow layers of LLM truly indispensable for task-related compression?*

This paper seeks to answer the question of whether an effective task-related token compression approach prior to the LLM exists but remains undiscovered, or whether it is inherently infeasible. To the best of our knowledge, our work is among the earliest efforts to investigate this issue. We first explore the use of explainability methods to assess visual token importance with respect to the instruction. Explainability methods for transformer-based architecture generally iteratively update a relevance map across layers using gradient-weighted multi-head attentions Chefer et al. (2021b;a), which effectively captures the global relevance scores of visual tokens. Relevance scores indicating the contributions of input tokens to output can be used to rank and prune less important visual tokens for compression. Comprehensive experiments conducted on both image and video data across three representative MLLMs demonstrate the effectiveness of such a compressor. The results indicate that, with proper selection, pruning tokens that are not relevant to the task at the LLM input stage is indeed feasible. Moreover, unlike previous works motivated by observations derived from specific network architectures (*e.g.*, LLaVA) Chen et al. (2024b); Tan et al. (2025), which limits their generality and transferability, our explainability-based approach is broadly applicable. Rather than relying on the behaviors of specific models, it leverages the inherent characteristics of the applied model.

After validating that the explanation results are effective compression indicators, a lightweight model capable of generating an alternative to the relevance map is further needed to enable efficient and practical deployment. Interestingly, this goal can be achieved by training a simple fully convolutional network that predicts relevance based on the first-layer attention map of the LLM. The training process is highly efficient (*e.g.*, training a 5-layer network using only 10K image data) and does not involve any changes to the MLLM itself. Using the predicted relevance, token compression can be performed prior to the prefill phase with negligible extra computational cost. As a result, both computational and memory overhead during inference are significantly reduced, with no modifications required to either the prefill or decode phases. Last but not least, our approach generalizes well across various architectures, benefiting from the broadly applicable nature of explainability methods and the MLLM-agnostic design of the auxiliary training.

To thoroughly assess the capability of our approach, we apply it to three prominent models with different architectures and visual representations: VILA1.5, LLaVA-OneVision, and Qwen2-VL. We include 11 widely used image and video benchmarks that span a wide range of visual complexities and tasks, ensuring a comprehensive evaluation. Notably, our method achieves significant compression by pruning 75% of video tokens while retaining more than 97% of the original performance across all benchmarks for both VILA1.5 and LLaVA-OneVision. It also performs well on image tasks, where up to 50% of image tokens can be removed with only a minimal performance drop: maintaining over 96% of baseline performance for Qwen2-VL and LLaVA-OneVision.

In summary, the contributions of the work are threefold: (i) reveal that explainability methods can well evaluate the importance of visual tokens, enabling effective token compression. (ii) propose a highly efficient token compressor by learning from explanation results. It allows token compression to be performed before the LLM, significantly reducing inference costs at both the prefill and decode phases. (iii) Validate the effectiveness and generalization of our method through extensive experiments on a wide range of image and video benchmarks across different leading MLLMs.

2 Related Work

Multimodal Large Language Models. Benefiting from advancements in large language models (LLMs) OpenAI (2023); Touvron et al. (2023); Bi et al. (2024), multimodal large language models (MLLMs) have gained considerable attention due to the powerful ability in multi-modal understanding and reasoning Liu et al. (2023); Chen et al. (2023); Bai et al. (2023); Chen et al. (2024d); Anil et al. (2023). Recent advances Li et al. (2024a); Wang et al. (2024b); Zhang et al. (2024a) tend to handle images with higher resolution and videos with more frames, which significantly increases the number of visual tokens and thus the computational burden. This reveals the necessity for to-

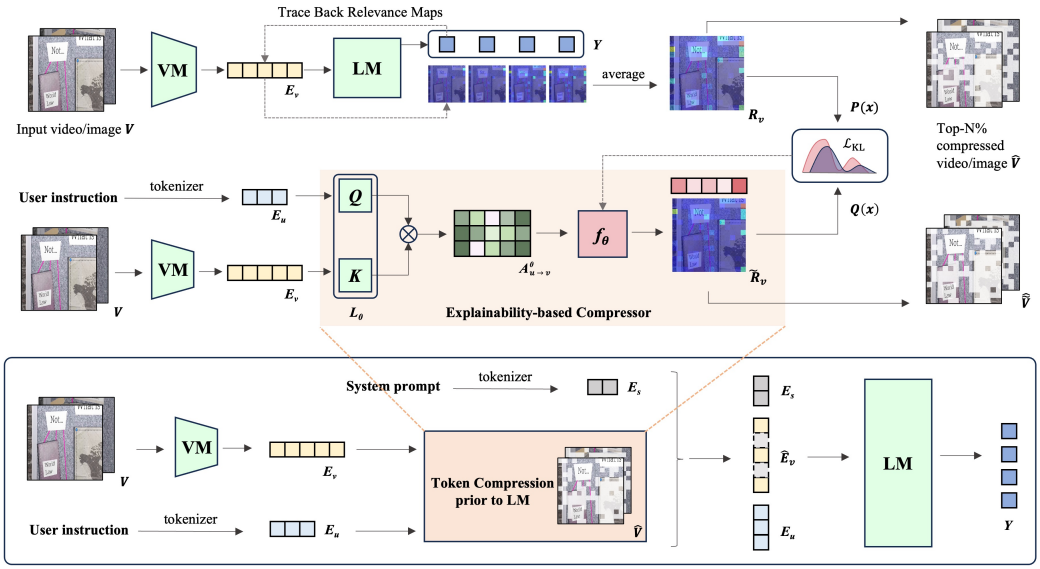


Figure 1: **Overview of our method.** The top portion illustrates the details of our explainability-based compression approach: an explainability method can reveal the important visual tokens (first row, Section 3.2); a lightweight model can then be trained to approximate this explainability and serve as a compression indicator (second row, Section 3.3). The bottom portion shows a general inference framework for MLLMs, where the resulting compressor is applied at the LLM input stage.

ken compression strategies that can balance efficiency and effectiveness. Our work proposes a new token compression paradigm, which removes task-irrelevant visual tokens at the LLM input stage, significantly reducing computational costs without sacrificing performance.

Visual Token Compression. Existing visual token compression methods for MLLMs can be broadly categorized into: task/instruction-agnostic compression Bolya et al. (2023); Yang et al. (2024); Shen et al. (2025b); Alvar et al. (2025) and task/instruction-related compression Chen et al. (2024b); Xing et al. (2024); Huang et al. (2024); Wen et al. (2025). The first category of methods typically introduces additional modules to merge redundant visual tokens based on similarity, addressing the limitations of existing models. However, many recent works have developed techniques to obtain more compact visual representations when building MLLMs Wang et al. (2024b); Chen et al. (2024a). Notably, task/instruction-related compression can further reduce the number of visual tokens on models already incorporate built-in compression mechanisms, offering greater potential for efficiency gains. FastV Chen et al. (2024b) represents a typical method of the second category, which rely on shallow-layer attention maps of the LLM for compression. In this work, we explores the feasibility of an effective task-related token compression prior to the LLM, which functions independently of the architecture and can be applied broadly across different MLLMs.

3 Method

3.1 Background and Motivation

Current Multimodal Large Language Models (MLLMs) typically follow a framework in which a vision encoder is incorporated to encode visual signals into a sequence of tokens Liu et al. (2023); Bai et al. (2023); Chen et al. (2023). Specifically, multiple frames or patches are sampled from a video or an image, and their corresponding visual tokens are encoded. These visual tokens are then flattened and concatenated with textual prompt tokens before being fed into a Large Language Model (LLM) to generate a response. Formally, let V be the video or image, and let $\mathbf{VM}$ and $\mathbf{LM}$ represent the vision encoder and the language model, respectively. The visual token embeddings E_v can be represented as $E_v = \mathbf{VM}(V) \in \mathbb{R}^{N_v \times C}$, where N_v is the number of visual tokens and C is the feature dimension. [1] Let $E_s \in \mathbb{R}^{N_s \times C}$ and $E_u \in \mathbb{R}^{N_u \times C}$ denote the token embeddings of the system

[1] A cross-modal projector is commonly employed in such architectures. For notational simplicity, we denote both the vision encoder and the projector by $\mathbf{VM}$.

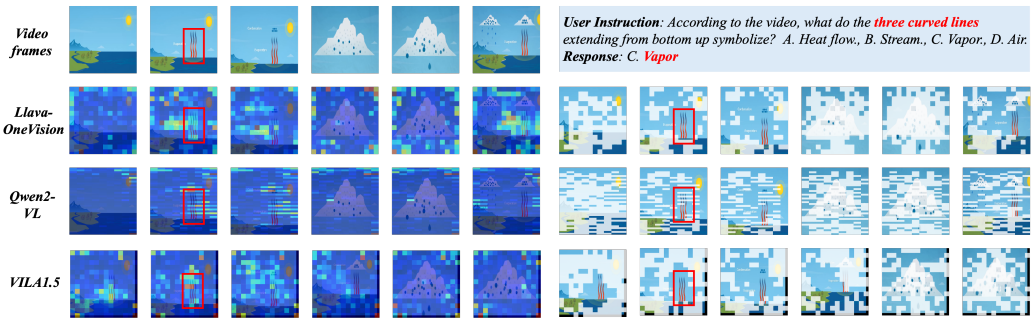


Figure 2: **Visualization of R_v obtained via the explainability method (left) and the corresponding token pruning results (right).** Based on R_v, the top 50% of visual tokens are retained and the rest masked in white. Given the instruction querying the "three curved lines", R_v highlights the regions corresponding to the "three curved lines", guiding the selective retention of the associated visual tokens. All three MLLMs generate the correct answer using only the retained tokens. More visualization cases are presented in Appendix A.

prompt and user instruction, respectively. By feeding E_v together with E_s and E_u into the LLM, a textual response is generated, *i.e.*, $Y = \mathbf{LM}(E_s, E_v, E_u)$. An additional compression module $\mathbf{Comp}$ can be introduced to prune visual tokens, while keeping the MLLM architecture—including both the $\mathbf{VM}$ and the $\mathbf{LM}$—unchanged during this pruning process.

