Research Article

# iLLaVA: An Image is Worth Fewer Than 1/3 Input Tokensin Large Multimodal Models

**Lianyu Hu[1], Fanhua Shang[1], Liang Wan[1], Wei Feng[1]**

1. College of Intelligence and Computing, Tianjin University, China

**In this paper, we introduce iLLaVA, a simple method that can be seamlessly deployed upon current Large Vision-Language Models (LVLMs) to greatly increase the throughput with nearly lossless model performance, without a further requirement to train. iLLaVA achieves this by finding and gradually merging the redundant tokens with an accurate and fast algorithm, which can merge hundreds of tokens within only one step. While some previous methods have explored directly pruning or merging tokens in the inference stage to accelerate models, our method excels in both performance and throughput by two key designs. First, while most previous methods only try to save the computations of Large Language Models (LLMs), our method accelerates the forward pass of both image encoders and LLMs in LVLMs, which both occupy a significant part of time during inference. Second, our method recycles the beneficial information from the pruned tokens into existing tokens, which avoids directly dropping context tokens like previous methods to cause performance loss. iLLaVA can nearly 2× the throughput, and reduce the memory costs by half with only a 0.2% - 0.5% performance drop across models of different scales including 7B, 13B and 34B. On tasks across different domains including single-image, multi-images and videos, iLLaVA demonstrates strong generalizability with consistently promising efficiency. We finally offer abundant visualizations to show the merging processes of iLLaVA in each step, which show insights into the distribution of computing resources in LVLMs. Code is available at https://github.com/hulianyuyy/iLLaVA.**

## 1. Introduction

Over the past several years, Large Vision-Language Models (LVLMs)[1][2] have demonstrated tremendous progress and enabled promising applications in various downstream tasks including cross-model retrieval[3][4][5][6], image captioning[7][8] and image/video generation[9][10]. Within these impressive models, images are usually first divided into patches and then transformed into token sequences, finally fed into a Large Language Model (LLM) and concatenated with system prompts and user instructions to generate the desired textual outputs. The direct combination of image tokens and textual contexts already shows impressive performance over a broad range of tasks, which enables the emergence of powerful AI tools like GPT-4o, Gemini 1.5 pro, DALL·E3 and Kling.

However, the surge of these models also faces severe challenges in computational complexity and computing resources in real life. As they usually rely on the self/cross-attention operations to model input information, the

inherent $\mathcal{O}(n^2)$ computational complexity of attention operations corresponding to input tokens introduces heavy computations. Moreover, when faced with image/video inputs, the transformed image input sequences always own thousands or ten thousands of tokens, which further drastically increase the computational burden. In practice, inference with a LVLM with 34B size would need around 80GB memory[11], which is unavailable for many institutions with limited computing resources in real-world scenarios. The slow inference speed for LVLMs also poses substantial concerns in cases requiring fast responses[12][13] for real-time processing. Thus, to promote the inference efficiency of LVLMs is of great significance in real-world scenarios.

Some works on model acceleration found that the input features are inherently redundant and proposed to directly prune unnecessary tokens for inference acceleration. In Vision Transformers (ViTs), ToMe[14] adopts a soft bipartite matching algorithm to merge tokens from one subset into another, and finally reduce the token number by a large portion. SparseViT[15] prunes the windowed activations in ViTs and applies the evolutionary search to efficiently find the optimal layerwise sparsity configuration. In LVLMs, FastV[12] tries to directly discard 50% unnecessary tokens in the language model to accelerate the inference process. LLaVA-PruMerge[16] designs an adaptive token selection strategy based on Interquartile Range to detect outliers and perform merging. These methods could notably increase the throughput by a large margin, showcasing the efficacy of reducing feature redundancy to accelerate models.

In this paper, we revisit the utilization of feature redundancy in LVLMs and design a novel simple method for model acceleration, termed iLLaVA, which excels previous methods in both accuracy and efficiency. iLLaVA accomplishes this goal by two key factors. First, while most previous methods only reduce tokens within the language model, our method accelerates the forward pass of both image encoders and Large Language Models (LLMs), which further lifts the upper bound of efficiency. Second, instead of directly dropping unnecessary features, our method recycles and merges the pruned tokens into existing tokens, avoiding discarding beneficial information for performance loss. As shown in Fig. 1, iLLaVA could nearly $2\times$ the throughput, reduce the memory costs by half with only a 0.2% - 0.5% performance drop. When evaluated upon various benchmarks including single-image benchmarks, multi-image benchmarks and video benchmarks, iLLaVA demonstrates great generalizability and robustness across models of different scales including 7B, 13B and 34B. We show that iLLaVA could decrease the computations and memory of a large model (e.g., 7B) to what a smaller-size model (e.g., 2B) requires while maintaining superior performance. Visualizations validate that iLLaVA successfully focuses on the meaningful targets in the images.

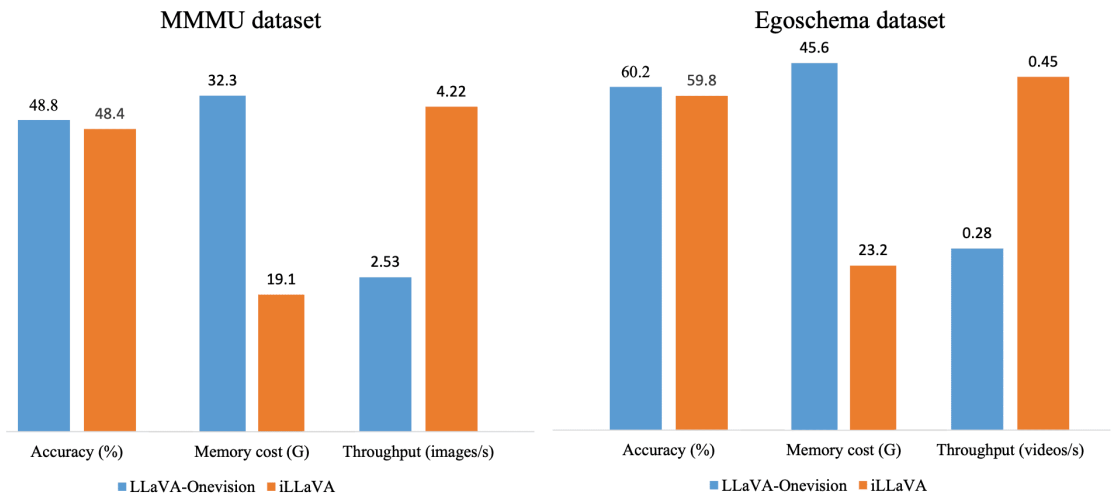


**Figure 1.** Comparison between iLLaVA and the base model LLaVA-Onevision 7B in terms of the accuracy, memory cost and throughputs over MMMU[17] and Egoschema[18] datasets.

## 2. Related Work

### *2.1. Large Vision-Language Models*

Our work is closely related to the surge of LVLMs. Traditional methods[19][20][21] usually collected large vision-language datasets and learned joint representation between vision and language from scratch to handle different tasks. These methods usually worked well in in-domain data but performed inferior in out-domain data that require common sense and world knowledge.

