

# HiPrune: Hierarchical Attention for Efficient Token Pruning in Vision-Language Models

Jizhihui Liu<sup>1\*</sup>, Guangdao Zhu<sup>1\*</sup>, Feiyi Du<sup>1\*</sup>, Niu Lian<sup>1</sup>, Jun Li<sup>1,2</sup>

Bin Chen<sup>1,2†</sup>, Weili Guan<sup>1,2</sup>, Yaowei Wang<sup>1,2</sup>

<sup>1</sup>Harbin Institute of Technology, Shenzhen <sup>2</sup>Pengcheng Laboratory  
danielement321@gmail.com, chenbin2021@hit.edu.cn

## Abstract

Vision-Language Models (VLMs) encode images and videos into abundant tokens, which contain substantial redundancy and computation cost. While visual token pruning mitigates the issue, most existing methods lack insight into the intrinsic property of the vision encoder itself. In this work, we dive into the vision encoder and prove that the middle layers pay more attention to the main objects of the image qualitatively and quantitatively, while the deep layers to tokens with rich global information. Utilizing this **H**ierarchical attention pattern, we propose **HiPrune**, a training-free and model-agnostic token **P**runing method. HiPrune identifies three types of visual tokens according to their attention in different phases of the vision encoder, which preserves different levels of information. By coupling with the similarity of text tokens, we propose a prompt-aware variance, **HiPrune++**, which further improves instruction following performance under a very low token budget. Extensive experiments across four representative VLMs show that HiPrune achieves up to 99.3% of task accuracy with only 1/3 of the tokens, while reducing inference FLOPs by 58.7%. HiPrune++ maintains up to 99.7% accuracy with 2/9 tokens, highlighting robustness under high-resolution. Our code is available at <https://github.com/Danielement321/HiPrune>.

## 1 Introduction

Built on the success of Large Language Models (LLMs) (Touvron et al., 2023; Yang et al., 2025a), Vision-Language Models (VLMs) (Team et al., 2023; Hurst et al., 2024; Liang et al., 2024) have demonstrated considerable capacity in various visual tasks. VLMs commonly comprise a vision encoder (Radford et al., 2021; Zhai et al., 2023), an adaptor, and an LLM. The vision encoder is a

<sup>1</sup>Equal Contribution.

<sup>2</sup>Corresponding Author.

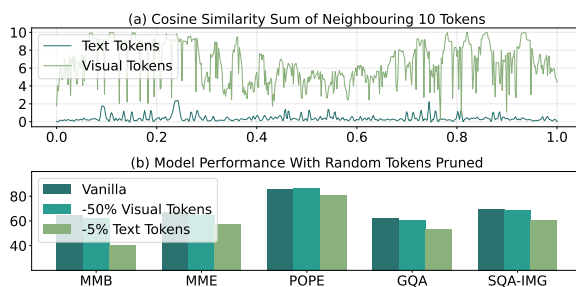


Figure 1: **Redundancy analyses on visual tokens.** (a) The sum of cosine similarity between each token and neighbouring 10 tokens. (b) The performance of LLaVA-1.5-7B when randomly removing 50% visual tokens or 5% text tokens.

vision transformer (ViT) (Dosovitskiy et al., 2021; Vaswani et al., 2017) that encodes the image into numerous tokens, which account for the biggest proportion of inputs and cause significant latency and memory demands. In LLaVA-1.5 (Liu et al., 2023, 2024a), an image is encoded into 576 tokens, much longer than its textual counterparts (Zhang et al., 2025c). For VLMs that incorporate a native dynamic-resolution encoder (Bai et al., 2025), one high-resolution webpage snapshot may require more than 10,000 tokens, resulting in a substantial computational cost and GPU memory allocation.

Although visual tokens constitute the majority of VLM input sequences, their necessity remains questionable. In Fig. 1(a), we compare the cosine similarity between each token and its 10 neighbors and find that visual tokens exhibit significantly higher redundancy than text tokens. Fig. 1(b) shows that randomly pruning 50% of visual tokens causes a performance drop much smaller than removing merely 5% of text tokens, while yielding substantial reductions in FLOPs. Moreover, previous works (Chen et al., 2024a; Zhang et al., 2025f) observe that visual tokens receive markedly less attention in the LLM decoder compared to text tokens. These findings point to a key insight: vi-

sual tokens are highly redundant. Based on this, many works seek to reduce the number of visual tokens to overcome the computational burden. Some methods (Chen et al., 2024a; Xing et al., 2025; Zhang et al., 2025g) conduct token pruning inside the LLM decoder, while some (Yang et al., 2025b; Alvar et al., 2025) conduct token selection based on static metrics like diversity or similarity. However, most methods do not fully utilize the intrinsic attention property of the vision encoder and are model-sensitive, necessitating careful tuning for practical deployment.

In this paper, we show that vision encoders process visual information in a progressive and structured hierarchy, where different layers attend to distinct semantic levels. Specifically, middle layers predominantly capture object-centric features, while deeper layers encode global contextual representations. This hierarchical pattern is consistently observed across various vision encoders (Radford et al., 2021; Zhai et al., 2023; Touvron et al., 2021; Assran et al., 2025), regardless of the architecture design or pre-training data. Building on this observation, we introduce **HiPrune**, a training-free and model-agnostic visual token pruning plugin. We extract attention maps from a designated object layer  $l$ , selecting tokens with the highest attention scores and tokens near them as **Anchor Tokens** and **Buffer Tokens**, which encode detailed local semantics. The remaining token budget is allocated to **Register Tokens**, selected by the attention scores in the output layer, capturing global and holistic contextual features. We further introduce an optional subset of visual tokens selected based on similarity with text tokens, which showcases an improved instruction-following ability (HiPrune++).

We evaluate our method across multiple popular VLMs. When applied to LLaVA-1.5, HiPrune maintains **99.3%** of original performance while requiring only **1/3** tokens, alongside bringing a **58.7%** FLOPs reduction. With a tight budget of **1/9** tokens, HiPrune++ still preserves **96.1%** accuracy score, accompanied by an outstanding hallucination reduction compared with baselines. With a different encoder and dynamic token length setting, HiPrune achieves state-of-the-art on Qwen, confirming its versatility on various architectures.

Our main contributions are as follows:

- We analyse the hierarchical attention patterns in vision encoders and reveal the focus of various layers qualitatively and quantitatively.

- We propose HiPrune and HiPrune++, a training-free and model-agnostic visual token pruning plugin enabling efficient inference.
- Extensive experiments on VLMs demonstrate the excellence of HiPrune, preserving **99.3%** performance with only **1/3** visual tokens and reducing inference FLOPs by up to **58.7%**.

## 2 Related Works

### 2.1 Vision-Language Models

VLMs (Bai et al., 2023; Wang et al., 2023; Chen et al., 2024b; Zhang et al., 2025b) have achieved impressive performance in various multimodal tasks. These models are typically composed of an image encoder, a projector, and an LLM. In the input sequence, visual tokens often constitute a significant portion. To improve the model’s capacity for fine-grained understanding, some Vision-Language models, such as LLaVA-NeXT (Liu et al., 2024b) and LLaVA-UHD (Guo et al., 2024), increase the resolution of the input image, which further raises the number of visual tokens in the sequence. The excessive number of visual tokens leads to considerable computational overhead and adversely affects the inference speed of the model, constraining its practical deployment. This motivates the need for visual token compression techniques that reduce redundancy while preserving model performance.