E_v can be considered as general-purpose representations of visual signals that are task/instruction-agnostic. Recent advances have developed techniques to reduce the number of visual tokens to obtain a more compact E_v when building MLLMs Wang et al. (2024b;a); Li et al. (2024c). Therefore, instead of further compressing E_v in isolation (namely task-agnostic compresssion methods like Bolya et al. (2023); Shen et al. (2025a)), our objective is to assess the importance of each token in E_v with respect to a given instruction, and subsequently prune those that are less essential. Moreover, we investigate how to perform token compression prior to LLM computation, *i.e.*, compressing E_v to $\hat{E}_v = \mathbf{Comp}(E_v|E_u) \in \mathbb{R}^{\hat{N}_v \times C}$ and then computing $Y = \mathbf{LM}(E_s, \hat{E}_v, E_u)$, where $\hat{N}_v$ is much smaller than N_v. In contrast to previous methods Chen et al. (2024b); Huang et al. (2024), our method does not require any modifications to the prefill and decode phases during inference, and computational and memory overhead can be significantly reduced in both phases.

The details of our approach are presented below. In Section 3.2, we introduce explainability methods to assess the importance of visual tokens and guide token compression. A learning mechanism is then proposed to predict the explanation results in Section 3.3, which ultimately enables effective token compression at the LLM input stage.

3.2 Token Compression with Explainability

To reduce task-related redundancy at the token level, we need to estimate the contribution of each visual token to the model response. Explainability methods for LLMs facilitate this goal by generating a relevance map through the integration of attention weights and corresponding gradients, effectively revealing where the model genuinely focuses. The resulting relevance map highlights the contributions, enabling us to rank and prune these visual tokens accordingly. The pipeline for this section is shown in the first row of Figure 1.

Relevance Maps by Explainability Method. We adopt a generic explainability method similar to Yao et al. (2024); Chefer et al. (2021b) to compute the relevance of the response-to-vision. The relevance values reveal the distribution of importance across visual tokens utilized by the LLM. Without loss of generality, assume that the LLM in an MLLM has L layers, and denote the generated sequence of textual tokens as $Y = \{y_0, y_1, \ldots, y_{T-1}\}$. Specifically, we trace back the semantic relevance flow from generated tokens to raw visual inputs. For each y_t at the t-th generation step, the relevance map R_t is first initialized as an identity matrix and then iteratively updated across layers. Denote A_t^l and ∇A_t^l as the multi-head attention map and the corresponding gradients in the l-th layer, obtained during the forward and backward passes, respectively. R_t is updated as:

$$R_t = R_t + \mathbb{E}_h(A_t^l \odot \nabla A_t^l) \cdot R_t, \tag{1}$$

where $\odot$ represents Hadamard product, and $\mathbb{E}_h$ is the mean across the heads dimension. The update is performed from the 0-th layer to the last layer. In the end, the relevance of y_t to visual signals can be extracted by indexing the corresponding positions in the last row of R_t, that is, $R_t[-1, N_s : N_s + N_v]$. Finally, we aggregate visual relevance across all time steps t by averaging, obtaining the overall visual relevance scores $R_v \in \mathbb{R}^{1\times N_v}$ with respect to the current response. This well-grounded importance assessment R_v can then be used to rank and select visual tokens.

Visual Token Compression Using Relevance Scores. The importance of visual tokens related to the instruction can be ranked according to R_v. We can prune the less important visual tokens down to a target count of $\hat{N}_v$, resulting in compressed token embeddings $\hat{E}_v$ as LLM input.

Observation. We visualize R_v and the corresponding token pruning results for LLaVA-OneVision, Qwen2-VL, and VILA1.5 in Figure 2. While visualizations from different MLLMs show varying appearances due to differences in how each model processes visual input, they exhibit clear common patterns — R_v for each model consistently highlights the regions corresponding to the "three curved lines" in the video, demonstrating the robustness of our method. Moreover, experimental results show that retaining 50% of the original visual tokens based on R_v preserves over 98% of the performance on image benchmarks and 99% on video benchmarks (see Section 4.2 for details). We draw the following conclusion: *the explanation results faithfully capture the visual information essential for the MLLM to answer the question, and retaining only the corresponding visual tokens does not compromise model performance.*

3.3 Explainability-based Compressor Learning

The relevance map offers valuable insights into achieving token compression at the LLM input level. However, its practical application is limited by the fact that R_v is derived post-hoc – only after the model has already generated the output. To address this limitation, we propose to approximate R_v using a standalone module trained independently of the MLLM. By learning to capture attention patterns and generate relevance estimates $\tilde{R}_v$, this module enables token compression prior to LLM without modifying or retraining the MLLM. Importantly, this module prioritizes efficiency: it is lightweight, requires a small amout of training data, and can be trained quickly, making it practically applicable. The pipeline for this section is shown in the second row of Figure 1.

Model Architecture. As shown in Eq. 1, the relevance map is essentially obtained by aggregating attention maps, suggesting that learning a mapping from attention maps to relevance maps is promising. Interestingly yet reasonably, we find in practice that a simple convolutional network applied to the first-layer attention suffices, which guarantees the compressor's efficiency in terms of model size, training time, and computation (implementation details and efficiency analysis can be found in Appendix B and Appendix C). Formally, let A^0 be the first-layer attention map. Similar to Chen et al. (2024b); Zhao et al. (2024), we focus specifically on the attention scores that visual tokens receive from textual instruction tokens. Accordingly, we extract the submap $A^0_{u\to v} \in \mathbb{R}^{N_u\times N_v}$ by indexing the corresponding positions. We then average the N_u scores for each visual token to obtain a compact representation, resulting in $A^0_v \in \mathbb{R}^{1\times N_v}$.[2] This averaged attention vector A^0_v is subsequently fed into a 1D convolutional model f_θ to predict visual relevance:

$$\tilde{R}_v = f_\theta(A^0_v). \tag{2}$$

Note that a softmax operation is applied at the end of f_θ, making $\tilde{R}_v$ a probability distribution. In addition, a separate instance of f_θ is used for each MLLM, because it is trained to approximate the explainability patterns specific to that particular MLLM.

Training Objectives. We process R_v into the training label R^*_v by masking the bottom 50% values Gu et al. (2021) and normalizing the remainder into a probability distribution. Instead of softmax—which yields near-uniform values due to the closeness of raw scores—we normalize by dividing each score by the total, thereby preserving relative differences. Finally, given R^*_v and $\tilde{R}_v$, the Kullback–Leibler (KL) divergence is used to measure the difference, defining the loss function:

$$\mathcal{L}_{KL} = \mathbf{KL}(R^*_v||\tilde{R}_v). \tag{3}$$

Oberservation. The learned f_θ can be seamlessly integrated into the MLLM inference pipeline to generate $\tilde{R}_v$, which can guide the token compression. As shown in Figure 1, a visualization of

[2]We omit the head dimension for notational simplicity.

Table 1: **The relevance R_v effectively guides token compression under different retention ratios.** Avg. means the average of performance preservation ratios across all image/video benmarks.

Method	Retention Ratio	Image Benchmark			Avg.(%)	Video Benchmark			Avg.(%)
		MME	MMStar	MMVet		Video-MME	MVBench	MMBench-V	
LLaVA-OneVision	100%	1997.7	60.5	48.7	100	53.6	41.2	0.41	100
LLaVA-OneVision	50%	1974.2	59.7	47.2	**98.1**	54.3	41.1	0.40	**99.5**
w/GAE-Based Compressor	25%	1977.3	59.3	47.0	**97.8**	53.8	40.9	0.40	**99.1**
Qwen2-VL	100%	2295.1	60.4	54.0	100	50.4	51.0	1.23	100
Qwen2-VL	50%	2297.1	60.3	53.2	**99.5**	51.0	50.7	1.19	**99.1**
w/GAE-Based Compressor	25%	2299.1	58.7	51.7	**97.7**	50.3	49.7	1.17	**97.5**
VILA1.5	100%	1700.3	38.7	39.3	100	47.3	34.0	1.29	100
VILA1.5	50%	1740.5	37.2	38.0	**98.4**	47.9	34.2	1.26	**99.8**
w/GAE-Based Compressor	25%	1722.1	35.7	35.6	**94.7**	47.1	35.1	1.28	**100.7**

R_v and $\tilde{R}_v$ is given in the first and second rows, along with their corresponding pruning results, respectively. One can see that $\tilde{R}_v$ closely resembles R_v. Important visual regions related to the question are highlighted in both maps. This observation provides evidence that the lightweight model f_θ can indeed be efficiently and effectively trained to approximate R_v, allowing lossless token compression at the LLM input stage. Quantitative experimental results further support this conclusion (see Section 4.3 for details).