Later, powered by the abundance of high-text data, LVLMs[1][2][8][10] shows impressive performance across a wide range of image understanding[17][22][23][24][25] and reasoning tasks[26][27]. These methods usually first transform the input images as patches, and then feed them into a vision-transformer-based image encoder. The extracted features are sent into a projector for dimension projection, whose outputs are further concatenated with the system prompts and user instructions to serve as inputs for the language model to generate textual outputs. As the attention mechanism with computational complexity $\mathcal{O}(n^2)$ is both used in the image encoder and language model in LVLMs, they have to consume high computational resources and own high inference latency when faced with long input sequences. Our method is deployed upon the state-of-the-art LLaVA-Onevision[1] and QWen2-VL[2] series to accelerate the model forward pass.

### *2.2. Token reduction*

Token reduction has been widely explored in both computer vision[28][15][29][30][31] and natural language processing (NLP)[32][33][34]. However, these methods usually require training, while our method can be done in a training-free

manner. In multimodal learning, a series of methods tried to prune the tokens of intermediate layers for model acceleration. FastV[12] proposes to select the TopK-activated tokens within the language model to accelerate the forward pass. FlexAttention[35] designs a special architecture to first encode low-resolution images and then select corresponding areas in high-resolutions with high responses. LOOK-M[36] utilizes the redundancy of image tokens to guide optimizing the KV cache in LVLMs to achieve reduced memory size. ZipVL[37] propose a dynamic strategy to prune the token across different layers by their received attention scores. Free Video-LLM utilizes an additional CLIP model to help determine which area is the most beneficial and crop the corresponding region into LVLMs for processing. HiRED[38] presents a dynamic high-resolution early dropping strategy for allocating computing resources between partitioned sub-images and the main image. VidCompress[39] designs a memory-enhanced token compressor to encode the short-term memory and long-term memory in videos for model acceleration. Zhang et al. [40] propose to identify redundant tokens by assessing the pattern repetitiveness of correlations between patches and then conducting pruning. LLaVA-PruMerge[16] leverages the attention scores between the [CLS] token and other tokens in the image encoder to guide token dropping before processing with the language model. While these methods mostly only focus on accelerating the language model in LVLMs, we try to make both the image encoder and language model lightweight, which could further enhance the model efficiency. Besides, unlike most previous methods that simply prune the redundant tokens, we reuse them and absorb helpful contexts from them into existing tokens to avoid performance drop.

## 3. iLLaVA

Our target is to design a plug-and-play token merging approach that can be seamlessly combined with existing LVLMs for higher efficiency with no performance sacrifice, without a further need to train.

### *3.1. Motivation*

Our motivation is from the structure of inference time for components in LVLMs, and the inherent feature redundancy of input tokens. First, in Fig. 2(a), we plot the inference time distribution for components in LVLMs including the image encoder, projector and language model. We compute the averaged scores upon single-image benchmarks including SeedBench[26], MMBench[22] and MME[41], mutli-image benchmarks including MuirBench[42] and MMMU[17], video benchmarks including VideoMME[43], Egoschema[18] and ActivityNet-QA[44]as output values for comparison. It can be observed that the image encoder and the language model both occupy a significant portion of inference time within LVLMs, and the proportion of the image encoder consistently increases as the input image tokens increase from single-image tasks, multi-image tasks to video tasks. While most previous methods only consider pruning tokens within the language model, exploring how to accelerate both the image encoder and language model would clearly offer higher upper bounds.

Second, we explore the feature redundancy from the input perspective. In Fig. 2(b), we visualize a raw image on the left and four attention maps drawn from different intermediate layers of the image encoder on the right. On the one

hand, one can see that the attention module only pays major attention to some areas in the image (the bird) and overlooks others, which shows feature redundancy does exist in raw images and the model itself could well learn to attend to important regions. On the other hand, we find that this location ability only performs well within shallow layers of the image encoder (Layer 1-10) and gradually diminishes for deep layers. This may be attributed to the fact that the input tokens for deep layers are mostly composed of context features instead of raw texture features. We thus conclude attention scores from the shallow layer are a great clue in revealing important tokens. Let's forward to the language model. Fig. 2(c) illustrates an attention map from Layer 1 of the language model in the left, and four attention maps from different intermediate layers in the right. As previous methods[12] disclosed, in Layer 1 the image tokens already receive much less attention (deeper color in Fig. 2(c)) compared to system prompts and user instructions. This phenomenon becomes more severe in deeper layers like Layer 3 and Layer 10 where the received attention scores for image tokens become further sparser. This shows that the image tokens in LLMs are overcrowded for the language model.

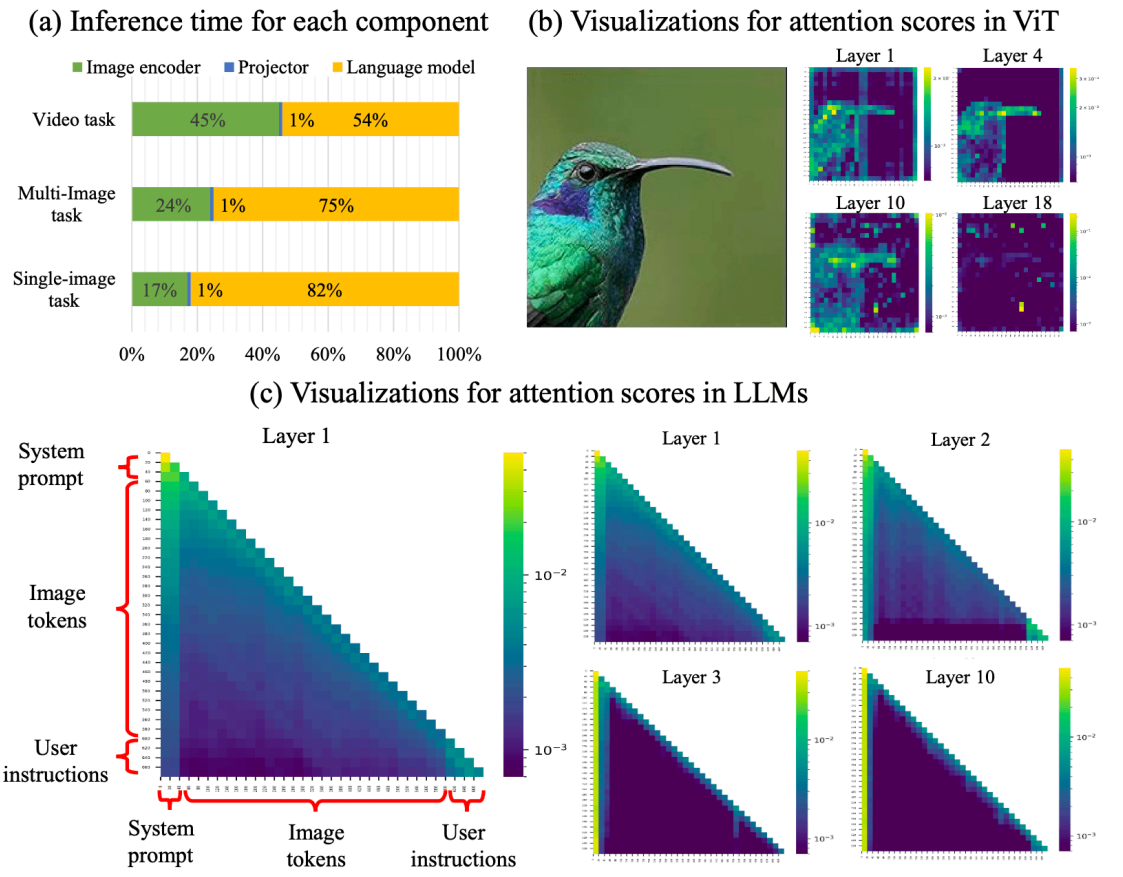


**Figure 2.** (a) The proportion of inference time for each component in LVLMs. (b) Visualizations for the attention scores in the image encoder. (c) Visualizations for the attention scores in LLMs.