### 2.2 Visual Token Compression Methods

Most visual token compression methods employ a pruning or merging strategy. FastV (Chen et al., 2024a) is a representative pruning method that compresses visual tokens by discarding those with low attention scores in the LLM. Following this work, some methods (Xing et al., 2025; Song et al., 2025; Hu et al., 2025) leverage text-image attention to prune tokens, which brings extra computational overhead. In transformer (Vaswani et al., 2017), the attention score controls the information flow from layer to layer. Based on this theory, many approaches (Zhang et al., 2025d; Arif et al., 2025) overcome this by the attention from CLS token. However, not every vision encoder features such a token (e.g., SigLIP (Zhai et al., 2023)), which constrains the adaptation of these methods for universal VLMs (Bai et al., 2025; Chen et al., 2024b). Apart from the drop strategy, merging-based approaches aim to reduce redundancy by fusing similar visual tokens. ToMe (Bolya et al., 2023) performs token merging by aggregating visual tokens

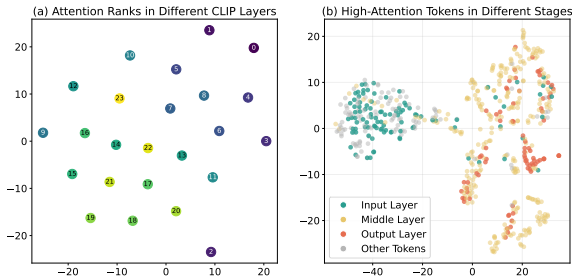


Figure 2: **Hierarchical attention pattern in CLIP.** (a) Attention rankings of different layers. Most layers are adjacent to their neighbouring layers. (b) Top 50% attentive tokens from different CLIP layers. The attention shifts from one cluster to another, showing a gradual and continuous transition, bridged by the middle-layer attention. Please refer to the Appendix B for details.

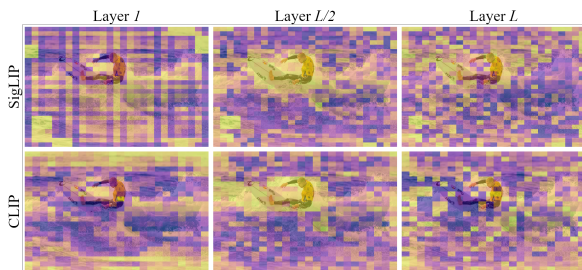


Figure 3: **Attention map for different layers of SigLIP and CLIP.** Patches with higher scores are in yellow. We can see that the middle layer is more object-centric.

with high feature similarity. Subsequently, several studies (Cao et al., 2023; Yang et al., 2025b; Huang et al., 2025) have explored hybrid pruning-and-merging strategies. Most merging methods need extra training or functions, since merging is usually gradual across multiple layers and not compatible with FlashAttention (Dao et al., 2022). In this work, we show that pruning tokens purely based on the vision encoder’s inherent hierarchical attention pattern can achieve outstanding results without special tokens and unnecessary complexity.

### 3 Motivated Insights

**What Is the Representation Structure Inside Vision Encoders?** To understand the focusing mechanism of vision encoders, we first take an insight into how attention distribution shifts across different layers. As shown in Fig. 2(a), we project the attention scores of different layers with t-SNE (Maaten and Hinton, 2008). The distribution reveals a continuous trajectory where most adjacent layers happen to be near each other, indicating the attention shifts across layers in a progressive and

Layer	CLIP-L	CLIP-B	SigLIP	SigLIP2	DeiT	VJEPa2
1	0.58×	0.34×	0.57×	0.62×	0.27×	0.82×
$L/2$	1×	1×	1×	1×	1×	1×
$L$	0.80×	0.79×	0.66×	0.64×	0.59×	0.26×

Table 1: **IoU of object segmentation mask and top 10% high-attention tokens.** Higher values stand for more overlap on objects in the image. ‘ $L$ ’ denotes the total layers in the encoder. The data is normalized for a better comparison.

ordered way. We label tokens with high attention from different layers in Fig. 2(b). From the input to the output, the attention transfers from one cluster to another cluster, with the middle layer attention bridging both clusters. This continuous transition proves **the existence of an ordered representation hierarchy in the vision encoder.**

**What do Middle Layers Focus on?** We visualize the attention map for CLIP and SigLIP in Fig. 3. The attention distribution in the middle layers exhibits a distinct pattern: the model focuses on the main object of the image, e.g., the surf-man. To confirm this empirical observation, we compute the IoU between the object segmentation mask and top 10% high-attention tokens in Table 1 using the COCO val2017 dataset (Lin et al., 2014). The results show that the high-attention tokens from the middle layer share more overlap with objects than the input or output layer, indicating that **the attention from the middle layers is more correlated to objects in the image.** Notably, this tendency occurs across various encoders, including world models (Assran et al., 2025), showing little correlation with model architecture or training data.

**What do Deep Layers Focus on?** Previous works have argued that tokens receiving high attention scores in the deep layer of ViT encode rich global information by conducting image classification tasks on these tokens (Darcet et al., 2024). In Fig. 3, we can see that in the output layer, the high-attention tokens diffuse across the whole image. Despite showing little correlation with the object, these tokens include patches of the image uniformly and can serve as an ideal indicator of the image under a limited token budget. Therefore, we can conclude that **tokens receiving high attention in the output layer encode global information.**

## 4 Method

An overview of our method is depicted in Fig. 4 and pseudo-code is given in Appendix A. In the following, we first revisit the self-attention, then we present the design of HiPrune and HiPrune++.

### 4.1 Preliminaries

In a ViT-based vision encoder, an image is encoded into multiple tokens, forming a visual token matrix  $\mathbf{T}_v \in \mathbb{R}^{N \times d}$ , where  $N$  denotes the number of patches in an image and  $d$  denotes the hidden dimension of the model. In each layer, the tokens are first mapped into  $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ , and subsequently, the attention matrix  $\mathbf{A}$  is computed by

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right) \in \mathbb{R}^{H \times N \times N}. \quad (1)$$

The information sharing between tokens only takes place in Eq. 1. Intuitively, the more ‘important’ a token is, the more its value in every token’s new state, which is assigned by  $\mathbf{A}$ . Therefore, in layer  $l$ , the importance of tokens can be weighted by their attention score:

$$\mathbf{a}^{[l]} = \frac{1}{H} \sum_{h=1}^H \sum_{n=1}^N \mathbf{A}^{[l]}[h, n, :], \quad (2)$$

$$= (a_1^{[l]}, a_2^{[l]}, \dots, a_N^{[l]}) \in \mathbb{R}^N. \quad (3)$$

### 4.2 Retained Tokens

**Anchor tokens** denote tokens with the highest attention score in the middle layers of the vision encoder. As discussed in our motivated insights, the middle layers tend to focus on object features, as evidenced by higher attention scores for tokens related to the object. Based on this, anchor tokens encode rich, detailed information about the raw image and should be retained when pruning.

**Buffer tokens** indicate tokens spatially adjacent to anchor tokens. As indicated by previous studies (Yang et al., 2024), noise exists in the attention map of ViTs. In Fig. 3, despite most high-attention tokens concentrating on the surf-man, a few tokens diffuse among the image, which may mislead the anchor tokens. To mitigate the noise issue and preserve spatial relationship, we include tokens neighbouring the anchor as a buffer.

**Register tokens** receive top attention scores in the output layer of the vision encoder. In deep layers of the vision encoder, the high-attention tokens

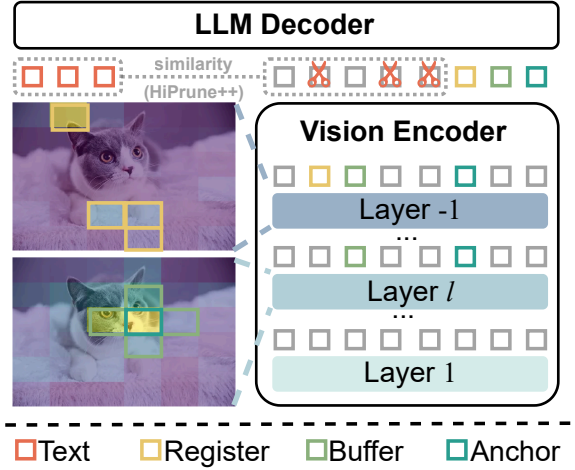


Figure 4: **Overview of HiPrune and HiPrune++.** HiPrune selects anchor and buffer tokens based on the attention from the object layer  $l$ , and register tokens from the last encoder layer. HiPrune++ additionally includes a small set of tokens selected by cosine similarity with text tokens to enhance instruction following ability.

distribute uniformly across the image, serving as an ideal indicator of global information (Darcet et al., 2024). To enhance the model’s overall understanding of an image, we supplement the token set with register tokens, which is a common practice in approaches for VLM token pruning (Zhang et al., 2025d; Arif et al., 2025).