4 Experiments

4.1 Experimental Setup

Models. Experiments are conducted on three MLLMs with different architectures for extensive validation, *i.e.*, LLaVA-OneVision-7B Li et al. (2024a), Qwen2-VL-7B Wang et al. (2024b) and VILA1.5-8B Liu et al. (2024). These models exemplify recent advances in handling high-resolution and long visual inputs. LLaVA-OneVision and Qwen2-VL support arbitrary resolution/length, with Qwen2-VL further introducing dynamic resolution and token aggregation for compact visual representations. VILA1.5 applies spatial token compression when processing images or video frames. They thus provide a strong basis for evaluating our task-related compression, which reduces instruction-related redundancy beyond their built-in instruction-agnostic compression.

Evaluation Tasks. We thoroughly evaluate our method on 11 widely used image and video benchmarks. For image tasks, MME Fu et al. (2023) (all-round capability), MMStar Chen et al. (2024c) (data contamination), MMVet Yu et al. (2024) (subjective evaluation), SEED-Bench Li et al. (2023a) (all-round capability), and POPE Li et al. (2023c) (hallucination evaluation) are included, covering various aspects of MLLM performance. For video evaluation, we select Video-MME(wo sub.) Fu et al. (2024), MVBench Li et al. (2024b), MMBench-Video Fang et al. (2024), NExT-QA Xiao et al. (2021), and ActivityNetQA Yu et al. (2019), providing comprehensive coverage of video understanding abilities across different tasks and video durations. For comparison, we take existing SOTA task-related token compression methods such as FastV Chen et al. (2024b), PyramidDrop Xing et al. (2024), Dart Wen et al. (2025) as the primary baselines, which perform compression in the LLM intermediate layers on visual tokens fused with textual information in the shallow layers.

Implementation Details. We conducted all experiments on A100 GPUs (80GB) and used VLMEvalKit Duan et al. (2024) for benchmarking. Implementation details, including R_v generation, the training data and procedure of f_θ, and inference settings are provided in Appendix B. Following prior works Chen et al. (2024b); Ye et al. (2025), we report FLOPs as the primary metric for evaluating inference efficiency. For a fair comparison, we configure baselines for comparable FLOPs (e.g., pruning at the 2nd layer for FastV). Our method achieves a significant reduction in inference cost with only negligible additional computation. We provide a comprehensive analysis of FLOPs of our method, please refer to Appendix C.

4.2 Effectiveness of Compression with Explainability

We conduct experiments to verify whether the explanation results can guide token compression, *i.e.*, compressing E_v to $\hat{E}_v$ according to R_v and then feeding $\hat{E}_v$ into $\mathbf{LM}$ to generate a response. To

Table 2: **Compare explainability-based compressor on image benchmarks.** Values marked with * in Retention Ratio denote the average retention ratio across LLM layers due to multi-stage compression in PDrop.

Method	Retention Ratio	FLOPs	MME	MMStar	MMVet	SEED	POPE	Avg.(%)
LLaVA-OneVision	100%	1.00×	1997.7	60.5	48.7	76.7	87.4	100
LLaVA-OneVision w/ FastV	50%	0.51×	679.2	42.7	28.8	60.1	10.8	50.9
LLaVA-OneVision w/ Pdrop	51%*	0.51×	1974.7	55.4	41.7	74.8	**87.0**	94.6
LLaVA-OneVision w/ Dart	50%	0.51×	1977.5	55.2	42.3	74.3	85.8	94.4
LLaVA-OneVision w/ Ours	50%	**0.48×**	**1980.8**	**57.5**	**46.2**	**75.3**	86.2	**97.2**
LLaVA-OneVision w/ FastV	25%	0.27×	527.7	41.5	20.6	56.1	10.6	44.5
LLaVA-OneVision w/ PDrop	25%*	0.25×	1888.3	50.1	34.7	70.4	79.6	86.3
LLaVA-OneVision w/ Dart	25%	0.27×	1905.0	48.7	36.7	69.5	80.9	86.9
LLaVA-OneVision w/ Ours	25%	**0.24×**	**1965.9**	**52.1**	**41.8**	**72.7**	**81.3**	**91.6**
Qwen2-VL	100%	1.00×	2295.1	60.4	54.0	75.8	87.5	100
Qwen2-VL w/ FastV	50%	0.51×	1489.3	41.4	34.4	56.0	83.0	73.2
Qwen2-VL w/ PDrop	51%*	0.51×	2288.1	55.4	46.3	73.0	86.3	94.4
Qwen2-VL w/ Dart	50%	0.51×	**2290.0**	55.5	49.4	72.4	**86.6**	95.5
Qwen2-VL w/ Ours	50%	**0.49×**	2288.3	**55.9**	**51.9**	**73.2**	86.4	**96.7**
Qwen2-VL w/ FastV	25%	0.27×	1415.3	37.6	31.4	51.4	77.6	67.7
Qwen2-VL w/ PDrop	25%*	0.25×	2216.3	51.1	42.3	67.4	83.2	88.7
Qwen2-VL w/ Dart	25%	0.27×	2184.5	51.3	45.6	68.2	84.3	90.2
Qwen2-VL w/ Ours	25%	**0.24×**	**2280.9**	**51.8**	**47.3**	**67.9**	**84.8**	**91.8**

assess effectiveness and generalization, we apply the method to three state-of-the-art MLLMs and test them on three image and three video benchmarks.

Table 1 reports the results under retention ratios of 50% and 25%. The strong performance across multiple models and benchmarks demonstrates the effectiveness and broad applicability of such an explainability-based token compressor. For Qwen2-VL, reducing visual tokens by 50% still preserves over 99% of the original performance on both image and video tasks. LLaVA-OneVision retains 99.1% of its video performance even with only 25% of tokens. VILA reduces visual tokens to 98 per image or frame at 50% retention, while maintaining 98% of the original image performance and nearly unchanged video performance. These observations indicate that token compression based on relevance R_v effectively preserves the visual tokens essential for MLLMs to answer the question.

4.3 Effectiveness of Explainability-based Compressor Learning

The performance of the $\tilde{R}_v$-guided token compressor is evaluated in this section. $\tilde{R}_v$ is generated by the learned f_θ, and the token pruning is performed accordingly before the LLM computation. Five image and six video benchmarks are included for evaluation.

Performance Comparison. Table 2 presents the results of LLaVA-OneVision and Qwen2-VL under different token compression retention ratios on image benchmarks. We exclude VILA here because it uses a fixed and relatively small number of image tokens, making compression less meaningful. As shown in the table, at a retention rate of 50%, our compressor demonstrates overall superiority over the baselines at comparable FLOPs, achieving average improvements of 2.6% and 1.2% across all benchmarks for LLaVA-OneVision and Qwen2-VL, respectively. When the retention rate is further reduced to 25%, the performance gains increase to 4.7% and 1.6%, highlighting the enhanced robustness of our method under higher compression rates.

In Table 3, we evaluate the compression performance of LLaVA-OneVision, Qwen2-VL, and VILA on video benchmarks. We make several observations. First, our compressor consistently outperforms baselines at comparable FLOPs, regardless of the model and retention ratio. Both LLaVA-OneVision and VILA are able to maintain 100% performance when 50% of the visual tokens are pruned. Second, VILA exhibits the smallest performance drop, while Qwen2-VL shows the largest, likely due to its attention patterns being harder to capture. Importantly, our task-related compression can still further reduce token redundancy on both Qwen2-VL and VILA, demonstrating that it complements the task-agnostic compression already presented in these models. Finally, comparing the results in Tables 1, 2, and 3, the performance degradation from the R_v-guided compressor to

Table 3: **Compare explainability-based compressor on video benchmarks.** As videos generally exhibit greater visual redundancy, we also evaluate a lower retention ratio 10% to further assess compression robustness, with detailed results reported in the Appendix D.