doi.org/10.32388/69KFLK

### 3.2. Method

While previous methods have strategically designed various token reduction methods to accelerate models during the inference stage, we find two drawbacks of them. First, most previous methods[12][38][37][35][40] only consider optimizing the inference procedure of LLMs in LVLMs, and our method tries to accelerate the forward pass of both image encoders and large language models, which greatly improves the upper bound of model efficiency. Second, most previous methods simply prune tokens and discard the unused ones, and our method recycles the beneficial information from the pruned tokens into existing tokens, avoiding dropping meaningful contexts. We next detail our design.

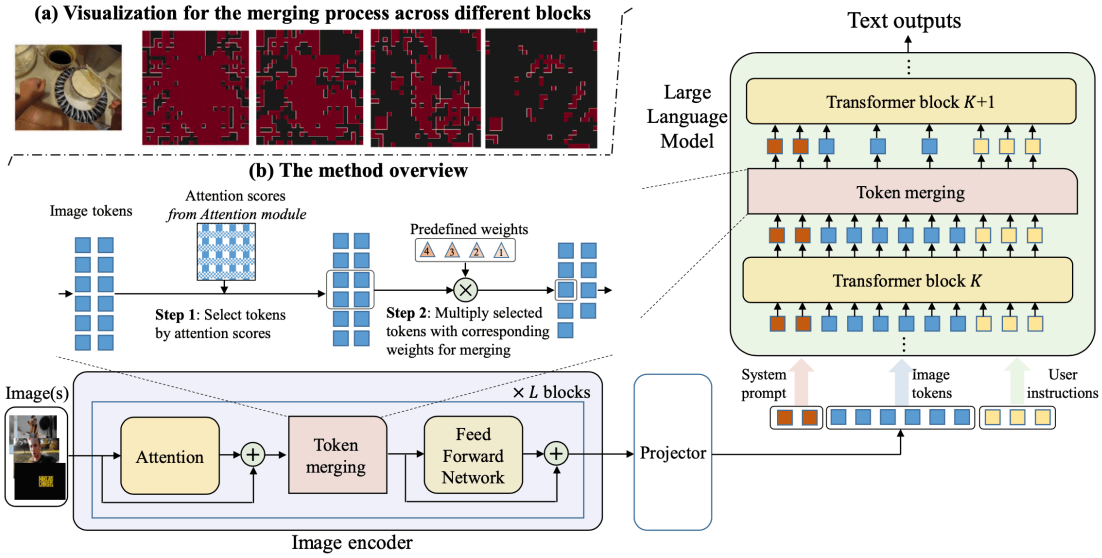


**Figure 3.** (a) Visualizations for the token merging process across different blocks in the image encoder. The red region denotes selected tokens and the black regions represent discarded tokens. (b) The framework overview for iLLaVA. Images are first sent into an image encoder to extract spatial features, followed by a projector. The projected image features are then concatenated with system prompts and user instructions and fed into the large language model to generate text outputs. We apply token merging at the intermediate layers of the image encoder and the language model to accelerate the forward pass.

#### *Token similarity*

Before merging tokens, we first which tokens are similar. As we have demonstrated in Sec. 3.1, attention scores from the attention module in the image encoder serve as a good indicator for revealing important tokens. Similarly, in the language model, we find the attention scores from the attention module also perform well in guiding the model to excavate context information. This motivates us to use the attention score as metrics for measuring token similarities. Another advantage of attention scores is that we don't need to perform re-calculation but could directly reuse the attention scores from the attention module from image encoders/language models. In the image encoder, given a pre-

computed attention map $A_v \in \mathbb{R}^{N_v \times N_v}$ with $N_v$ denoting the number of image tokens, we define the similarity $S_v^i$ between token $i$ and others as the averaged attention scores over all other tokens as:

$$S_v^i = \frac{1}{N_v - 1} \sum_{k=0}^{N_v, k \neq i} A_v^i. \tag{1}$$

In the language model, given a pre-computed attention map $A_t \in \mathbb{R}^{N_t \times N_t}$ with $N_t$ denoting the number of image tokens in the language model, we similarly define the similarity $S_t^i$ between token $i$ and others as the averaged attention scores over other image tokens as:

$$S_t^i = \frac{1}{N_t - 1} \sum_{k=0}^{N_t, k \neq i} A_t^i. \tag{2}$$

A slight difference for the language model is that we find using the image-to-image attention scores gives slightly better performance than adopting text-to-image attention scores.

### *Token merging.*

We apply the same token merging strategy with different merging steps for the image encoder and language model. Fig. 2(a) visualizes the merging process in LVLMs. In the image encoder of $L$ blocks, we select $L_v$ blocks and gradually merge tokens by $r_v$ per layer. Note that $r_v$ is a quantity instead of a ratio. In the language model, inspired by previous methods[12], we only perform token merging at one specific layer $K$ and reduce the token number by a constant $r_t$. Thus, assuming the input tokens are $N$, we finally only keep $N - r_v \times L_v - r_t$ tokens (e.g., 729 - 4×92 - 108 = 253 tokens, around $\frac{1}{3}$ input tokens).

An ideal token merging algorithm should meet two requirements, fast and accurate. It should be computed as fast as possible to avoid incurring any additional inference latency, and be as accurate as possible to avoid losing beneficial information. A naive solution is to conduct greedy merging by gradually merging tokens with the highest similarity until meeting the required number. However, we find in practice that repeatedly conducting the $\mathrm{Argmax}$ function in the greedy merging process over a matrix with thousands of rows and columns to find the elements of highest similarities causes severe time delay, which even occupies $\frac{1}{3}$ times of the whole inference time. This motivates us to find a token merging algorithm to be as simple and robust as possible.

Taking the image encoder as an example, we divide the tokens into two subsets $P_{unmerged}$ with length of $N_v - (r_v + 1)$ and $P_{merged}$ with a length of $r_v + 1$, according to attention scores received from other tokens. We apply merging operations for $P_{merged}$ only. Unlike gready merging that performs repeated merging operations, we finish the merging process within one operation. The operations are shown as follows. We first reorder the tokens in $P_{merged}$ by their similarity, with tokens receiving the least attention at the start of the queue. As our goal is to keep the tokens receiving the most attention, we should gradually merge the tokens with the least attention one by one until meeting the required number. We here take a linear schedule by gradually merging the first token with the second one from the start of $P_{merged}$ one by one. As the merging process is deterministic after obtaining the attention scores for tokens in $P_{merged}$, this process can be simplified to a simple multiplication between tokens in $P_{merged}$ with a pre-calculated

weight list $W_v \in \mathcal{R}^{r_v+1} = \{r_v + 1, r_v, \ldots, 1\}$. Thus, we can finish it by a simple multipication once to obtain $P_{new} \in \mathcal{R}^1 = W_v \times P_{merged}$, with quite few computations of only $\mathcal{O}(1)$ computational complexity. We concatenate $P_{new}$ with $P_{unmerged}$ to form the pruned token sequences $[P_{unmerged}, P_{new}]$, with length of $N_v - r_v$. Note that this is not equal to the greedy merging algorithm, but we find it performs already well in practice. We apply it for token merging in both the image encoder and language model.