### 4.3 Pruning Pipeline

Given a target token budget  $N'$ , we designate the object layer  $l$  and object proportion  $\alpha$ , denoting the layer from which to determine anchor and buffer tokens, and the added-up proportion of them. Taking the cross strategy as an example (4 buffers around 1 anchor), we first draw the attention score  $\mathbf{a}^{[l]}$  from the object layer  $l$  and calculate the number of anchor tokens by  $N_a = \lfloor \frac{\alpha \cdot N'}{5} \rfloor$ . The anchor token indices set  $\mathcal{I}_A$  is decided by

$$\mathcal{I}_A = \{i \mid |\{j \mid a_j^{[l]} > a_i^{[l]}\}| < N_a\}. \quad (4)$$

Once the anchor tokens are decided, we proceed to retain buffer tokens. It is worth noting that the buffer selection scheme can be diverse, but little difference exists between choices as long as the size is big enough, which is to be discussed in our ablation studies. Assuming an  $H_r \times W_c$  image is patchified into  $r \times c$  tokens, the buffer token indices  $\mathcal{I}_B$  is calculated by

$$\mathcal{I}_B = \cup\{\mathcal{I}_A - 1, \mathcal{I}_A + 1, \mathcal{I}_A - c, \mathcal{I}_A + c\} \cap [0, rc - 1]. \quad (5)$$

The register token selection occurs after anchor tokens and buffer tokens, given the attention score  $\mathbf{a}^{[-1]}$  of the output layer of the vision encoder, the registers are selected by

$$\mathcal{I}_R = \{i \mid |\{j \mid a_j^{[-1]} > a_i^{[-1]}\}| < N' - |\mathcal{I}_A \cup \mathcal{I}_B|\} \quad (6)$$

$$\wedge i \notin \mathcal{I}_A \cup \mathcal{I}_B\}. \quad (7)$$

It is notable that  $\mathcal{I}_A$ ,  $\mathcal{I}_B$ , and  $\mathcal{I}_R$  are just the indices of tokens; the tokens for the LLM component are still chosen from the output layer of the vision encoder. After obtaining these token indices, HiPrune directly selects the corresponding tokens from the original token matrix  $\mathbf{T}_v \in \mathbb{R}^{N \times d}$  and discards the rest, leaving a pruned token matrix

$$\mathbf{T}'_v = \mathbf{T}_v[\mathcal{I}_A \cup \mathcal{I}_B \cup \mathcal{I}_R, :] \in \mathbb{R}^{N' \times d}. \quad (8)$$

#### 4.4 Text Guidance

HiPrune is intentionally designed to be text-agnostic, meaning that the pruning process itself does not rely on any textual guidance. This design enables direct application to ViT-based vision models beyond the VLM paradigm, with more robustness and generalization. Nevertheless, HiPrune remains orthogonal to most text-aware token pruning methods and can be seamlessly combined with them. To confirm the compatibility, we introduce a text-aware extension **HiPrune++**, which performs lightweight text-relevance pruning after HiPrune.

Specifically, for each previously unselected visual token  $\mathbf{t}_v^i \in \mathbb{R}^d$ ,  $i \notin \mathcal{I}_A \cup \mathcal{I}_B \cup \mathcal{I}_R$ , we compute its cosine similarity  $\mathbf{r}^i$  with the averaged text embedding  $\mathbf{t}_t \in \mathbb{R}^d$  by

$$\mathbf{r}^i = \frac{\mathbf{t}_v^i \cdot \mathbf{t}_t}{\|\mathbf{t}_v^i\| \cdot \|\mathbf{t}_t\|}. \quad (9)$$

Following the common practice (Zhang et al., 2025e), for vision encoders with a paired text encoder (Radford et al., 2021), we obtain  $\mathbf{t}_t$  with the corresponding encoder. For those without a paired text encoder (Bai et al., 2025), we use the average of all text embeddings. We then retain  $\lceil \beta \cdot N \rceil$  visual tokens by the magnitude of  $\mathbf{r}$ , where  $\beta$  is the proportion of visual tokens selected by text-relevance.

## 5 Experiments

### 5.1 Experiment Settings

Following popular works (Zhang et al., 2025d; Chen et al., 2024a), we conduct evaluations against

other token pruning methods on four widely used VLMs, i.e., LLaVA-1.5-7B (Liu et al., 2024a), LLaVA-NeXT-7B (Liu et al., 2024b), Qwen2.5-VL-3B (Bai et al., 2025), and Video-LLaVA (Lin et al., 2023). Model descriptions, benchmark datasets, and comparison details are in Appendix C.

**Implementation Details.** We follow the default settings for each compared method as specified in their code repositories. In HiPrune, for both LLaVA-1.5-7B and LLaVA-NeXT-7B, we set  $l = 9$  and  $\alpha = 0.1$ . For Qwen, we set  $l = 16$  and  $\alpha = 0.1$  since it has more layers in the vision encoder. We set  $\beta = 0.1$  for all the models when evaluating HiPrune++. Most of the evaluations are performed with the LMMs-Eval toolkit (Zhang et al., 2024b), and FLOPs are computed with the calflops package. All the experiments are conducted on one NVIDIA A100-PCIE (40G) unless otherwise specified.

### 5.2 Accuracy Results

**Results on LLaVA-1.5.** The accuracy results are shown in Table 2 with 192, 128, and 64 tokens retained as the common practice. Across all the settings, HiPrune consistently outperforms existing methods, demonstrating superior performance. Specifically, with **1/3** tokens retained, HiPrune and HiPrune++ preserve **99.3%** and **99.9%** of the original model’s average performance, almost matching the vanilla model. We present Pareto analyses on the token budget against hallucination and accuracy performance in Fig. 5. Interestingly, the superiority of HiPrune++ against HiPrune is more significant with fewer tokens, highlighting the necessity of text guidance under a lower budget.

**Results on LLaVA-NeXT.** In Table 3 we present evaluations on LLaVA-NeXT-7B (Liu et al., 2024b), a high-resolution VLM with more visual tokens. When retaining only **2/9** visual tokens, HiPrune++ preserves **99.7%** accuracy, while HiPrune maintains **99.4%**, which are quite close to the original model. With **11.1%** and **5.6%** of tokens retained, HiPrune++ still preserves **98.4%** and **94.4%** performance, respectively, demonstrating robustness on high-resolution models that handle more images and visual tokens.

**Results on Qwen.** To verify the versatility of HiPrune, we further insert it into Qwen2.5-VL (Bai et al., 2025) in Table 4. Unlike CLIP, some approaches relying on the text encoder or special tokens may have limited performance or cannot