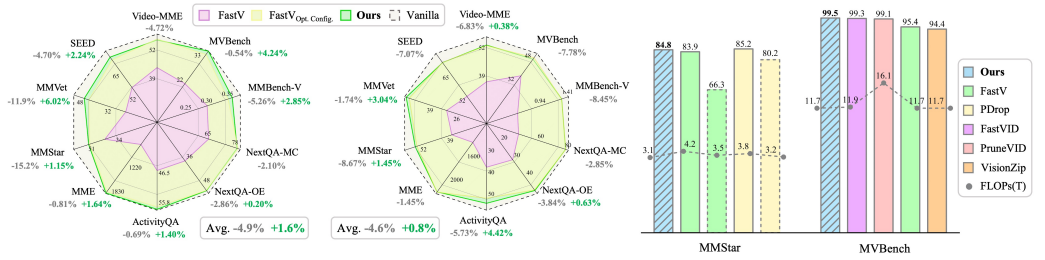
Method	Retention Ratio	FLOPs	Video-MME	MVBench	MMBench-Video	Next-QA		Activity-QA	Avg.(%)
						multi-choice	open-ended		
LLaVA-OV	100%	1.00×	53.6	41.2	0.41	79.2	49.0	56.9	100
LLaVA-OV w/ FastV	50%	0.48×	42.3	25.0	0.35	66.5	36.0	48.5	78.0
LLaVA-OV w/ PDrop	50%*	0.47×	52.7	40.2	0.36	78.4	48.2	56.0	96.6
LLaVA-OV w/ Dart	50%	0.48×	53.2	40.0	0.36	78.0	49.0	56.2	96.9
LLaVA-OV w/ Ours	50%	**0.46×**	**53.4**	**40.5**	**0.43**	**78.6**	**49.7**	**56.5**	**100.4**
LLaVA-OV w/ FastV	25%	0.25×	39.6	23.6	0.30	64.2	33.6	44.5	72.0
LLaVA-OV w/ PDrop	25%*	0.24×	50.8	38.2	0.35	76.3	48.2	53.5	93.6
LLaVA-OV w/ Dart	25%	0.25×	**51.5**	38.7	0.33	76.6	47.0	**55.1**	93.3
LLaVA-OV w/ Ours	25%	**0.22×**	51.3	**39.0**	**0.42**	**77.0**	**49.0**	54.5	**97.3**
Qwen2-VL	100%	1.00×	50.4	51.0	1.23	76.8	45.5	53.6	100
Qwen2-VL w/ FastV	50%	0.48×	32.4	36.3	0.52	43.9	28.3	38.2	61.4
Qwen2-VL w/ PDrop	50%*	0.47×	48.9	49.6	1.14	75.2	45.4	50.8	96.6
Qwen2-VL w/ Dart	50%	0.48×	49.6	49.4	1.17	76.4	44.8	52.0	97.6
Qwen2-VL w/ Ours	50%	**0.46×**	**50.0**	**49.8**	**1.18**	**75.6**	**45.9**	**52.4**	**98.3**
Qwen2-VL w/ FastV	25%	0.25×	31.2	36.1	0.48	42.0	26.8	35.1	58.5
Qwen2-VL w/ PDrop	25%*	0.24×	47.3	46.2	1.11	73.9	44.1	47.8	92.8
Qwen2-VL w/ Dart	25%	0.25×	47.4	47.1	1.10	74.1	**44.5**	49.2	93.7
Qwen2-VL w/ Ours	25%	**0.22×**	**48.1**	**46.7**	**1.11**	**74.2**	44.3	**50.5**	**94.2**
VILA	100%	1.00×	47.3	34.0	1.29	69.9	46.2	55.6	100
VILA w/ FastV	50%	0.49×	42.2	20.7	0.98	62.9	36.8	47.1	80.1
VILA w/ PDrop	50%*	0.49×	47.3	35.0	1.22	69.4	45.8	55.1	99.2
VILA w/ Dart	50%	0.49×	46.1	34.7	1.25	69.2	46.5	55.2	99.2
VILA w/ Ours	50%	**0.47×**	**47.6**	**35.2**	**1.25**	**70.3**	**46.4**	**55.4**	**100.3**
VILA w/ FastV	25%	0.26×	41.4	20.5	0.97	61.5	36.4	46.8	79.0
VILA w/ PDrop	25%*	0.26×	45.2	33.6	1.24	68.2	45.1	54.8	97.4
VILA w/ Dart	25%	0.26×	45.3	34.6	1.23	68.1	45.6	54.2	97.7
VILA w/ Ours	25%	**0.23×**	**45.5**	**35.6**	**1.22**	**69.4**	**46.4**	**54.8**	**99.0**

Table 4: **Efficiency analysis based on Qwen2-VL on MMStar.** We evaluate the inference costs in terms of total inference time, prefilling time, FLOPs, and KV cache memory. KV cache memory is computed with consideration of the Grouped Query Attention (GQA) used in practical inference.

Method	Retention Ratio	FLOPs(×)	Total Inference Time	Prefilling Time	KV Cache	Total Speedup	Prefilling Speedup	MMStar
Qwen2-VL	100%	1.00×	15min24s	6min36s	71.2MB	1.00×	1.00×	61.1
Qwen2-VL w/ FastV	25%	0.27×	12min19s	4min14s	19.7MB	1.25×	1.56×	39.6
Qwen2-VL w/ PDrop	25%*	0.25×	12min15s	4min10s	18.1MB	1.26×	1.58×	53.1
Qwen2-VL w/ Dart	25%	0.30×	12min20s	4min16s	21.6MB	1.25×	1.55×	54.3
Qwen2-VL w/ Ours	25%	**0.24×**	**12min16s**	**4min08s**	**17.8MB**	**1.26×**	**1.60×**	**55.8**

the $\tilde{R}_v$-guided compressor is more pronounced in image tasks. This is likely also due to the greater redundancy in videos, which reduces the learning difficulty.

Applying to Larger Images and Longer Videos. We evaluate the generalization of directly applying the trained f_θ on larger images and longer videos; experimental results are shown in Figure 3, with configurations detailed in Appendix B. Sub-figure(a) show the compression performance on 4 image and 6 video benchmarks based on LLaVA-OneVision and Qwen2-VL. Our method consistently outperforms FastV and achieves higher average performance than its optimal configuration, demonstrating strong generalization to larger images and longer videos than it seen during training. Detailed comparison results on these 10 benchmarks are provided in the Appendix D. Sub-figure(b) show the comparisons on two challenging benchmarks, *i.e.*, MMStar and MVBench. Several strong methods are introduced for comparison: PruneVID Huang et al. (2024), FastVID Shen et al. (2025a), VisionZip Yang et al. (2024), and PyramidDrop Xing et al. (2024). Our approach achieves SOTA performance with the lowest FLOPs. Remarkably, even directly applying the trained f_θ to longer videos with more frames, it still performs favorably compared to methods specifically designed for videos (i.e., FastVID and PruneVID). Beyond superior performance, our lightweight compres-



(a) Radar charts of LLaVA-OneVision(left) and Qwen2-VL(right) with 25% visual tokens retained

(b) Performance preservation(%) of methods based on LLaVA-OneVision

Figure 3: **Comparison results on larger images and longer videos.** Performance preservation ratio measures the performance retained relative to Vanilla. Gray text denotes the gap from Vanilla and green text highlights improvements over the $\text{FastV}_{\text{Opt.Config.}}$.

Table 5: **Ablation study on explainability methods for relevance map generation.** We evaluate two strategies for aggregating multi-head attention maps—gradient-weighted summation and simple averaging—to generate relevance maps for token compression on video and image benchmarks.

Model	Method	Rentation Ratio	Image Benchmark			Video Benchmark			Avg.(%)
			MME	MMStar	MMVet	Video-MME	MVBench	MMBench-V	
LLaVA-OneVision	Vanilla	100%	1997.7	60.5	48.7	53.6	41.2	0.41	100
	Mean-weighted	50%	1974.5	58.5	45.9	53.6	40.8	0.39	97.3
	Grad-weighted		1974.2	59.7	47.2	54.3	41.1	0.40	**98.8**
Qwen2-VL	Vanilla	100%	2295.1	60.4	54.0	50.4	51.0	1.23	100
	Mean-weighted	50%	2300.6	58.2	49.2	49.9	49.9	1.15	96.3
	Grad-weighted		2297.1	60.3	53.2	51.0	50.7	1.19	**99.3**
VILA1.5	Vanilla	100%	1700.3	38.7	39.3	47.3	34.0	1.29	100
	Mean-weighted	50%	1720.8	38.0	34.2	48.0	34.1	1.20	96.9
	Grad-weighted		1740.5	37.2	38.0	47.9	34.2	1.26	**99.1**

sor significantly improves MLLM inference efficiency with negligible additional cost. We follow Dart Wen et al. (2025) and report efficiency in terms of total inference time, prefilling time, FLOPs, and KV cache memory, as shown in Table 4. Our compressor achieves both the best performance and highest efficiency. The prefill-stage acceleration (1.60×) and reduced KV cache footprint (71.2MB →17.8MB) enable efficient processing in prefill and decode stages, keeping total inference time comparable to other methods (see Appendix D for more results).

4.4 Ablation Study

As shown in Eq. 1, the relevance map is updated by aggregating attention maps across layers, where the multiple heads in each layer are combined either via simple averaging or gradient-weighted averaging (used in our approach). Table 5 shows that employing gradient-weighted aggregation to generate R_v for token compression performs consistently better than simple averaging across image and video benchmarks. A reasonable explanation is that differing contributions of attention heads make simple averaging prone to distorting relevance maps Voita et al. (2019).

5 Conclusion

In this work, we demonstrate the feasibility of task-related visual token compression at the LLM input stage. We first demonstrate experimentally that the relevance scores derived from explainability methods well evaluate the task-related importance of visual tokens, which can be used for effective token compression. To enable efficient and practical deployment, we employ a simple convolutional network to learn a mapping from the LLM first-layer attention maps to the explainability-derived relevance scores. Using the predicted relevance scores from lightweight model, token compression can be performed prior to the LLM. Extensive experiments demonstrate the effectiveness and generalizability of our task-related token compression method. Since the relevance scores are obtained via backward computations, their generation is resource-intensive. This poses a challenge in scaling the compressor training to high-resolution images or long video sequences. Future work will explore stronger compressors and the use of relevance scores to guide token compression during training.

Reproducibility Statement. We provide the necessary information to facilitate reproducibility. Experimental settings and implementation details are described in the Section 4.1 and Appendix B. All the datasets used in our experiments are publicly available, and the preprocessing steps of training data are documented in Appendix B.

REFERENCES

Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. Divprune: Diversity-based visual token pruning for large multimodal models. In *CVPR*, 2025.

Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, et al. Gemini: A family of highly capable multimodal models. *arXiv*, 2023.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, et al. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv*, 2023.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, et al. Deepseek LLM: scaling open-source language models with longtermism. *arXiv*, 2024.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, et al. Token merging: Your vit but faster. In *ICLR*, 2023.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, et al. Language models are few-shot learners. *arXiv*, 2020.

Hila Chefer, Shir Gur, and Lior Wolf. Transformer interpretability beyond attention visualization. In *CVPR*, pp. 782–791, 2021a.

Hila Chefer, Shir Gur, and Lior Wolf. Generic attention-model explainability for interpreting bi-modal and encoder-decoder transformers. In *ICCV*, 2021b.