# 4. Main Results

## *4.1. Model Settings*

Unless otherwise stated, we use LLaVA-Onevision 7B[1] as the base model, and gradually merge $r_v = 92$ tokens for Layer 3, 4, 5 and 6 with $N_v = 4$ in the image encoder, and merge $r_t = 108$ tokens for Layer $K = 2$ in the language model. After merging, given 729 input tokens, the intermediate layers in the image encoder just process 361 tokens while the intermediate layers of language model only process 253 tokens, around $\frac{1}{3}$ times the input tokens.

## *4.2. Model Efficiency*

We first validate the effectiveness of iLLaVA by improving model efficiency across different model scales. In Fig. 4, we plot the performance-throughput curve for 7B-size models including LLaVA-Onevision 7B[1] and QWen2-VL 7B[2], and larger models including LLaVA-Next 13B[45] and LLaVA-Next 34B[45]. We respectively select single-image benchmarks including MMMU[17], MME[41], NoCaps[46], Flickr30k[47] and video benchmarks including Egoschema for comparison. The metrics used for comparison include accuracy (Primary Axis), memory usage (Secondary Axis) and throughput (X-Axis) We observe that across benchmarks of different settings, iLLaVA could consistently be around 2$\times$ the throughput and reduce the memory usage by around 1.7$\times$-2.0$\times$ for models of different sizes with competitive performance.

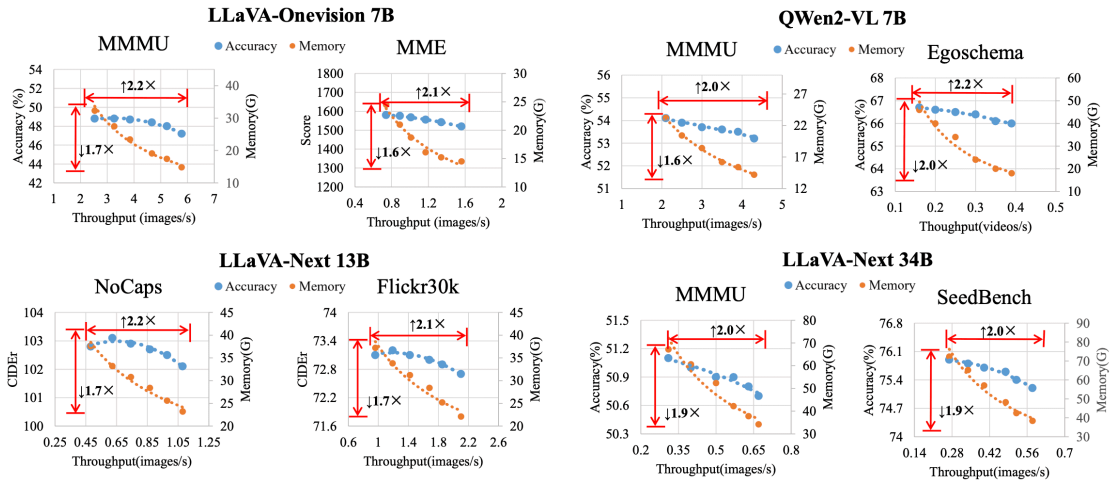


**Figure 4.** The performance-throughput curves by deploying iLLaVA upon 7B-size models including LLaVA-Onevision 7B and QWen2-VL 7B, and larger models including LLaVA-Next 13B and LLaVA-Next 34B, respectively. It can be observed that across benchmarks of different settings, iLLaVA could 2× the throughput without severe performance sacrifice.

| Model | Single-image benchmarks | | | | | | | Multi-image benchmarks | |
|---|---|---|---|---|---|---|---|---|---|
| | AI2D | MMBench | MME | MMMU | MMStar | MMVet | SeedBench | MuirBench | Q-Bench |
| | test | en-dev | test | val | test | test | image | test | test |
| LLaVA-OneVision 7B | 81.4 | 80.8 | 418/1580 | 48.8 | 61.7 | 57.5 | 75.4 | 41.8 | 74.5 |
| iLLaVA 7B | 78.6 (-2.8%) | 79.9 (-0.9%) | 423/1566 | 48.4 (-0.4%) | 59.0 (-2.7%) | 57.2 (-0.3%) | 73.6 (-1.8%) | 41.1 (-0.7%) | 74.2 (-0.3%) |
| QWen2-VL 7B | - | 81.7 | - | 54.0 | 60.5 | 62.0 | - | - | - |
| iLLaVA 7B | - | 81.2 (-0.5%) | - | 53.6 (-0.4%) | 59.1 (-1.4%) | 61.5 (-0.5%) | - | - | - |

**Table 1.** Results of deploying iLLaVA upon LLaVA-Onevision 7B and QWen2-VL 7B over single-image/multi-image benchmarks.


doi.org/10.32388/69KFLK

| Model | Video benchmarks | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | ActivityNet-QA | Egoschema | MLVU | NextQA | PerceptionTest | SeedBench | VideoChatGPT | VideoDC | VideoMME |
| | test | test | m-avg | mc | val | video | test | test | wo/w-subs |
| LLaVA-OneVision 7B | 56.6 | 60.1 | 64.7 | 79.4 | 57.1 | 56.9 | 3.51 | 3.75 | 58.2/61.5 |
| iLLaVA 7B | 56.4 (-0.2%) | 60.2 (+0.1%) | 64.4 (-0.3%) | 79.0 (-0.5%) | 56.8 (-0.1%) | 56.5 (-0.4%) | 3.50 (-0.01) | 3.73 (-0.02) | 58.2/61.4 (-0.0%/-0.1%) |
| QWen2-VL 7B | - | 66.7 | - | - | 62.3 | - | - | - | 63.2/68.8 |
| iLLaVA 7B | - | 66.3 (-0.3%) | - | - | 61.8 (-0.5%) | - | - | - | 62.9/68.5 (-0.3%/-0.3%) |

**Table 2.** Experimental results by deploying iLLaVA upon LLaVA-Onevision 7B and QWen2-VL 7B over video benchmarks. Here, we abbreviate PerceptionTest as PTest and Video Detail Description as VideoDC.

### *4.3. Broader Validation*

We further validate the effectiveness of iLLaVA upon a much wider range of benchmarks by deploying it upon state-of-the-art open-source LVLMs, LLaVA-Onevision 7B[1] and QWen2-VL 7B[2]. The benchmarks are composed of single-image benchmarks including AI2D[48], MMBench[22], MME[41], MMMU[17], MMStar[27], MMVet[49], SeedBench (Image subset)[26], multi-image benchmarks including MuirBench[42] and Q-Bench[50] in Tab. 1, and video benchmarks including ActivityNet-QA[44], Egoschema[18], MLVU[51], NextQA[52], PerceptionTest[53], SeedBench (Video subset)[26], VideoChatGPT[54], Video Detailed Descriptions (VideoDC)[55] and VideoMME[43] in Tab. 2. From Tab. 1 and Tab. 2, we can observe that in most cases iLLaVA could achieve competitive performance with LLaVA-Onevision 7B with only around 0.2%-0.5% performance loss, while in practice it could bring around 2$\times$ the throughput and nearly reduce the memory consumption by half.