Method	Venue	GQA	MMB	MMB <sup>CN</sup>	MME	POPE	SQA <sup>IMG</sup>	VQA <sup>V2</sup>	VQA <sup>Text</sup>	VizWiz	Average
<i>Vanilla, 576 Tokens (100%)</i>											
LLaVA-1.5-7B	<i>CVPR'24</i>	61.9	64.7	58.1	1862	85.9	69.5	78.5	58.2	50.0	100.0%
<i>Retain 192 Tokens (33.3%)</i>											
ToMe	<i>ICLR'23</i>	54.3	60.5	-	1563	72.4	65.2	68.0	52.1	-	88.5%
FastV	<i>ECCV'24</i>	52.7	61.2	57.0	1612	64.8	67.3	67.1	52.5	50.8	90.4%
HiRED <sup>†</sup>	<i>AAAI'25</i>	58.8	62.6	54.5	1742	83.0	67.9	75.0	-	51.1	96.4%
TRIM <sup>†</sup>	<i>COLING'25</i>	59.9	64.1	53.6	1765	87.1	67.8	76.2	54.9	50.4	97.1%
PyramidDrop	<i>CVPR'25</i>	57.3	63.3	56.8	1797	82.3	69.2	75.1	56.5	51.1	97.2%
VisionZip	<i>CVPR'25</i>	59.3	63.0	-	1783	85.3	68.9	76.8	57.3	-	97.7%
SparseVLM <sup>†</sup>	<i>ICML'25</i>	59.5	64.1	58.0	1780	85.4	68.8	77.0	57.7	50.6	98.6%
<b>HiPrune</b>	<b>Ours</b>	59.2	62.8	57.0	1814	86.1	68.9	76.7	57.6	54.5	<u>99.3%</u>
<b>HiPrune++</b>	<b>Ours</b>	60.3	63.5	57.5	1818	86.9	68.8	77.2	57.5	54.7	<b>99.9%</b>
<i>Retain 128 Tokens (22.2%)</i>											
ToMe	<i>ICLR'23</i>	52.4	53.3	48.8	1343	62.8	59.6	63.0	49.1	50.2	83.0%
FastV	<i>ECCV'24</i>	49.6	56.1	56.4	1490	59.6	60.2	61.8	50.6	51.3	85.4%
HiRED <sup>†</sup>	<i>AAAI'25</i>	57.1	61.7	53.9	1714	79.8	68.1	73.5	-	51.4	95.0%
TRIM <sup>†</sup>	<i>COLING'25</i>	58.9	63.3	51.5	1732	87.2	68.4	74.8	52.7	50.6	95.7%
PyramidDrop	<i>CVPR'25</i>	57.1	61.6	56.6	1761	82.3	68.4	72.9	56.6	51.0	96.2%
VisionZip	<i>CVPR'25</i>	57.6	62.0	-	1762	83.2	68.9	75.6	56.8	-	96.2%
SparseVLM <sup>†</sup>	<i>ICML'25</i>	53.8	64.4	58.1	1761	85.0	68.5	76.3	56.7	50.2	97.0%
<b>HiPrune</b>	<b>Ours</b>	57.3	62.2	56.4	1782	82.8	68.3	74.9	56.6	54.3	<u>97.5%</u>
<b>HiPrune++</b>	<b>Ours</b>	59.0	62.3	57.0	1780	86.4	68.5	76.2	56.0	54.6	<b>98.8%</b>
<i>Retain 64 Tokens (11.1%)</i>											
ToMe	<i>ICLR'23</i>	48.6	43.7	38.9	1138	52.5	50.0	57.1	45.3	50.2	73.1%
FastV	<i>ECCV'24</i>	46.1	48.0	52.7	1256	48.0	51.1	55.0	47.8	50.8	76.7%
HiRED <sup>†</sup>	<i>AAAI'25</i>	54.6	60.2	51.3	1595	73.7	68.2	69.8	-	53.3	91.8%
TRIM <sup>†</sup>	<i>COLING'25</i>	56.9	61.5	44.9	1603	86.7	69.0	71.9	50.0	50.6	92.1%
PyramidDrop	<i>CVPR'25</i>	47.5	58.8	50.5	1561	55.9	69.0	69.2	50.6	50.7	86.6%
VisionZip	<i>CVPR'25</i>	55.1	60.1	-	1690	77.0	69.0	72.4	55.5	-	<u>92.7%</u>
SparseVLM <sup>†</sup>	<i>ICML'25</i>	53.7	60.1	52.5	1559	77.5	69.7	70.2	53.4	50.4	91.8%
<b>HiPrune</b>	<b>Ours</b>	53.6	59.5	53.4	1646	73.0	68.9	69.2	54.9	54.4	<u>92.7%</u>
<b>HiPrune++</b>	<b>Ours</b>	56.4	60.3	53.8	1767	84.3	68.9	72.8	54.5	54.7	<b>96.1%</b>

Table 2: **Results on LLaVA-1.5-7B.** ‘†’ denotes our reproduced results, others are from (Zhang et al., 2025a).

be implemented. When applied to Qwen, HiPrune achieves SOTA performance across the three settings. At an **11.1%** retention rate, HiPrune preserves **93.0%** of the model’s original performance, outperforming FastV and VisionZip by **6.6%** and **1.5%**, respectively. The results on the Qwen series further support our key insights on vision encoders, regardless of their pre-training data or architecture.

**Results on Video-LLaVA.** Video-LLaVA results are in Appendix D.1 due to the space limitation.

### 5.3 Efficiency Results

**Latency and Throughput.** We analyze the decoding latency and throughput of HiPrune on LLaVA with a simple example of around 600 tokens in Table 5. Compared with SparseVLM using 192 visual tokens, HiPrune achieves **1.32×** faster during LLM prefill and gains a **1.20×** speedup in generation, demonstrating its superior efficiency. When reducing the visual token number to 64, the FLOPs of a single forward pass are reduced by **78.3%**, resulting in a **2.47×** faster prefill.

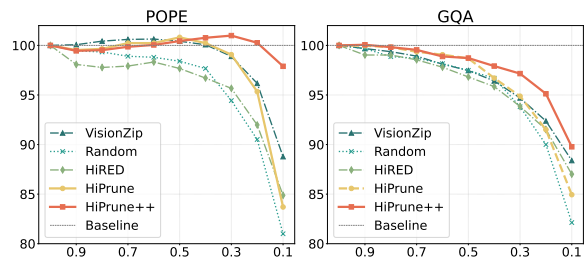


Figure 5: **Pareto analyses.** The horizontal axis is the token retention ratio, while the vertical axis is the percentage normalized accuracy results on LLaVA-1.5-7B.

**Overhead Analyses.** In Fig. 6 we plot the time consumption of each component in HiPrune and HiPrune++ during the prefill phase. Both methods operate before the LLM decoder and thus are compatible with FlashAttention(Dao et al., 2022). The sort introduced by HiPrune consumes only 1% of the total time, which is negligible compared to the largely reduced prefill latency. With a text encoder, HiPrune++ still maintains a total overhead of under 10%, which is acceptable in deployment.

Method	MMB	MMB <sup>CN</sup>	POPE	SQA <sup>IMG</sup>	VizWiz	Avg
<i>Vanilla, 2880 Tokens (100%)</i>						
LLaVA	67.4	60.6	86.5	70.1	57.6	100%
<i>Retain 640 Tokens (22.2%)</i>						
HiRED	66.0	57.0	85.0	68.3	59.1	98.1%
TRIM	66.8	55.8	86.9	66.9	54.8	96.0%
VisionZip	66.3	58.1	86.3	68.1	57.1	98.1%
DivPrune	65.0	56.4	85.4	67.9	58.6	97.4%
VisPruner	65.2	56.0	85.7	67.8	60.9	98.1%
<b>HiPrune</b>	67.0	59.3	85.3	68.0	59.9	<b>99.4%</b>
<b>HiPrune++</b>	67.2	59.1	87.1	67.8	59.9	<b>99.7%</b>
<i>Retain 320 Tokens (11.1%)</i>						
HiRED	64.2	56.4	83.3	66.8	58.3	96.2%
TRIM	63.5	51.0	86.5	66.2	53.5	93.1%
VisionZip	63.1	55.6	82.1	67.3	56.2	94.8%
DivPrune	63.9	55.2	83.0	67.7	57.4	95.6%
VisPruner	63.8	55.4	80.8	68.3	60.4	<u>96.4%</u>
<b>HiPrune</b>	65.3	57.0	78.9	67.3	59.9	<u>96.4%</u>
<b>HiPrune++</b>	66.2	57.4	85.6	67.2	60.1	<b>98.4%</b>
<i>Retain 160 Tokens (5.6%)</i>						
TRIM	61.6	45.2	84.8	65.5	52.9	89.9%
VisionZip	60.1	50.4	74.8	68.3	55.5	90.5%
DivPrune	62.5	52.3	78.4	68.3	57.5	<u>93.4%</u>
VisPruner	59.2	51.3	73.5	68.9	60.1	92.0%
<b>HiPrune</b>	59.8	50.7	67.7	68.7	57.2	89.6%
<b>HiPrune++</b>	61.5	50.6	85.0	68.0	58.6	<b>94.4%</b>

Table 3: **Results on LLaVA-NeXT-7B.** The full table (Table 9) for more comparisons is in Appendix D.1.