Jieneng Chen, Luoxin Ye, Ju He, Zhaoyang Wang, and Daniel Khashabi. Efficient large multi-modal models via visual context compression. In *NeurIPS*, 2024a.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, et al. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *ECCV*, 2024b.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, et al. Are we on the right way for evaluating large vision-language models? In *NeurIPS*, 2024c.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. *arXiv*, 2023.

Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, et al. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. *arXiv*, 2024d.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, et al. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*, 2022.

Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, et al. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *MM*, 2024.

Xinyu Fang, Kangrui Mao, Haodong Duan, Xiangyu Zhao, et al. Mmbench-video: A long-form multi-shot benchmark for holistic video understanding. In *NeurIPS*, 2024.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, et al. MME: A comprehensive evaluation benchmark for multimodal large language models. *arXiv*, 2023.

Chaoyou Fu, Yuhan Dai, Yondong Luo, Lei Li, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. *arXiv*, 2024.

Jie Gu, Feng Wang, Qinghui Sun, Zhiquan Ye, et al. Exploiting behavioral consistence for universal user representation. In *AAAI*, 2021.

Shuhao Gu, Jialing Zhang, Siyuan Zhou, Kevin Yu, et al. Infinity-mm: Scaling multimodal performance with large-scale and high-quality instruction data. *arXiv*, 2024.

Xiaohu Huang, Hao Zhou, and Kai Han. Prunevid: Visual token pruning for efficient video large language models. *arXiv*, 2024.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, et al. Llava-onevision: Easy visual task transfer. *arXiv*, 2024a.

Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, et al. Seed-bench: Benchmarking multimodal llms with generative comprehension. *arXiv*, 2023a.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, 2023b.

Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, et al. Mvbench: A comprehensive multi-modal video understanding benchmark. In *CVPR*, 2024b.

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

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, and Wayne Xin Zhao. Evaluating object hallucination in large vision-language models. In *EMNLP*, 2023c.

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

Zhijian Liu, Ligeng Zhu, Baifeng Shi, Zhuoyang Zhang, et al. NVILA: efficient frontier visual language models. *arXiv*, 2024.

OpenAI. GPT-4 technical report. *arXiv*, 2023.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, et al. Efficiently scaling transformer inference. In *Conf. Mach. Learn. Syst.*, 2023.

Leqi Shen, Guoqiang Gong, Tao He, Yifeng Zhang, et al. Fastvid: Dynamic density pruning for fast video large language models. *arXiv*, 2025a.

Leqi Shen, Tianxiang Hao, Tao He, Sicheng Zhao, et al. Tempme: Video temporal token merging for efficient text-video retrieval. In *ICLR*, 2025b.

Xudong Tan, Peng Ye, Chongjun Tu, Jianjian Cao, et al. Tokencarve: Information-preserving visual token compression in multimodal large language models. *arXiv*, 2025.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, et al. Llama: Open and efficient foundation language models. *arXiv*, 2023.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, et al. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *ACL*, 2019.

Haicheng Wang, Zhemeng Yu, Gabriele Spadaro, Chen Ju, et al. FOLDER: accelerating multi-modal large language models with enhanced performance. *arXiv*, 2025.

Han Wang, Yuxiang Nie, Yongjie Ye, Guanyu Deng, et al. Dynamic-vlm: Simple dynamic visual token compression for videollm. *arXiv*, 2024a.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv*, 2024b.

Zichen Wen, Yifeng Gao, Shaobo Wang, Junyuan Zhang, and Qintong Zhang. Stop looking for important tokens in multimodal language models: Duplication matters more. *EMNLP*, 2025.

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

Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, et al. Pyramiddrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. *arXiv*, 2024.

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, et al. Visionzip: Longer is better but not necessary in vision language models. *arXiv*, 2024.

Linli Yao, Lei Li, Shuhuai Ren, Lean Wang, et al. Deco: Decoupling token compression from semantic abstraction in multimodal large language models. *arXiv*, 2024.

Weihao Ye, Qiong Wu, Wenhao Lin, and Yiyi Zhou. Fit and prune: Fast and training-free visual token pruning for multi-modal large language models. In *AAAI*, 2025.

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, et al. Mm-vet: Evaluating large multimodal models for integrated capabilities. In *ICML*, 2024.

Zhou Yu, Dejing Xu, Jun Yu, Ting Yu, et al. Activitynet-qa: A dataset for understanding complex web videos via question answering. In *AAAI*, 2019.

Pan Zhang, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, et al. Internlm-xcomposer-2.5: A versatile large vision language model supporting long-contextual input and output. *arXiv*, 2024a.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, et al. Sparsevlm: Visual token sparsification for efficient vision-language model inference. *arXiv*, 2024b.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, et al. Video instruction tuning with synthetic data. *arXiv*, 2024c.

Wangbo Zhao, Yizeng Han, Jiasheng Tang, Zhikai Li, et al. A stitch in time saves nine: Small VLM is a precise guidance for accelerating large vlms. *arXiv*, 2024.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, et al. Minigpt-4: Enhancing vision-language understanding with advanced large language models. In *ICLR*, 2024.

APPENDIX

A MORE VISUALIZATION RESULTS

A.1 VISUALIZATION RESULTS ACROSS DIFFERENT MLLMS

We present visualization results for LLaVA-OneVision, Qwen2-VL, and VILA1.5 on both video and image inputs in Figures 4-8. Given an input image or video V, we first show the visual relevance scores R_v with respect to the current response obtained using an explainability method. Based on R_v, we visualize the results of token pruning at 50% and 25% retention ratios (labeled as Top-50% compressed $\hat{V}$ and Top-25% compressed $\hat{V}$ in the figures). Then, we visualize the pruning results produced by our trained compressor (f_θ) under the same compression ratios (labeled as Top-50% compressed $\hat{v}$ and Top-25% compressed $\hat{v}$ in the figures).

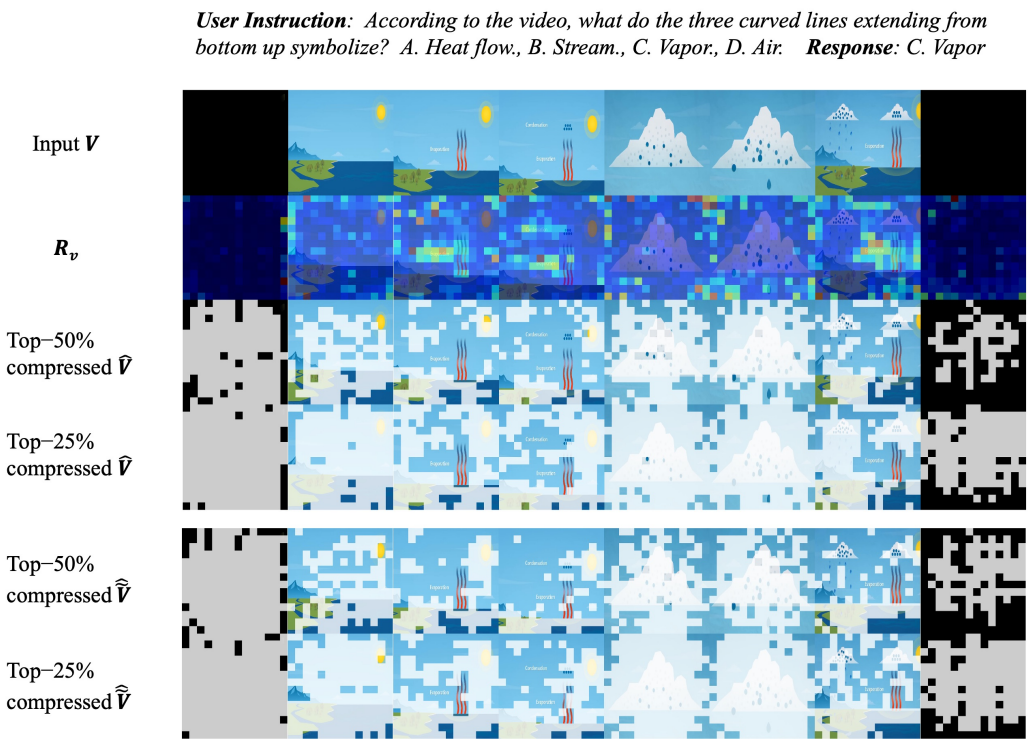


Figure 4: **Video Input Visualizations for LLaVA-OneVision**.

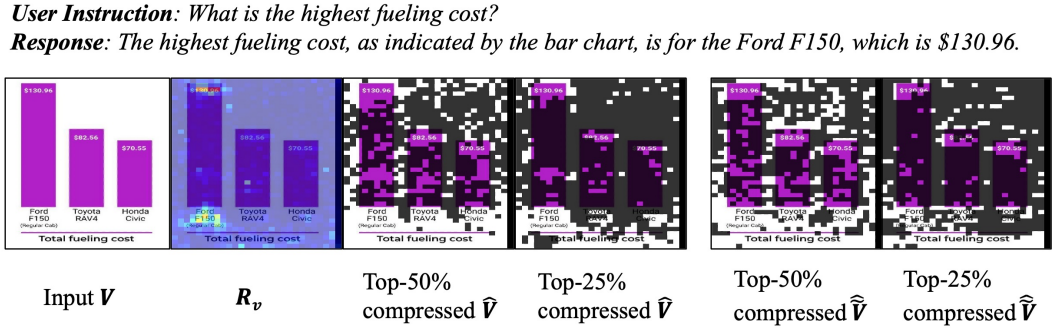


Figure 5: **Image Input Visualizations for LLaVA-OneVision**.