### *4.4. Comparison with smaller models*

We deploy iLLaVA upon LVLMs and compare its performance and throughput against a LVLM of a smaller size. In the upper part of Fig. 5, we deploy iLLaVA upon LLaVA-Next 13B and compare it with a smaller LLaVA-Next 7B model, plotting the performance-throughput curves over MMMU and SeedBench benchmarks. We find that iLLaVA upon a 13B model achieves much better performance-throughput trade-off than a 7B model. iLLaVA outperforms LLaVA-Next 7B by around 2.0% and 2.2% on MMMU and SeedBench, respectively, with higher throughputs. In the bottom

part of Fig. 5, we deploy iLLaVA upon QWen2-VL 7B[2] and compare it with QWen2-VL 2B, plotting the performance-throughput curves over MMMU and MMStar benchmarks. iLLaVA demonstrates clear advantages than QWen2-VL 2B by outperforming it with 15% and 16% in accuracy upon two benchmarks, and better throughputs.

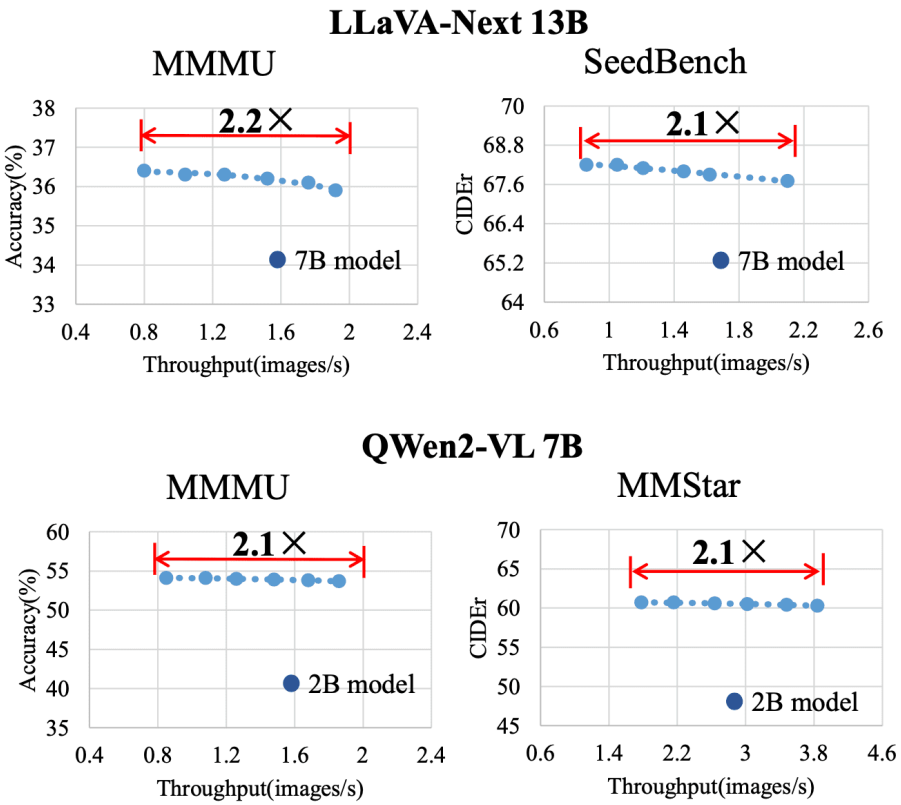


**Figure 5.** Deploying iLLaVA over LVLMs and comparing it against a LVLM with a smaller size. The upper part: deploying iLLaVA upon LLaVA-Next 13B and comparing it with LLaVA-Next 7B. The bottom part: deploying iLLaVA upon QWen2-VL 7B and comparing it with QWen2-VL 2B.

### *4.5. Comparison between iLLaVA and Other Efficient Multi-modal Methods*

In Tab. 3, we compare iLLaVA with other efficient multi-modal methods by deploying them on a LLaVA-Onevision 7B[1] backbone. These methods keep similar number of tokens after pruning/merging for fair comparison. We notice that iLLaVA achieves much higher performance while keep comparative or better throughput and memory usage than these candidates, validating its effectiveness.

| Methods | Accuracy(%) | Throughput(%) | Memory(%) |
|---|---|---|---|
| FastV | 47.6 | 4.20 | **19.1** |
| ToMe | 46.8 | **4.22** | **19.1** |
| LLaVA-PruMerge | 47.4 | 4.18 | 19.2 |
| Free-VideoLLM | 47.2 | 3.82 | 22.3 |
| iLLaVA | **48.4** | **4.22** | **19.1** |

**Table 3.** Comparison between iLLaVA based on LLaVA-Onevision 7B[1] with other efficient multi-modal methods over accuracy, throughput and memory on the MMMU benchmark.

### *4.6. Model Choices*

We report the averaged performance across MMMU, MMBench and MuirBench as evaluation metrics.

| Style | Methods | Acc (%) | Inference time |
|---|---|---|---|
| Pruning | Attention-based | 56.6 | **4h10min** |
| | Random | 42.1 | **4h10min** |
| Token merging | Greedy merging | **57.8** | 5h40min |
| | Bipartite matching | 55.2 | **4h10min** |
| | Kmeans (5 iters) | 55.7 | 4h30min |
| | CLIP | 53.2 | 4h50min |
| | CLIPSeg | 53.2 | 10h30min |
| | iLLaVA | 57.6 | **4h10min** |

**Table 4.** Token merging strategy in the image encoder.


doi.org/10.32388/69KFLK

| Kept tokens | Acc (%) | Inference time |
|---|---|---|
| 729 (Original) | 57.6 | 7h40min |
| 576 | **57.6** | 4h24min |
| 425 | **57.6** | 4h24min |
| 361 | **57.6** | 4h10min |
| 289 | 57.3 | 4h |
| 225 | 57.1 | 3h45min |
| 169 | 56.7 | 3h30min |

**Table 5.** The number of kept tokens after the image encoder.

| Layers | Acc (%) | Inference time |
|---|---|---|
| {1} | 55.2 | **4h08min** |
| {5} | 56.4 | **4h08min** |
| {2,3} | 56.8 | 4h10min |
| {7,8} | 56.8 | 4h09min |
| {2,3,4,5} | 57.4 | 4h09min |
| {5,6,7,8} | **57.6** | 4h10min |
| {7,8,9,10} | 57.5 | 4h11min |

**Table 6.** Ablations on the layers to apply token merging in the image encoder.

### *Token merging strategy in the image encoder*

In Tab. 4, we ablate the token merging strategy in the image encoder. The approaches used for comparison are divided into pruning strategies, including attention-based and random pruning, and merging strategies, including greedy merging, bipartite matching, kmeans merging. We additionally include methods that adopt CLIP[5] (denoted as CLIP) and CLIPSeg[56] (denoted as CLIPSeg) to determine the most significant areas corresponding to input texts, which crop the corresponding region and feed it into the LVLM for processing. Notably, we find that token merging approaches achieve better results than pruning approaches, among which attention-based pruning beats random

pruning largely. For token merging approaches, greedy merging achieves the best performance. However, it consumes too much time during the merging process and severely lags the inference process. Our method achieves competitive results with it but is much faster. Bipartite matching performs well in token merging in ViTs[14], but doesn't offer promising results in LVLMs. We speculate that it may not fully attend to the required contexts from inputs. Adopting kmeans with 5 iterations for merging slightly slows down the inference procedure, and also causes performance drop. Finally, we try using CLIP or CLIPSeg to locate the meaningful areas and crop the corresponding region for LVLMs. However, we find that they both don't achieve promising performance. This may be attributed to that they can only perform well in word-level understanding but are not skilled in understanding a complex sentence, thus failing to locate beneficial regions.