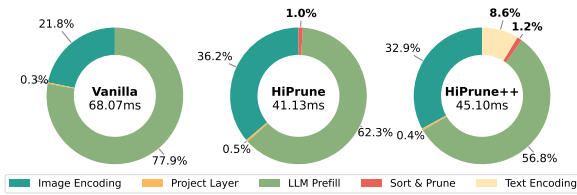


Figure 6: **Component overhead analyses.** The numbers inside each circle denote the wall-clock prefill latency for LLaVA-1.5-7B measured on RTX 5090. The budget for HiPrune and HiPrune++ is 192.

## 5.4 Ablation Studies

We provide ablation studies on key designs in this subsection. Extended studies on hyperparameters and buffer selections are provided in Appendix D.2.

**Attention Pattern.** Many methods prune tokens guided by the CLS token’s attention to other tokens; however, not every model has this token, making these methods model-specific. We compare computing attention score  $\mathbf{a}^{[l]}$  by Eq. 3 and CLS in Table 6(a). For our method, the global attention achieves slightly better results and features much stronger versatility since it is model-agnostic.

**Token Types.** We examine the retained anchor, buffer, and register tokens in HiPrune. As Table 6(b) shows, removing either type degrades the model’s performance. Specifically, removing regis-

Method	MMB	MMB <sup>CN</sup>	POPE	SQA <sup>IMG</sup>	VizWiz	Avg
<i>Vanilla, 100% Tokens</i>						
Qwen	77.3	73.0	87.0	80.4	68.3	100%
<i>Retain 33.3% Tokens</i>						
FastV	74.4	70.6	85.0	79.3	66.9	97.4%
VisionZip	74.9	69.8	85.4	80.1	67.1	97.7%
<b>HiPrune</b>	75.9	71.4	85.9	79.4	68.4	<b>98.7%</b>
<b>HiPrune++</b>	76.0	71.1	85.9	79.9	68.0	<b>98.7%</b>
<i>Retain 22.2% Tokens</i>						
FastV	72.4	69.2	82.7	79.6	66.2	95.9%
VisionZip	73.5	67.4	84.6	80.0	66.3	96.2%
<b>HiPrune</b>	73.7	69.1	84.9	80.2	67.1	<b>97.1%</b>
<b>HiPrune++</b>	74.2	69.4	84.4	80.2	66.7	<b>97.1%</b>
<i>Retain 11.1% Tokens</i>						
FastV	56.2	60.7	73.3	79.3	63.8	86.4%
VisionZip	67.8	63.3	80.2	79.5	62.8	91.5%
<b>HiPrune</b>	69.6	65.4	80.4	79.1	64.6	<b>93.0%</b>
<b>HiPrune++</b>	70.5	64.7	79.9	79.4	64.5	<b>93.0%</b>

Table 4: **Results on Qwen2.5-VL-3B-Instruct.** All the results are reproduced by us.

Method	FLOPs (T)↓	Prefill (ms)↓	Decode (ms)↓	Throughput (tokens/s)↑	VRAM (GB)↓
<i>Vanilla, 576 Tokens (100%)</i>					
LLaVA-7B	8.63	54.31±0.37	21.85±0.26	44.42±0.37	14.52
<i>Retain 192 Tokens (33.3%)</i>					
HiPrune	3.56	28.56±0.30	21.65±0.08	45.65±0.07	14.52
HiPrune++	3.57	28.89±0.11	21.73±0.15	45.23±0.08	15.38
Random	3.56	28.83±0.14	20.97±0.07	47.17±0.05	14.52
<i>Retain 128 Tokens (22.2%)</i>					
HiPrune	2.71	25.59±0.17	21.53±0.08	45.59±0.07	14.35
HiPrune++	2.73	25.76±0.16	21.59±0.10	45.47±0.07	15.38
Random	2.71	25.52±0.14	20.59±0.15	47.59±0.20	14.07
<i>Retain 64 Tokens (11.1%)</i>					
HiPrune	1.87	21.96±0.10	21.33±0.10	45.93±0.07	14.35
HiPrune++	1.88	22.03±0.11	21.41±0.12	45.80±0.10	15.38
Random	1.87	21.81±0.15	21.00±0.39	46.80±0.64	14.02

Table 5: **Wall-clock latency and throughput.** ‘Random’ randomly prunes tokens and adds no computational overhead, serving as a reference.

ter tokens causes the most significant degradation, highlighting the critical role of global information, which is in line with previous studies (Yang et al., 2025b; Zhang et al., 2025d).

**Buffer Selection Scheme.** In Table 6(c) we present different buffer token selection schemes. A square or cross can both achieve similar results; however, when the size of the buffer is too small or missing (row 2 in (b)), the results begin to drop. The buffer tokens are introduced to resist the noise in the attention map. When the window is too small, the effects of the buffer become limited.

**Object Layer Setting.** We adopt a dispersion-based searching strategy to decide the Object Layer  $l$ , where the attention score for object and buffer tokens is extracted. Intuitively, tokens on the same

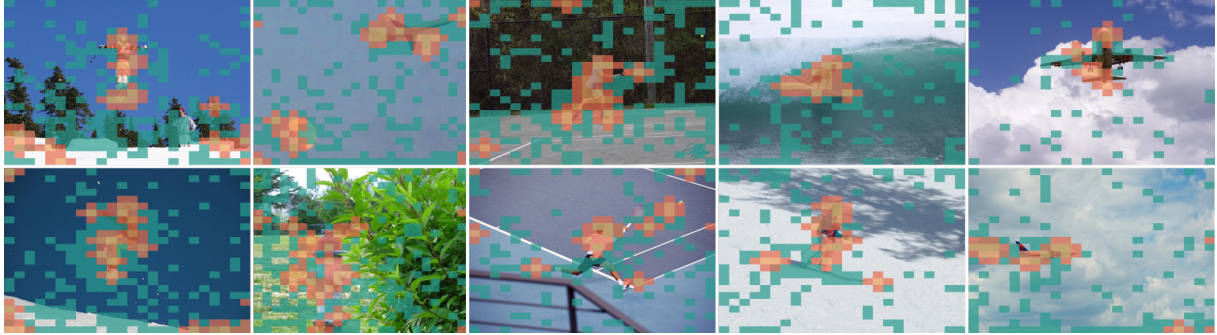


Figure 7: **Visualization on tokens retained by HiPrune.** Anchor tokens are in **yellow**, buffer tokens in **orange**, and register tokens in **teal**. Anchor and buffer tokens focus on the sports player, the aircraft, and the fire extinguisher. The images are slightly resized for better visualization and are randomly chosen from the COCO val2017 dataset.

Setting	GQA	MME	POPE	VizWiz	Avg
<i>(a) Attention Pattern</i>					
CLS Token	59.4	1772	85.4	55.5	99.8%
Global*	59.2	1814	86.1	54.5	<b>100.0%</b>
<i>(b) Token Type</i>					
w/o Register	58.4	1693	85.5	54.7	97.9%
w/o Buffer	59.1	1807	85.9	54.2	99.7%
w/o Buf+Anc	59.1	1805	85.9	54.4	99.7%
Full*	59.2	1814	86.1	54.5	<b>100.0%</b>
<i>(c) Selection Scheme</i>					
Square(8)	59.2	1817	86.0	54.4	<b>100.0%</b>
Row(2)	59.2	1795	85.9	54.3	99.6%
Cross(4)*	59.2	1814	86.1	54.5	<b>100.0%</b>

Table 6: **Ablation study on HiPrune.** Each set is evaluated on LLaVA-1.5-7B with 192 tokens and  $\alpha = 0.1$ . The number in row (c) denotes the buffer token number. ‘\*’ denotes the default setting.

object should feature high similarity. We plot the average pairwise distance of high-attention tokens and model performance with different  $l$  in Fig. 8. For LLaVA, model performance achieves an optimal when setting  $l$  as 9. We assume that at this layer, tokens with similar semantic information are close in the embedding space, and at a critical point from object-centric to global information.