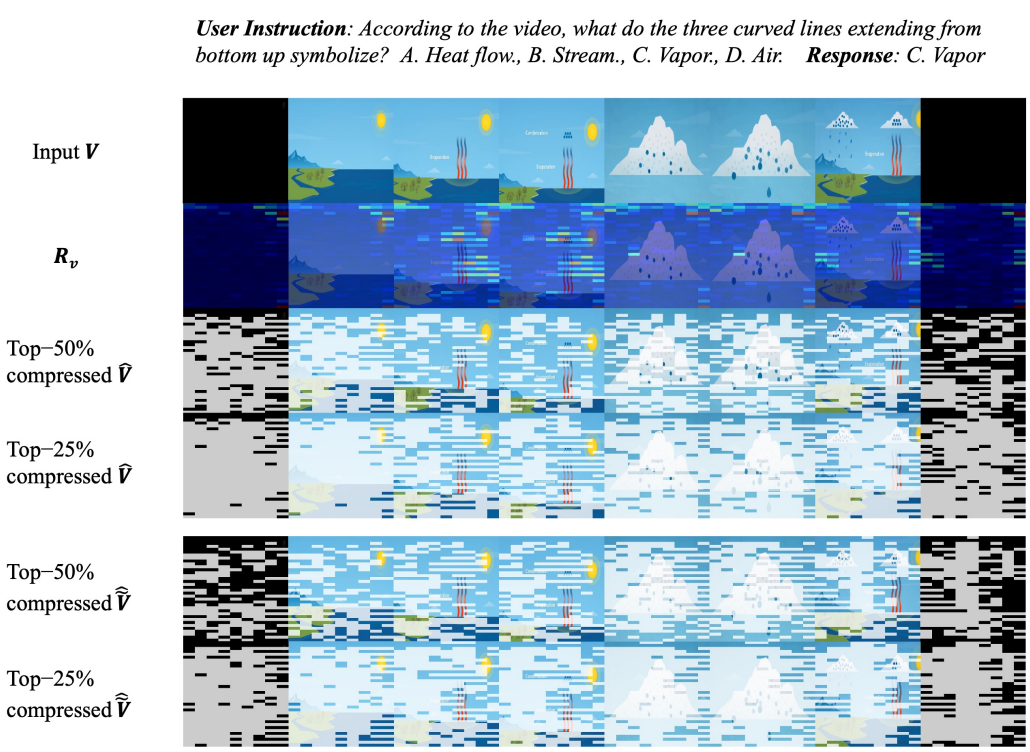


Figure 6: **Video Input Visualizations for Qwen2-VL**.

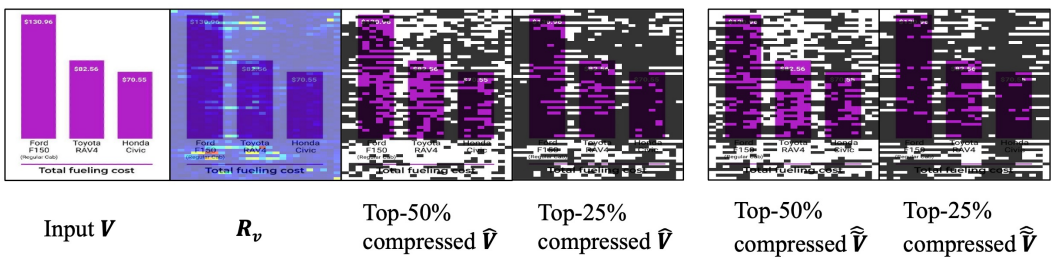


Figure 7: **Image Input Visualizations for Qwen2-VL**.

A.2 Case Study: Explainability Reveals Instruction-Related Visual Tokens

To demonstrate the effectiveness of explainability methods in identifying visual tokens that are highly relevant to instructions, we present 2 case studies covering both video and image inputs.

Given the same input V, the explainability method generates visual relevance scores R_v that selectively emphasize different visual tokens according to varying user instructions. As shown in Figure 9, when the user instruction specifically targets clothing-related information, the visual tokens corresponding to the person's clothing in the video obtain higher relevance scores compared to instructions requesting a general summary. Similarly, in Figure 10, visual tokens relevant to the user instruction exhibit higher relevance scores. When the user instruction specifies excluding the Ford F150, the visual attention shifts primarily to the other two columns. In contrast, when the instruction highlights the highest fueling cost, the Ford F150 column attracts nearly all the attention.

From a visualization standpoint, we further corroborate that the explanation results faithfully reflect the critical visual information required by the MLLM to answer the question.

User Instruction: *According to the video, what do the three curved lines extending from bottom up symbolize? A. Heat flow., B. Stream., C. Vapor., D. Air.* ***Response***: *C. Vapor*

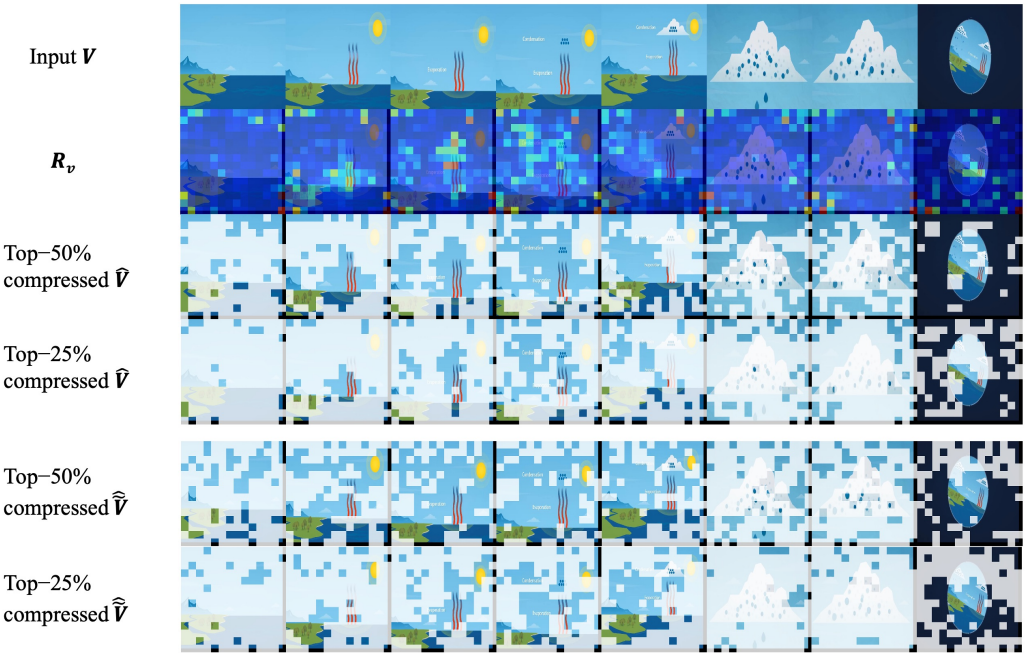


Figure 8: **Video Input Visualizations for VILA**.

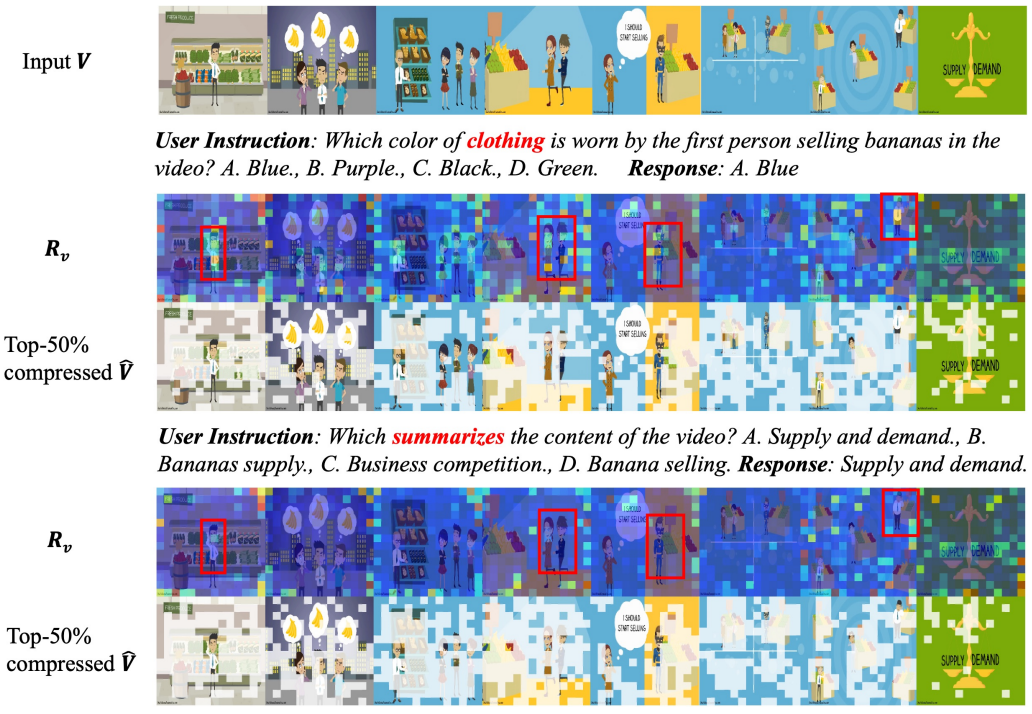


Figure 9: **Case Study 1**.

B Implementation Details.