### *Kept tokens after the image encoder*

We explore the relationships between the number of kept tokens and performance in the image encoder in Tab. 5. Given 729 input image tokens, we gradually decrease the kept tokens from 425 to 169 and test the performance and inference time. We notice decreasing kept tokens until 361 doesn't hurt the performance, but drastically lowers the inference time by around half. Further reducing the kept tokens consistently decreases the inference time but causes a slight performance drop. We set the kept token as 361 by default.

### *Which layers to apply token merging in the image encoder*

We ablate the layers to apply token merging in the image encoder in Tab. 6. We find that directly completing token merging at a layer once would hurt the performance much. By maintaining the overall reduced token number, we gradually increase the number of layers for token merging and consistently observe better performance. When we apply token merging with four layers, we find the combination of {3,4,5,6} layers performs slightly better than other combinations. Generally, we find that conducting token merging with different layer combinations owns inference time, and set {3,4,5,6} layers for token merging by default.

| Style | Methods | Acc (%) | Inference time |
|---|---|---|---|
| Pruning | FastV | 57.2 | **4h10min** |
| | Random | 48.4 | **4h10min** |
| Token merging | Greedy merging | **57.8** | 4h50min |
| | Bipartite matching | 56.2 | **4h10min** |
| | Kmeans (5 iters) | 56.8 | 4h17min |
| | iLLaVA | 57.6 | **4h10min** |

**Table 7.** Ablations for the token merging strategy in the language model.

| Kept tokens | Acc (%) | Inference time |
|---|---|---|
| 361 (Original) | **57.6** | 4h33min |
| 300 | 57.4 | 4h21min |
| 253 | 57.4 | 4h10min |
| 225 | 57.2 | 3h58min |
| 200 | 57.1 | 3h50min |
| 169 | 56.8 | 3h40min |

**Table 8.** The number of kept tokens within the language model.

| Layers | Acc (%) | Inference time |
|---|---|---|
| {2} | 57.4 | **4h09min** |
| {5} | 57.5 | 4h09min |
| {8} | **57.6** | 4h10min |
| {10} | 57.4 | 4h10min |
| {5,6} | 56.9 | 4h10min |
| {8,9} | 57.0 | 4h10min |

**Table 9.** Ablations on the layers to apply token merging in the language model.

### *Token merging strategy in the language model*

In Tab. 7, we ablate the token merging strategy used for the language model. We compare iLLaVA with pruning strategies including FastV[12] and random pruning, and token merging strategies including greedy merging, bipartite matching and kmeans (5 iterations). Similarly, we find greedy merging achieves the best results with much longer inference time. As a pruning method, FastV performs superiseingly well and beats most token merging methods. Thanks to the information recycling design, iLLaVA achieves the best performance-latency trade-off among these candidates.

### *Kept tokens within the language model*

With 361 kept tokens after the image encoder, we explore how many tokens to keep for the intermediate layers in the language model in Tab. 8. With 361 input image tokens, we gradually decrease the kept tokens from 300 to 169. We

notice decreasing tokens would consistently lead to performance drop, and the performance decrease is more obvious as kept tokens decrease. We speculate that after token merging in the image encoder, the most remained image tokens contain beneficial information for understanding the image, which are hard to prune. Considering the performance-latency trade-off, we set the kept tokens as 253.

### *The layer to apply token merging in the language model*

We ablate the layer(s) to apply token merging in the language model in Tab. 9. Generally, we find directly performing token merging at a layer once offers better results than over multiple layers. Conducting token merging at different layers owns similar inference time, we adopt the 2nd one by default for its better performance.

| Approaches | DocVQA (%) | InfoVQA (%) | ChartQA (%) |
|---|---|---|---|
| LLaVA-Onevision 7B | 87.5 | 68.8 | 80.0 |
| iLLaVA | 79.5 (-8.0%) | 56.8 (-12.0%) | 69.8 (-10.2%) |

**Table 10.** Performance of iLLaVA on DocVQA[25], InfoVQA[24] and ChartQA[23].

## *4.7. Visualizations*

To better understand the token merging process in iLLaVA, we visualize the selected image tokens for single-image benchmarks, multi-image benchmarks and video benchmarks in Fig. 6, respectively. Over each benchmark, we select three images and visualize the raw image, selected tokens from the image encoder and selected tokens from the language model in order. We notice that both the image encoder and language model can well attend to the important information from raw images, like the words in the first row and the person in the last row. We also have an observation that the language tends to allocate more tokens to images closer to the output texts, and pay less attention to images earlier input into the language model. This may affect the behavior of LVLMs in memorying long contexts.

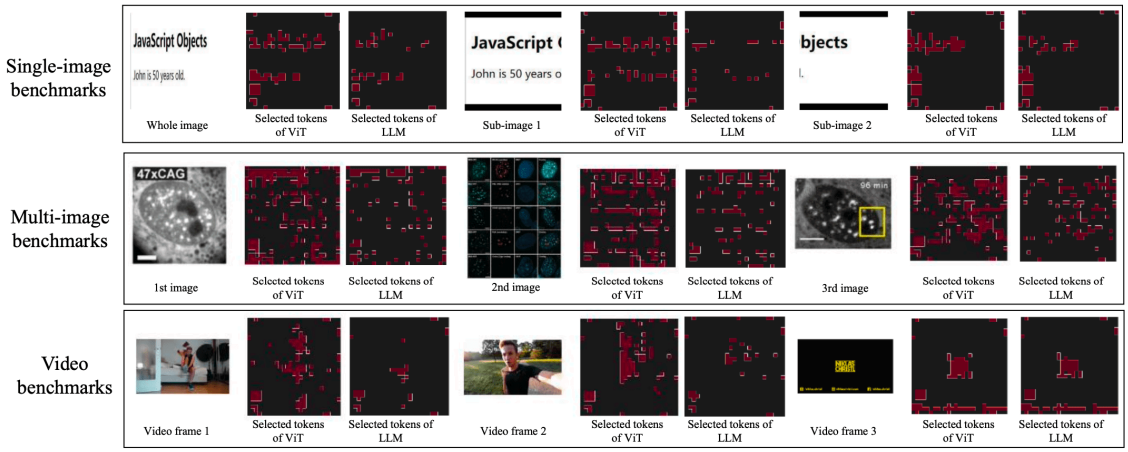


**Figure 6.** Visualizations for selected tokens over single-image benchmarks, multi-image benchmarks and video benchmarks. Three images are visualized for each benchmark, with the raw image on the left, selected tokens from the image encoder in the middle and selected tokens from the language model on the right, respectively.