## 5.5 Visualizations

**Selected Tokens.** In Fig. 7 we present a visualization of retained tokens. Anchor tokens and buffer tokens are mainly distributed on the main objects of the image, such as the body of the player, the aircraft in the sky, etc. Preserving these tokens can bring more information about image details and alleviate hallucination. The register tokens diffuse among the whole image uniformly. Despite showing little correlation with image segmentation, they carry indispensable global information. The reason has been discussed in our Motivated

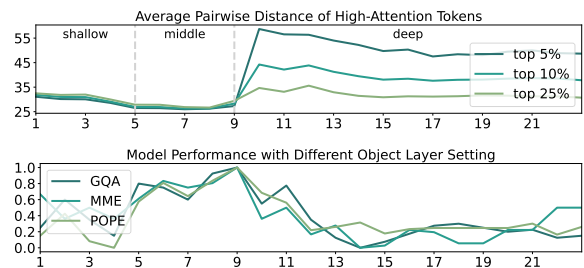


Figure 8: **Average pairwise distance of high-attention tokens and model performance with different Object Layer.** Based on this trend, we divide CLIP into three phases and set the Object Layer at the critical point.

Insights. The combination of these tokens strikes a balance between overall and detail information. More visualizations are in Appendix F.

## 6 Conclusion

In this paper, we investigate the layer-wise attention patterns of vision encoders and reveal that middle layers predominantly capture object-centric features, while deeper layers emphasize global representations. Motivated by this, we propose HiPrune, a model-agnostic and training-free token pruning method that leverages the hierarchical attention structure within the vision encoder. HiPrune mainly preserves anchor, buffer, and register tokens, and an optional set of visual tokens selected by similarity with text embeddings. Extensive experiments across diverse VLMs demonstrate the robustness and generality of HiPrune, which achieves state-of-the-art results while significantly reducing computational overhead. We believe our findings offer valuable perspectives on the internal representation of vision encoders, and HiPrune will facilitate more efficient deployment of VLMs and inspire future research in this direction.

## Acknowledgement

This work is supported in part by the National Natural Science Foundation of China under grants U24A20328, 62301189, 62476071, 62576122, 62536003, 62521006, Guangdong Basic and Applied Basic Research Foundation under grants 2025A1515011732, 2026A1515011139.

## Limitations

HiPrune is most readily applicable to open-source models where architectural modifications are feasible, and cannot be incorporated into commercial models like Gemini and ChatGPT due to their black-box nature. Additionally, HiPrune relies on the multi-head self-attention in the transformer architecture and is not directly transferable to models using a CNN encoder. Besides, we only provide an explicit and simple way to couple HiPrune and text guidance. Despite the outstanding results, future work may focus on taking a better approach to couple text guidance and HiPrune.

## References

- Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. 2025. Divprune: Diversity-based visual token pruning for large multimodal models. In *CVPR*, pages 9392–9401.
- Kazi Hasan Ibn Arif, JinYi Yoon, Dimitrios S Nikolopoulos, Hans Vandierendonck, Deepu John, and Bo Ji. 2025. Hired: Attention-guided token dropping for efficient inference of high-resolution vision-language models. In *AAAI*, pages 1773–1781.
- Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zhohus, and 1 others. 2025. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*.
- Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*, 1(2):3.
- Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.
- Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2023. Token merging: Your vit but faster. In *ICLR*.
- Qingqing Cao, Bhargavi Paranjape, and Hannaneh Hajishirzi. 2023. PuMer: Pruning and merging tokens for efficient vision language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12890–12903, Toronto, Canada. Association for Computational Linguistics.
- Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 2024a. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *ECCV*, pages 19–35. Springer.
- Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, and 1 others. 2024b. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. In *CVPR*.
- Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *NerulPS*, pages 16344–16359.
- Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. 2024. Vision transformers need registers. In *ICLR*.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, and 1 others. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*.
- Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, and Rongrong Ji. 2024. Mme: A comprehensive evaluation benchmark for multimodal large language models. *Preprint*, arXiv:2306.13394.
- Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *CVPR*.
- Zonghao Guo, Ruyi Xu, Yuan Yao, Junbo Cui, Zanlin Ni, Chunjiang Ge, Tat-Seng Chua, Zhiyuan Liu, and Gao Huang. 2024. Llava-uhd: an lmm perceiving any aspect ratio and high-resolution images. In *ECCV*, pages 390–406. Springer.
- Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. 2018. Vizwiz grand challenge: Answering visual questions from blind people. In *CVPR*, pages 3608–3617.
- Anwen Hu, Haiyang Xu, Liang Zhang, Jiabo Ye, Ming Yan, Ji Zhang, Qin Jin, Fei Huang, and Jingren Zhou. 2025. mPLUG-DocOwl2: High-resolution

- compressing for OCR-free multi-page document understanding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5817–5834, Vienna, Austria. Association for Computational Linguistics.
- Xiaohu Huang, Hao Zhou, and Kai Han. 2025. PruneVid: Visual token pruning for efficient video large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19959–19973, Vienna, Austria. Association for Computational Linguistics.
- Drew A Hudson and Christopher D Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *CVPR*, pages 6700–6709.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.
- Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, and 1 others. 2024. Mvbench: A comprehensive multi-modal video understanding benchmark. In *CVPR*, pages 22195–22206.
- Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Evaluating object hallucination in large vision-language models. *arXiv preprint arXiv:2305.10355*.
- Zijing Liang, Yanjie Xu, Yifan Hong, Penghui Shang, Qi Wang, Qiang Fu, and Ke Liu. 2024. A survey of multimodal large language models. In *Proceedings of the 3rd International Conference on Computer, Artificial Intelligence and Control Engineering*, pages 405–409.
- Bin Lin, Yang Ye, Bin Zhu, Jiayi Cui, Munan Ning, Peng Jin, and Li Yuan. 2023. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*.
- Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer.
- Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024a. Improved baselines with visual instruction tuning. In *CVPR*, pages 26296–26306.
- Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. 2024b. Llava-next: Improved reasoning, ocr, and world knowledge.
- Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. In *NeurIPS*, pages 34892–34916.
- Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, and 1 others. 2024c. Mmbench: Is your multi-modal model an all-around player? In *ECCV*, pages 216–233. Springer.
- Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *NeurIPS*.
- Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.
- Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. 2021. Docvqa: A dataset for vqa on document images. In *WACV*, pages 2200–2209.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 32.
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763. PMLR.
- Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. 2019. Towards vqa models that can read. In *CVPR*, pages 8317–8326.
- Dingjie Song, Wenjun Wang, Shunian Chen, Xidong Wang, Michael Guan, and Benyou Wang. 2025. Less is more: A simple yet effective token reduction method for efficient multi-modal llms. In *COLING*.
- Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.
- Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In *ICML*, pages 10347–10357. PMLR.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.
- Tiannan Wang, Wangchunshu Zhou, Yan Zeng, and Xinsong Zhang. 2023. [EfficientVLM: Fast and accurate vision-language models via knowledge distillation and modal-adaptive pruning](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13899–13913, Toronto, Canada. Association for Computational Linguistics.
- Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, and 1 others. 2025. Pyramidrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. In *CVPR*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Jiawei Yang, Katie Z Luo, Jiefeng Li, Congyue Deng, Leonidas Guibas, Dilip Krishnan, Kilian Q Weinberger, Yonglong Tian, and Yue Wang. 2024. Denoising vision transformers. In *ECCV*, pages 453–469. Springer.
- Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. 2025b. Visionzip: Longer is better but not necessary in vision language models. In *CVPR*, pages 19792–19802.
- Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. 2023. Sigmoid loss for language image pre-training. In *ICCV*, pages 11975–11986.
- Ce Zhang, Kaixin Ma, Tianqing Fang, Wenhao Yu, Hongming Zhang, Zhisong Zhang, Yaqi Xie, Kattia Sycara, Haitao Mi, and Dong Yu. 2025a. Vs-can: Rethinking visual token reduction for efficient large vision-language models. *arXiv preprint arXiv:2505.22654*.
- Daoze Zhang, Yuze Zhao, Jintao Huang, and Yingda Chen. 2025b. [Sharper and faster mean better: Towards more efficient vision-language model for hour-scale long video understanding](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4423–4439, Vienna, Austria. Association for Computational Linguistics.
- Jianrui Zhang, Cai Mu, and Yong Jae Lee. 2024a. [Vinoground: Scrutinizing lmms over dense temporal reasoning with short videos](#). *arXiv*.
- Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, and Ziwei Liu. 2024b. [Lmms-eval: Reality check on the evaluation of large multimodal models](#). *Preprint*, arXiv:2407.12772.
- Qizhe Zhang, Aosong Cheng, Ming Lu, Renrui Zhang, Zhiyong Zhuo, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. 2025c. Beyond text-visual attention: Exploiting visual cues for effective token pruning in vlms. In *ICCV*.
- Qizhe Zhang, Aosong Cheng, Ming Lu, Zhiyong Zhuo, Minqi Wang, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. 2025d. [cls] attention is all you need for training-free visual token pruning: Make vlm inference faster. In *ICCV*.
- Qizhe Zhang, Mengzhen Liu, Lichen Li, Ming Lu, Yuan Zhang, Junwen Pan, Qi She, and Shanghang Zhang. 2025e. Beyond attention or similarity: Maximizing conditional diversity for token pruning in mllms. *arXiv preprint arXiv:2506.10967*.
- Shaolei Zhang, Qingkai Fang, Zhe Yang, and Yang Feng. 2025f. LLaVA-mini: Efficient image and video large multimodal models with one vision token. In *ICLR*.
- Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and 1 others. 2025g. Sparsevlm: Visual token sparsification for efficient vision-language model inference. In *ICML*.