Generating R_v. To derive R_v, our implementation employs eager attention, allowing access to full-layer attention maps required by the explainability method Chefer et al. (2021b). Compared to

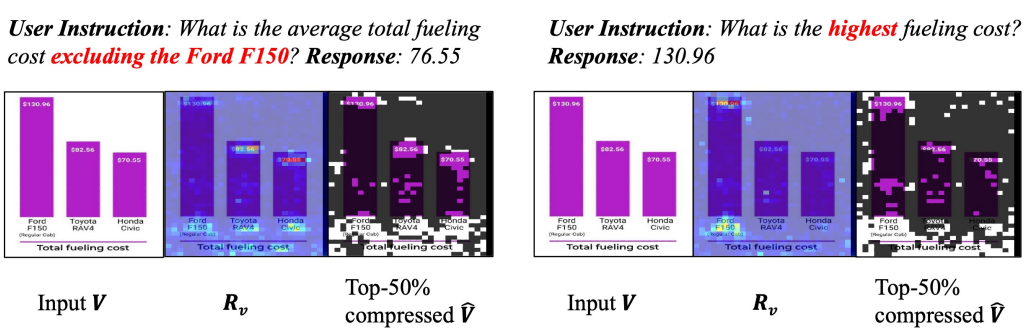


Figure 10: **Case Study 2**.

FlashAttention Dao et al. (2022) and inference based on KV cache Pope et al. (2023), eager attention requires more memory. To avoid out-of-memory errors and ensure efficient data generation, we limit the number of visual tokens to approximately 1500 per sample. Specifically, for video inputs, LLaVA-OneVision, VILA and Qwen2-VL are all set to sample 8 frames, resulting in 1569, 1568 and 1296 visual tokens per video, respectively. For image inputs, LLaVA-OneVision and Qwen2-VL use similar image resolutions, resulting in 1849 and 1500 visual tokens per image, respectively. VILA always processes an image as 196 tokens, eliminating the need for additional configuration. The generated R_v can be used directly to guide token pruning or to train f_θ.

Training f_θ. f_θ is implemented as a five-layer fully convolutional network with channel dimensions of $32, 64, 128, 256,$ and 512. Each layer employs a 1D depthwise separable convolution Chollet (2017), *i.e.*, a depthwise convolution with a kernel size of 3 followed by a pointwise convolution. An additional pointwise convolution layer is applied at the end for channel aggregation. The network is trained by using Adam Kingma & Ba (2015) with default settings and a batch size of 128. Training data is collected from open-source datasets: a subset of LLaVA-Video Zhang et al. (2024c) for videos and a subset of Infinity-MM Gu et al. (2024) for images, each containing approximately 10K samples. Note that f_θ is specific to MLLM, so each MLLM generates its own A_v^0 and R_v based on the input image- or video-text pair for training. The training is performed for roughly 100 epochs, taking about half an hour for image data and less than four hours for video data on a single A100 GPU.

Details of Data for Training f_θ**.** We train our explainability-based compressor based on subsets sampled from high-quality open-source datasets. First, the details of the sampling are as follows:

Image Dataset. For training the compressor used in image tasks, we sample a subset of Infinity-MM that ensures high quality and diversity. The training set primarily consists of data used during Stage 4, including 9k samples randomly sampled from the *Data Generated by GPT-4* subset and 4k from *Synthetic Data*.

Video Dataset. For training the compressor used in video tasks, we sample a subset of LLaVA-Video. Specifically, we include 7k samples from *LLaVA-Video*, 6k from *NeXT-QA* and 4k from *ActivityNetQA*. Note that the training sets of *NeXT-QA* and *ActivityNetQA* have no overlap with the testing sets used in the evaluation. During sampling, since LLaVA-Video contains several parts categorized by task type (open-ended and multi-choice) and video duration (0–30s, 30-60s, 1–2min and 2-3min), we ensure a balanced distribution by randomly selecting an equal number of training examples from each part.

Moreover, we assume that the visual attention distributions (R_v) associated with correct answers exhibit higher quality than those that lead to incorrect answers. Therefore, when training f_θ for a specific MLLM, the sampled data are evaluated by this MLLM, and the samples with incorrect answers are filtered out. Only samples for which the MLLM produces correct answers are retained and used as training data. The number of the retained samples ranges from 8K to 12K.

Inference. The learned f_θ can be seamlessly integrated into existing inference pipelines (no modifications are required for the prefill and decode phases of LLM inference). More interestingly, f_θ is capable of processing longer A_v^0 thanks to the fully convolution design. That is, our compression method can handle larger images and longer videos, even though the visual token number is limited to approximately 1500 during training. Corresponding experiments have been conducted.

In these experiments, Qwen2-VL dynamically processes both images (with 'max_pixels' set to half of its default value) and videos (with 'VIDEO_MAX_PIXELS' and 'FPS_MAX_FRAMES' set to 384×28×28 and 32, respectively). These configurations are set to accommodate hardware resource constraints. LLaVA-OneVision also processes images dynamically with default settings, while sampling 32 frames per video as in Huang et al. (2024) for a fair comparison. For VILA, the input image size cannot be changed, and the number of input video frames is set to 16.

C Efficiency Analysis in Inference

To evaluate computational efficiency during inference, we report the FLOPs of the visual token part. Specifically, we consider the FLOPs of the multihead attention and the feed-forward network (FFN) modules as:

$$\text{FLOPs}_{\text{layer}} = 4nd^2 + 2n^2d + lnm, \tag{4}$$

where n is the number of visual tokens, d is the hidden state size, m is the intermediate size of the FFN, and l is the number of layers in the FFN. To compute the total FLOPs for the entire LLM, we simply multiply Eq. 4 by the number of Transformer layers N_L, i.e., $\text{FLOPs}_{\text{LLM}} = N_L(4nd^2 + 2n^2d + lnm)$.

At the input stage of the LLM, our compressor introduces additional computation. First, we consider the FLOPs introduced by the first-layer attention map:

$$\text{FLOPs}_{\text{attn}} = nd^2 + nd. \tag{5}$$

Note that only the key projection computation for visual tokens and the attention computation from textual tokens to visual tokens are required, corresponding to the term nd^2 and nd, respectively. Only the FLOPs incurred by the visual part are included.

Next, we account for the FLOPs introduced by the 1D depthwise separable convolution:

$$\text{FLOPs}_{\text{conv}} = \sum_{l=1}^{L} n(C_{in}^l k + C_{in}^l C_{out}^i), \tag{6}$$

where C_{in}^l and C_{out}^l denote the number of input and output channels of the l-th layer, respectively. We ensure that the output shape of each convolutional layer remains the same as its input by applying appropriate padding with respect to the kernel size k. As a result, the number of visual tokens n remains constant across all layers. Then the total FLOPs is computed as the sum of the operations across all L convolutional layers.

To intuitively understand the additional computational cost introduced by our method, we adopt a typical parameter configuration used in MLLMs. Specifically, we set the number of visual tokens n to 1568, the hidden dimension d to 3584, the intermediate size m to 18944, and assume 3 layers per FFN block ($l = 3$). For the full LLM, we consider a 28-layer Transformer blocks ($N_L = 28$). For f_θ, we follow the configuration described in Section 4.1 (Experimental Setup). Concretely, the convolutional network consists of 5 layers ($L = 5$) with kernel size $k = 3$, and channel dimensions increasing across layers: 32, 64, 128, 256, and 512. Based on these settings, $\text{FLOPs}_{\text{attn}}$ amounts to approximately 0.02 trillion, $\text{FLOPs}_{\text{conv}}$ is approximately 0.0003 trillion, while $\text{FLOPs}_{\text{LLM}}$ reaches approximately 11.69 trillion. It can be observed that the computational overhead introduced by our compressor is negligible. The computational costs of these two parts account for only **0.17%** and **0.0026%** of the total computational cost, respectively.

The FLOPs reported in the Table 2, Table 3, Table 4 and Figure 3 are computed using a standardized input setting. For image input, FLOPs are computed using a 384 × 512 input image as the reference (the number of visual tokens n is 1728 for LLaVA-OneVision and 1302 for Qwen2-VL). For video input, LLaVA-OneVision and VILA1.5 sample 32 and 16 frames, respectively, resulting in visual token counts n of 6272 and 3136. We fix Qwen2-VL's input to 32 frames at 720 × 1280 resolution (n=5824) for FLOPs calculation.