## 5. Limitations

While we have validated that iLLaVA performs well across a wide range of benchmarks, we find that upon benchmarks that require detailed image context understanding, conducting token merging may lead to performance drop. For example, as shown in Tab. 10, iLLaVA leads to consistent performance decrease on DocVQA, InfoVQA and ChartQA which require a large quantity of image tokens to perform detailed document understanding. In this case, reducing token numbers would inevitably hurt the performance.

## 6. Conclusion

In this paper, we revisited the existence of feature redundancy in Large Vision-Language Models (LVLMs) and utilizeed it to accelerate the inference procedure. Compared to previous methods, our method is characterized by two key designs. First, instead of only conducting token pruning/merging within the language model, we accelerated both the forward pass of the image encoder and language model in LVLMs, which offers high efficiency. Second, unlike previous methods that simply prune the unnecessary tokens, we recycled the beneficial information from them to avoid incurring performance drop. Experimental results across different model sizes over single-image, multi-image and video benchmarks validated the effectiveness and generalizability of our method.

## References

1. a, b, c, d, e, f, g, h *Li B, Zhang Y, Guo D, Zhang R, Li F, Zhang H, Zhang K, Li Y, Liu Z, Li C (2024). "Llava-onevision: Easy visual task transfer". arXiv preprint arXiv:2408.03326.*

2. a, b, c, d, e, f Wang P, Bai S, Tan S, Wang S, Fan Z, Bai J, Chen K, Liu X, Wang J, Ge W, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. arXiv preprint arXiv:2409.12191. 2024.
3. ^ Dzabraev M, Kalashnikov M, Komkov S, Petiushko A. Mdmmt: Multidomain multimodal transformer for video retrieval. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021. p. 3354-3363.
4. ^ Kim W, Son B, Kim I (2021). "Vilt: Vision-and-language transformer without convolution or region supervision". In: International Conference on Machine Learning. PMLR. pp. 5583–5594.
5. a, b Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, Agarwal S, Sastry G, Askell A, Mishkin P, Clark J, et al. Learning transferable visual models from natural language supervision. In: International conference on machine learning. PMLR; 2021. p. 8748–8763.
6. ^ Gabeur V, Sun C, Alahari K, Schmid C. Multi-modal transformer for video retrieval. In: Computer Vision--ECCV 2020: 16th European Conference, Glasgow, UK, August 23--28, 2020, Proceedings, Part IV 16. Springer; 2020. p. 214--229.
7. ^ Li LH, Zhang P, Zhang H, Yang J, Li C, Zhong Y, Wang L, Yuan L, Zhang L, Hwang JN, et al. Grounded language-image pre-training. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022. p. 10965-10975.
8. a, b Alayrac JB, Donahue J, Luc P, Miech A, Barr I, Hasson Y, Lenc K, Mensch A, Millican K, Reynolds M, et al. (2022). "Flamingo: a visual language model for few-shot learning". Advances in Neural Information Processing Systems. 35: 23716–23736.
9. ^ Li J, Li D, Xiong C, Hoi S. "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation." In: International Conference on Machine Learning. PMLR; 2022. p. 12888-12900.
10. a, b Li J, Li D, Savarese S, Hoi S (2023). "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models". arXiv preprint arXiv:2301.12597. Available from: https://arxiv.org/abs/2301.12597.
11. ^ Xu M, Gao M, Gan Z, Chen HY, Lai Z, Gang H, Kang K, Dehghan A (2024). "Slowfast-llava: A strong training-free baseline for video large language models". arXiv preprint arXiv:2407.15841.
12. a, b, c, d, e, f, g Chen L, Zhao H, Liu T, Bai S, Lin J, Zhou C, Chang B. 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. Springer; 2024. p. 19-35.
13. ^ Han K, Guo J, Tang Y, He W, Wu E, Wang Y (2024). "Free Video-LLM: Prompt-guided Visual Perception for Efficient Training-free Video LLMs". arXiv preprint arXiv:2410.10441.
14. a, b Bolya D, Fu CY, Dai X, Zhang P, Feichtenhofer C, Hoffman J (2022). "Token merging: Your vit but faster". arXiv preprint arXiv:2210.09461.
15. a, b Chen X, Liu Z, Tang H, Yi L, Zhao H, Han S (2023). "Sparsevit: Revisiting activation sparsity for efficient high-resolution vision transformer". Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2061--2070.
16. a, b Shang Y, Cai M, Xu B, Lee YJ, Yan Y (2024). "Llava-prumerge: Adaptive token reduction for efficient large multimodal models". arXiv preprint arXiv:2403.15388.

17. a, b, c, d, eYue X, Ni Y, Zhang K, Zheng T, Liu R, Zhang G, Stevens S, Jiang D, Ren W, Sun Y, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In: Proceedings of the IEEE/CVF Conferen ce on Computer Vision and Pattern Recognition. 2024. p. 9556-9567.
18. a, b, cMangalam K, Akshulakov R, Malik J (2023). "Egoschema: A diagnostic benchmark for very long-form video langua ge understanding". Advances in Neural Information Processing Systems. 36: 46212–46244.
19. ^Ramachandram D, Taylor GW (2017). "Deep multimodal learning: A survey on recent advances and trends". IEEE Signa l Processing Magazine. 34 (6): 96–108.
20. ^Xu P, Zhu X, Clifton DA (2023). "Multimodal learning with transformers: A survey". IEEE Transactions on Pattern Analy sis and Machine Intelligence. 45 (10): 12113–12132.
21. ^Ochoa X, Lang AC, Siemens G (2017). "Multimodal learning analytics". The handbook of learning analytics. 1: 129–141.
22. a, b, cLiu Y, Duan H, Zhang Y, Li B, Zhang S, Zhao W, Yuan Y, Wang J, He C, Liu Z, et al. Mmbench: Is your multi-modal m odel an all-around player? In: European Conference on Computer Vision. Springer; 2024. p. 216-233.
23. a, bMasry A, Long DX, Tan JQ, Joty S, Hoque E (2022). "Chartqa: A benchmark for question answering about charts with v isual and logical reasoning". arXiv preprint arXiv:2203.10244. arXiv:2203.10244.
24. a, bMathew M, Bagal V, Tito R, Karatzas D, Valveny E, Jawahar CV. "Infographicvqa." In: Proceedings of the IEEE/CVF Wi nter Conference on Applications of Computer Vision. 2022. p. 1697--1706.
25. a, bMathew M, Karatzas D, Jawahar CV (2021). "Docvqa: A dataset for vqa on document images". Proceedings of the IEE E/CVF winter conference on applications of computer vision. pages 2200–2209.
26. a, b, c, dLi B, Ge Y, Ge Y, Wang G, Wang R, Zhang R, Shan Y (2024). "SEED-Bench: Benchmarking Multimodal Large Lang uage Models". Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024: 13299–1330 8.
27. a, bChen L, Li J, Dong X, Zhang P, Zang Y, Chen Z, Duan H, Wang J, Qiao Y, Lin D, et al. Are we on the right way for evalua ting large vision-language models? arXiv preprint arXiv:2403.20330. 2024.
28. ^Meng L, Li H, Chen BC, Lan S, Wu Z, Jiang YG, Lim SN (2022). "Adavit: Adaptive vision transformers for efficient image r ecognition." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12309--1231 8.
29. ^Pan Z, Zhuang B, He H, Liu J, Cai J (2022). "Less is more: Pay less attention in vision transformers". Proceedings of the A AAI Conference on Artificial Intelligence. 36 (2): 2035–2043.
30. ^Ryoo M, Piergiovanni AJ, Arnab A, Dehghani M, Angelova A (2021). "Tokenlearner: Adaptive space-time tokenization f or videos". Advances in Neural Information Processing Systems. 34: 12786–12797.
31. ^Rao Y, Zhao W, Liu B, Lu J, Zhou J, Hsieh C (2021). "Dynamicvit: Efficient vision transformers with dynamic token sparsi fication". Advances in neural information processing systems. 34: 13937–13949.
32. ^Goyal S, Choudhury AR, Raje S, Chakaravarthy V, Sabharwal Y, Verma A. "Power-bert: Accelerating bert inference via p rogressive word-vector elimination." In: International Conference on Machine Learning. PMLR; 2020. p. 3690-3699.