## A Pseudo-code for HiPrune

In Algorithm 1, we provide a pseudo-code for HiPrune and HiPrune++ written in PyTorch style (Paszke et al., 2019) to better explain our method. This example is adapted from LLaVA and adopts the cross as the strategy to select buffer tokens.

---

### Algorithm 1 HiPrune and HiPrune++

**Input:** Image tensor image

**Parameter:** Token Budget  $N$ , Object Layer  $l$ , Object Proportion  $\alpha$ , Encoder Patch Size  $p$

**Output:** Pruned token tensor retained\_tokens

```
1 image_tokens, all_attns = encoder(image)
2 ## Compute attention score from object layer l
3 mid_attn = all_attns[l].squeeze(0) # Remove batch
4 mid_attn = mid_attn.mean(0) # Average multi-head
5 mid_attn = mid_attn.sum(0) # Attention to each
  token
6 mid_attn = mid_attn[1:] # Exclude cls
7 ## Assign anchor tokens
8 a_sum = round(N * alpha / 5) # 5 tokens in a
  cluster
9 a_idx = topk(mid_attn, k=a_sum).indices
10 ## Assign buffer tokens
11 b_idx = cat([a_idx-1, a_idx+1, a_idx-p, a_idx+p])
12 a_b_idx = unique(cat([a_idx, b_idx]).clamp(0, p
  **2))
13 ## Compute attention score from output layer
14 deep_attn = all_attns[-1].squeeze(0)
15 deep_attn = mid_attn.mean(0).sum(0)[1:]
16 ## Assign register tokens
17 mask = zeros(N).scatter_(a_b_idx, 1)
18 deep_attn -= mask # Mask already-chosen
19 r_sum = N - a_b_idx.shape[0]
20 r_idx = topk(deep_attn, k=r_sum).indices
21 ## Text Guidance in HiPrune++
22 t_sum = round(N * beta / 5)
23 text_tokens = text_encoder(text)
24 avg_text_tokens = text_tokens.mean(-2)
25 avg_text_tokens /= avg_text_tokens.norm(-1)
26 image_tokens /= image_tokens.norm(-1)
27 similarity = avg_text_tokens @ image_tokens
28 mask = mask.scatter_(r_idx, 1)
29 similarity -= mask # Mask already-chosen
30 t_idx = topk(similarity, k=t_sum)
31 ## Retain these tokens
32 retained_idx = cat([a_idx, b_idx, r_idx, t_idx])
33 retained_tokens = image_tokens[retained_idx]
34 return retained_tokens
```

---

## B Hierarchical Attention Pattern Details

In Fig. 2, we show how attention distribution shifts across layers in CLIP. Here, we explain details about Fig. 2(a) and Fig. 2(b). Since HiPrune prunes visual tokens by their rankings rather than absolute values, we focus on the ranking of each token’s attention. In Alg. 2, we state our acquisition process of Fig. 2(a), which shows the ranking of attentions across layers. In Fig. 2(b), it is worth noting that each dot (regardless of color) is the projected token **from the output layer**, and the color does not mean that the token is drawn from middle layers. We have included a comprehensive visualization in the next section.

## C Evaluation Details

**Models.** HiPrune is model-agnostic and training-free, applicable to any VLM with at least one vision

---

### Algorithm 2 Acquisition process of Fig. 2(a).

**Input:** Image tensor image

**Output:** Attention Ranking Coordinates  $\text{coor\_2d}$

```
1 all_ranks = []
2 tokens, all_attns = encoder(image)
3 for attn in all_attns:
4     ## Same attn extraction process in Alg. 1
5     attn = attn.squeeze(0)
6     attn = attn.mean(0).sum(0)[1:]
7     ## Transform attn values into ranks
8     ## [0.2, 0.3, 0.5, 0.1] -> [1, 2, 3, 0]
9     attn_rank = argsort(argsort(attn)) # [1, 576]
10    all_ranks.append(attn_rank)
11 all_ranks = stack(all_ranks, dim=0) # [24, 576]
12 tsne = TSNE(n_components=2) # Project into 2D
  space
13 coor_2d = tsne.fit_transform(all_ranks) # [24, 2]
14 return coor_2d # plt.scatter(coor_2d)
```

encoder and an LLM. We conduct HiPrune on models with various vision encoders and visual token partition strategies. Following most previous work, we evaluate on LLaVA-1.5-7B (Liu et al., 2024a) and LLaVA-NeXT-7B (Liu et al., 2024b), which encode images into fixed-length token sequences. We also include evaluations on Qwen2.5-VL-7B-Instruct and Qwen2.5-VL-32B-Instruct (Bai et al., 2025), which utilizes a dynamic-resolution ViT and encodes images into sequences of varying lengths. For video evaluations, we apply HiPrune on Video-LLaVA (Lin et al., 2023). All the models utilized in this paper are downloaded from Huggingface.

**Comparisons.** We compare HiPrune with 9 SOTA visual token reduction methods: ToMe (Bolya et al., 2023), FastV (Chen et al., 2024a), SparseVLM (Zhang et al., 2025g), HiRED (Arif et al., 2025), TRIM (Song et al., 2025), VisionZip (Yang et al., 2025b), and PyramidDrop (Xing et al., 2025). Some comparisons on Qwen are missing because the corresponding method either can only be applied to LLaVA or does not open-source code.

Among these, ToMe employed a fusion strategy, while FastV accelerated the inference process by reducing unnecessary tokens. SparseVLM utilized sparsity technology to compress tokens in the language model. HiRED decreased model complexity by selectively retaining important tokens. TRIM optimized processing speed and memory usage by eliminating unnecessary tokens. PyramidDrop applied a pyramid structure to reduce tokens layer by layer, and VisionZip enhanced efficiency by intelligently selecting and compressing tokens.