D Additional Experimental Results

For video tasks, we also investigate a more aggressive compression setting with a 10% retention ratio in Table 6, as a supplement to Table 3. Our method attains the lowest FLOPs while preserving

Table 6: **Compare explainability-based compressor on video benchmarks.**

Method	Retention Ratio	FLOPs	Video-MME	MVBench	MMBench-Video	Next-QA		Activity-QA	Avg.(%)
						multi-choice	open-ended		
LLaVA-OV	100%	1.00×	53.6	41.2	0.41	79.2	49.0	56.9	100
LLaVA-OV w/ FastV	10%	0.12×	37.7	22.4	0.27	60.3	30.6	39.9	66.5
LLaVA-OV w/ PDrop	10%*	0.10×	47.0	37.0	0.35	72.3	43.7	50.0	88.5
LLaVA-OV w/ Dart	10%	0.12×	**47.3**	36.9	0.36	72.7	43.5	50.3	89.1
LLaVA-OV w/ Ours	10%	0.09×	47.1	**37.4**	**0.40**	**76.5**	**45.6**	**51.6**	**92.8**
Qwen2-VL	100%	1.00×	50.4	51.0	1.23	76.8	45.5	53.6	100
Qwen2-VL w/ FastV	10%	0.12×	29.1	37.5	0.44	39.4	23.3	32.0	54.9
Qwen2-VL w/ PDrop	10%*	0.10×	45.2	40.3	0.82	71.5	41.8	45.1	84.1
Qwen2-VL w/ Dart	10%	0.12×	45.8	41.6	0.85	**72.1**	42.6	45.5	85.7
Qwen2-VL w/ Ours	10%	0.09×	**46.1**	**42.5**	**1.00**	72.0	**43.3**	**47.5**	**88.9**
VILA	100%	1.00×	47.3	34.0	1.29	69.9	46.2	55.6	100
VILA w/ FastV	10%	0.12×	37.8	19.5	0.88	57.9	33.9	43.2	73.2
VILA w/ PDrop	10%*	0.11×	43.0	34.4	1.13	65.2	43.5	50.6	93.0
VILA w/ Dart	10%	0.12×	42.9	34.8	1.14	64.8	43.7	51.1	93.4
VILA w/ Ours	10%	0.09×	**43.6**	**35.0**	**1.14**	**67.0**	**44.7**	**53.0**	**95.3**

Table 7: **Compare generalization performance of our compressor on image benmarks.**

Method	Rentention Ratio	FLOPs	MME	MMStar	MMVet	SEED	Avg.(%)
LLaVA-OneVision	100%	1.00×	2002.0	62.0	52.0	76.7	100
LLaVA-OneVision w/ $\text{FastV}_{\text{Opt.Config.}}$	50%	0.51×	1990.3	57.3	48.4	75.7	95.9
LLaVA-OneVision w/ Ours	50%	0.48×	1988.0	57.8	50.2	75.4	**96.8**
LLaVA-OneVision w/ $\text{FastV}_{\text{Opt.Config.}}$	25%	0.27×	1953.7	52.0	43.2	71.5	89.4
LLaVA-OneVision w/ Ours	25%	0.24×	1985.8	52.6	45.8	73.1	**91.9**
Qwen2-VL	100%	1.00×	2316.6	61.1	51.7	76.4	100
Qwen2-VL w/ $\text{FastV}_{\text{Opt.Config.}}$	50%	0.51×	2295.8	57.7	52.4	74.8	98.2
Qwen2-VL w/ Ours	50%	0.49×	2311.7	57.9	53.9	73.9	**98.9**
Qwen2-VL w/ $\text{FastV}_{\text{Opt.Config.}}$	25%	0.27×	2288.2	55.0	49.3	71.1	94.3
Qwen2-VL w/ Ours	25%	0.24×	2283.1	55.8	50.8	71.0	**95.3**

competitive accuracy, achieving average improvements of 3.7%, 3.2%, and 1.9% across all benchmarks for LLaVA-OneVision, Qwen2-VL, and VILA, respectively, compared to the best-performing baseline. Notably, even under such extreme compression, our method consistently delivers strong results, highlighting its robustness across different MLLMs.

We provide full tables of results corresponding to the generalization experiments shown in the Figure 3 (a) in the main text (Applying to Larger Images and Longer Videos), with detailed results for the image and video benchmarks listed in Table 7 and Table 8, respectively. In addition, we provide detailed comparison results shown in the Figure 3 (b) for two challenging benchmarks, MMStar and MVBench, in Table 9 and Table 10.

Table 11 exhibits the additional efficiency analysis on MMVet. Our lightweight compressor achieves substantial reductions in KV-cache usage and accelerates the prefill stage, while achieving the highest task scores and keeping overall inference time comparable to baselines. These results demonstrate that our approach maintains both strong task performance and computational efficiency.

E THE USE OF LARGE LANGUAGE MODELS(LLMS)

In preparing this manuscript, we used a large language model (LLM, specifically GPT-5-mini) solely as a general-purpose writing and editing assistant. The LLM was employed to improve clarity, grammar, and overall presentation of the text. All technical content, experiments results, and interpretations were generated and verified by the authors. The LLM did not contribute to research ideation, experimental design, data analysis, or the writing of original technical content. The authors take full responsibility for all content presented in this paper.

Table 8: **Compare generalization performance of our compressor on video benchmarks.**

Method	Retention Ratio	FLOPs	Video-MME	MVBench	MMBench-Video	Next-QA		Activity-QA	Avg.(%)
						MC	OE		
LLaVA-OV	100%	1.00×	59.3	37.1	0.38	80.9	52.5	58.4	100
LLaVA-OV w/ $\text{FastV}_{\text{Opt.Config.}}$	50%	0.48×	58.8	36.1	0.38	80.5	51.4	58.2	98.9
LLaVA-OV w/ Ours	50%	0.46×	58.8	37.2	0.38	80.2	52.0	58.1	**99.5**
LLaVA-OV w/ $\text{FastV}_{\text{Opt.Config.}}$	25%	0.25×	57.0	35.4	0.35	79.7	50.9	57.2	96.2
LLaVA-OV w/ Ours	25%	0.22×	56.5	36.9	0.36	79.2	51.0	58.0	**97.3**
Qwen2-VL	100%	1.00×	57.1	52.7	1.42	80.7	49.5	57.6	100
Qwen2-VL w/ $\text{FastV}_{\text{Opt.Config.}}$	50%	0.48×	55.4	51.3	1.40	79.6	49.0	55.7	97.9
Qwen2-VL w/ Ours	50%	0.46×	55.7	51.4	1.41	79.5	48.7	56.3	**98.2**
Qwen2-VL w/ $\text{FastV}_{\text{Opt.Config.}}$	25%	0.25×	53.0	49.6	1.30	78.6	47.3	52.0	93.6
Qwen2-VL w/ Ours	25%	0.22×	53.2	48.6	1.30	78.4	47.6	54.3	**94.1**
VILA	100%	1.00×	48.7	31.7	1.30	70.4	45.8	55.2	100
VILA w/ $\text{FastV}_{\text{Opt.Config.}}$	50%	0.49×	48.1	31.5	1.31	70.1	46.5	55.1	100.0
VILA w/ Ours	50%	0.47×	48.4	34.3	1.34	70.0	47.0	56.0	**102.4**
VILA w/ $\text{FastV}_{\text{Opt.Config.}}$	25%	0.26×	46.3	31.8	1.26	69.6	45.6	54.6	98.3
VILA w/ Ours	25%	0.23×	47.4	35.0	1.29	70.0	46.7	55.7	**101.5**

Table 9: **Efficiency and performance comparison across different methods on MMStar.** Values marked with * indicate that the retention ratio refers to the average proportion of retained tokens across all LLM layers, due to multi-stage compression in PDrop. For FastV, the same retention ratio corresponds to different FLOPs when compression is applied at different layers (2nd and 4th).

Method	Retention Ratio	FLOPs(T)	Performance Preservation(%)
LLaVA-OneVision	100%	12.9	100
LLaVA-OneVision w/FastV	25.0%	4.2	83.9
LLaVA-OneVision w/FastV	25.0%	3.5	66.3
LLaVA-OneVision w/PDrop	30.0%*	3.8	**85.2**
LLaVA-OneVision w/PDrop	25.4%*	<u>3.2</u>	80.2
LLaVA-OneVision w/Ours	25.0%	**3.1**	<u>84.8</u>
Qwen2-VL	100%	9.6	100
Qwen2-VL w/FastV	25.0%	<u>3.1</u>	<u>90.0</u>
Qwen2-VL w/Ours	25.0%	**2.4**	**91.3**

Table 10: **Efficiency and performance comparison across different methods on MVBench.** Values marked with * indicate that the retention ratio is reported from the original paper.

Method	Retention Ratio	FLOPs(T)	Performance Preservation(%)
LLaVA-OneVision	100%	52.7	100
LLaVA-OneVision w/FastVID	25.0%	11.7	<u>99.3</u>
LLaVA-OneVision w/PruneVID	17.0%*	11.9	99.1
LLaVA-OneVision w/FastV	25.0%	16.1	95.4
LLaVA-OneVision w/VisionZip	25.0%	<u>11.7</u>	94.4
LLaVA-OneVision w/Ours	25.0%	**11.7**	**99.5**
Qwen2-VL	100%	48.4	100
Qwen2-VL w/FastV	25.0%	<u>14.9</u>	**94.1**
Qwen2-VL w/Ours	25.0%	**10.9**	<u>92.2</u>
VILA1.5	100%	27.0	100
VILA1.5 w/FastV	25.0%	<u>8.2</u>	<u>100.3</u>
VILA1.5 w/Ours	25.0%	**6.3**	**110.4**

Table 11: **Efficiency analysis based on Qwen2-VL on MMVet.** We evaluate the inference costs in terms of total inference time, prefilling time, FLOPs, and KV cache memory. KV cache memory is computed with consideration of the Grouped Query Attention (GQA) used in practical inference.

Method	Retention Ratio	FLOPs(×)	Total Inference Time	Prefilling Time	KV Cache	Total Speedup	Prefilling Speedup	MMVet
Qwen2-VL	100%	1.00×	7min58s	1min30s	71.2MB	1.00×	1.00×	52.0
Qwen2-VL w/ FastV	25%	0.27×	6min50s	0min56s	19.7MB	1.17×	1.61×	33.1
Qwen2-VL w/ PDrop	25%	0.25×	6min49s	0min55s	18.1MB	1.17×	1.64×	47.0
Qwen2-VL w/ Dart	25%	0.30×	6min51s	0min57s	21.6MB	1.16×	1.58×	44.5
Qwen2-VL w/ Ours	25%	**0.24×**	**6min50s**	**0min54s**	**17.8MB**	**1.17×**	**1.67×**	**50.8**