33. ^Kim G, Cho K (2020). "Length-adaptive transformer: Train once with length drop, use anytime with search". arXiv pre print arXiv:2010.07003.
34. ^Lassance C, Maachou M, Park J, Clinchant S (2021). "A study on token pruning for ColBERT". arXiv preprint arXiv:2112. 06540.
35. a, bLi J, Chen D, Cai T, Chen P, Hong Y, Chen Z, Shen Y, Gan C. "Flexattention for efficient high-resolution vision-languag e models." In: European Conference on Computer Vision. Springer; 2025. p. 286-302.
36. ^Wan Z, Wu Z, Liu C, Huang J, Zhu Z, Jin P, Wang L, Yuan L (2024). "Look-m: Look-once optimization in kv cache for effi cient multimodal long-context inference". arXiv preprint arXiv:2406.18139.
37. a, bHe Y, Chen F, Liu J, Shao W, Zhou H, Zhang K, Zhuang B (2024). "ZipVL: Efficient Large Vision-Language Models with Dynamic Token Sparsification and KV Cache Compression". arXiv preprint arXiv:2410.08584.
38. a, bArif KHI, Yoon J, Nikolopoulos DS, Vandierendonck H, John D, Ji B (2024). "HiRED: Attention-Guided Token Dropping for Efficient Inference of High-Resolution Vision-Language Models in Resource-Constrained Environments". arXiv prep rint arXiv:2408.10945.
39. ^Lan X, Yuan Y, Jie Z, Ma L (2024). "VidCompress: Memory-Enhanced Temporal Compression for Video Understanding i n Large Language Models". arXiv preprint arXiv:2410.11417. Available from: https://arxiv.org/abs/2410.11417.
40. a, bZhang R, Lyu Y, Shao R, Chen G, Guan W, Nie L (2024). "Token-level Correlation-guided Compression for Efficient M ultimodal Document Understanding". arXiv preprint arXiv:2407.14439.
41. a, b, cXu P, Shao W, Zhang K, Gao P, Liu S, Lei M, Meng F, Huang S, Qiao Y, Luo P (2023). "Lvlm-ehub: A comprehensive e valuation benchmark for large vision-language models". arXiv preprint arXiv:2306.09265.
42. a, bWang F, Fu X, Huang JY, Li Z, Liu Q, Liu X, Ma MD, Xu N, Zhou W, Zhang K, et al. MuirBench: A Comprehensive Bench mark for Robust Multi-image Understanding. arXiv preprint arXiv:2406.09411. 2024.
43. a, bFu C, Dai Y, Luo Y, Li L, Ren S, Zhang R, Wang Z, Zhou C, Shen Y, Zhang M, et al. Video-mme: The first-ever comprehe nsive evaluation benchmark of multi-modal llms in video analysis. arXiv preprint arXiv:2405.21075. 2024.
44. a, bYu Z, Xu D, Yu J, Yu T, Zhao Z, Zhuang Y, Tao D (2019). "Activitynet-qa: A dataset for understanding complex web vid eos via question answering". Proceedings of the AAAI Conference on Artificial Intelligence. 33 (01): 9127--9134.
45. a, bLiu H, Li C, Li Y, Li B, Zhang Y, Shen S, Lee YJ. "LLaVA-NeXT: Improved reasoning, OCR, and world knowledge". 2024 Jan. Available from: https://llava-vl.github.io/blog/2024-01-30-llava-next/.
46. ^Agrawal H, Desai K, Wang Y, Chen X, Jain R, Johnson M, Batra D, Parikh D, Lee S, Anderson P. "Nocaps: Novel object cap tioning at scale." In: Proceedings of the IEEE/CVF international conference on computer vision. 2019. p. 8948-8957.
47. ^Plummer BA, Wang L, Cervantes CM, Caicedo JC, Hockenmaier J, Lazebnik S (2015). "Flickr30k entities: Collecting regio n-to-phrase correspondences for richer image-to-sentence models". Proceedings of the IEEE international conference o n computer vision. 2641--2649.
48. ^Kembhavi A, Salvato M, Kolve E, Seo M, Hajishirzi H, Farhadi A. A diagram is worth a dozen images. In: Computer Visio n--ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11--14, 2016, Proceedings, Part IV 14. Springer; 2016. p. 235--251.

49. ^*Yu W, Yang Z, Li L, Wang J, Lin K, Liu Z, Wang X, Wang L (2023). "Mm-vet: Evaluating large multimodal models for int egrated capabilities". arXiv preprint arXiv:2308.02490. Available from: https://arxiv.org/abs/2308.02490.*

50. ^*Wu H, Zhang Z, Zhang E, Chen C, Liao L, Wang A, Li C, Sun W, Yan Q, Zhai G, et al. Q-bench: A benchmark for general-p urpose foundation models on low-level vision. arXiv preprint arXiv:2309.14181. 2023.*

51. ^*Zhou J, Shu Y, Zhao B, Wu B, Xiao S, Yang X, Xiong Y, Zhang B, Huang T, Liu Z (2024). "MLVU: A Comprehensive Bench mark for Multi-Task Long Video Understanding". arXiv preprint arXiv:2406.04264.*

52. ^*Xiao J, Shang X, Yao A, Chua TS. "Next-qa: Next phase of question-answering to explaining temporal actions". In: Proc eedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021. p. 9777–9786.*

53. ^*Patraucean V, Smaira L, Gupta A, Recasens A, Markeeva L, Banarse D, Koppula S, Malinowski M, Yang Y, Doersch C, et a l. Perception test: A diagnostic benchmark for multimodal video models. Advances in Neural Information Processing Syst ems. **36**, 2024.*

54. ^*Maaz M, Rasheed H, Khan S, Khan FS. "Video-chatgpt: Towards detailed video understanding via large vision and lang uage models". arXiv preprint arXiv:2306.05424. 2023.*

55. ^***Lmms-Lab** (2024). "Video detail caption."*

56. ^*Lüdecke T, Ecker A (2022). "Image segmentation using text and image prompts". Proceedings of the IEEE/CVF conferen ce on computer vision and pattern recognition. pp. 7086–7096.*

## Declarations

**Funding:** No specific funding was received for this work.

**Potential competing interests:** No potential competing interests to declare.