**Datasets.** We conduct thorough experiments across various multimodal benchmarks, including visual question answering benchmarks such as GQA (Hudson and Manning, 2019), SQA (Lu

Method	Token Num	GQA	MMB	MMB <sup>CN</sup>	MME	POPE	SQA <sup>IMG</sup>	VQA <sup>V2</sup>	VQA <sup>Text</sup>	VizWiz
LLaVA-1.5-13B	576	63.2	67.7	63.5	1818	85.9	72.8	80.0	61.3	53.6
w/ HiPrune	192	59.4	67.1	62.5	1798	85.4	73.7	78.0	59.5	55.6
	128	57.9	66.8	63.1	1730	82.8	74.1	76.1	58.3	54.9
	64	54.2	64.8	59.2	1634	72.4	74.6	70.3	56.7	56.0
w/ HiPrune++	192	60.2	66.5	62.5	1808	86.7	73.2	78.5	59.5	55.4
	128	59.1	67.2	62.8	1745	86.2	73.8	77.4	58.8	55.4
	64	56.9	65.0	58.3	1736	84.4	74.1	73.9	56.7	56.0
LLaVA-NeXT-13B	2880	64.4	68.5	61.2	1901	85.3	73.1	82.3	63.2	59.1
w/ HiPrune	640	62.6	70.2	65.3	1877	84.9	71.6	80.0	61.6	61.1
	320	59.3	68.6	64.9	1800	79.7	72.1	75.7	60.1	59.2
	160	54.4	66.3	61.3	1647	71.0	70.9	68.4	57.1	55.9
w/ HiPrune++	640	63.5	69.9	64.8	1894	86.6	72.2	80.5	61.5	60.7
	320	61.5	68.8	63.4	1823	86.3	71.2	77.6	59.8	59.1
	160	58.2	66.3	60.7	1774	86.2	71.2	72.3	55.9	56.7

Table 7: Performance comparisons on LLaVA-1.5-13B and LLaVA-NeXT-13B (Liu et al., 2024a).

Method	Token Budget	GQA	MMB	MMB <sup>CN</sup>	MME	POPE	SQA <sup>IMG</sup>	VQA <sup>text</sup>	VizWiz
Qwen2.5-VL-3B	100%	59.9	77.3	73.0	2144	87.0	80.4	77.8	68.9
w/ HiPrune	33.3%	57.5	75.9	71.8	2061	86.0	79.8	70.1	68.0
	22.2%	55.6	73.7	69.1	2002	84.5	80.0	62.9	67.0
	11.1%	51.5	69.7	65.1	1881	80.0	79.4	50.9	64.5
w/ HiPrune++	33.3%	57.4	75.6	70.9	2069	85.9	79.7	69.2	68.2
	22.2%	55.7	73.5	68.9	1980	84.7	79.8	61.7	66.9
	11.1%	51.3	69.8	64.9	1844	80.0	79.1	48.8	64.4
Qwen2.5-VL-7B	100%	60.5	83.2	80.1	2331	86.2	87.4	83.1	70.4
w/ HiPrune	33.3%	58.9	82.6	79.5	2297	85.1	87.0	78.5	69.2
	22.2%	57.2	80.2	77.6	2168	84.0	85.7	74.1	68.6
	11.1%	52.5	76.1	73.5	1998	80.2	82.8	62.6	66.4
w/ HiPrune++	33.3%	58.6	82.1	79.2	2310	85.1	86.6	77.9	69.0
	22.2%	57.3	80.3	77.0	2172	83.6	85.2	73.2	68.5
	11.1%	52.8	75.6	73.3	1999	79.5	82.8	60.7	66.7

Table 8: Performance comparisons on Qwen2.5-VL-3B-Instruct and Qwen2.5-VL-7B-Instruct (Bai et al., 2025).

et al., 2022), VQAv2 (Goyal et al., 2017), MME (Fu et al., 2024), and TextVQA (Singh et al., 2019). Additionally, we include POPE (Li et al., 2023) and VizWiz (Gurari et al., 2018) to study the hallucination when visual tokens are pruned. We also include MMB and MMB-CN (Liu et al., 2024c) to study the multilingual ability of VLM since some approaches rely on the CLIP text encoder to work and draw back on non-English tasks. For video tasks, we adopt MVBench (Li et al., 2024) and Vinoground (Zhang et al., 2024a) to evaluate the model’s overall performance in multiple domains.

**Toolkits.** For HiPrune, most of our evaluations are completed with LMMs-Eval toolkit (Zhang et al., 2024b). However, since some benchmarks

either are extremely slow on LMMs-Eval or need Internet for an online evaluation, the MMB, MMB-CN (Liu et al., 2024c), TextVQA (Singh et al., 2019), and VQAv2 (Goyal et al., 2017) results in Table 2 are obtained with the public codebase released by LLaVA (Liu et al., 2023). The rest of the results in Table 2 and all the results in Tables 3 and 4 are obtained with LMMs-Eval.

## D Extended Experiments

### D.1 Accuracy Results

**Video Evaluations.** We apply HiPrune on Video-LLaVA-7B (Lin et al., 2023). As Fig. 10 shows, for Vinoground, the accuracy results remain stable and keep **99.2%** performance with **1/16** visual tokens.

Method	MMB	MMB <sup>CN</sup>	POPE	SQA <sup>IMG</sup>	VizWiz	Avg
Vanilla, 2880 Tokens (100%)						
LLaVA	67.4	60.6	86.5	70.1	57.6	100%
Retain 640 Tokens (22.2%)						
FastV	63.1	53.5	79.5	67.4	53.9	92.7%
HiRED	66.0	57.0	85.0	68.3	59.1	98.1%
TRIM	66.8	55.8	86.9	66.9	54.8	96.0%
VisionZip	66.3	58.1	86.3	68.1	57.1	98.1%
DivPrune	65.0	56.4	85.4	67.9	58.6	97.4%
PDrop	64.1	55.2	83.8	66.7	53.8	94.3%
VisPruner	65.2	56.0	85.7	67.8	60.9	98.1%
SparseVLM	65.9	58.6	85.3	67.6	53.6	96.5%
<b>HiPrune</b>	67.0	59.3	85.3	68.0	59.9	99.4%
<b>HiPrune++</b>	67.2	59.1	87.1	67.8	59.9	<b>99.7%</b>
Retain 320 Tokens (11.1%)						
FastV	53.4	42.5	49.5	66.6	51.3	78.1%
HiRED	64.2	56.4	83.3	66.8	58.3	96.2%
TRIM	63.5	51.0	86.5	66.2	53.5	93.1%
VisionZip	63.1	55.6	82.1	67.3	56.2	94.8%
DivPrune	63.9	55.2	83.0	67.7	57.4	95.6%
PDrop	55.5	44.7	60.8	66.7	49.7	81.6%
VisPruner	63.8	55.4	80.8	68.3	60.4	96.4%
SparseVLM	63.1	56.7	76.9	67.2	54.2	93.2%
<b>HiPrune</b>	65.3	57.0	78.9	67.3	59.9	96.4%
<b>HiPrune++</b>	66.2	57.4	85.6	67.2	60.1	<b>98.4%</b>
Retain 160 Tokens (5.6%)						
TRIM	61.6	45.2	84.8	65.5	52.9	89.9%
VisionZip	60.1	50.4	74.8	68.3	55.5	90.5%
DivPrune	62.5	52.3	78.4	68.3	57.5	93.4%
VisPruner	59.2	51.3	73.5	68.9	60.1	92.0%
<b>HiPrune</b>	59.8	50.7	67.7	68.7	57.2	89.6%
<b>HiPrune++</b>	61.5	50.6	85.0	68.0	58.6	<b>94.4%</b>

Table 9: **Results on LLaVA-NeXT-7B.** Comparison results are reported from (Zhang et al., 2025e).

For MVBench, the performance drop is slightly observable but still acceptable.

**Different Model Sizes.** We further apply HiPrune to LLaVA-1.5-13B, LLaVA-NeXT-13B, Qwen2.5-VL-7B-Instruct, and Qwen2.5-VL-32B-Instruct (Liu et al., 2023, 2024a; Bai et al., 2025). These experiments follow the same experiment settings described in Section Experiment. The results for LLaVA-series are reported in Table 7 and Table 9, while the results for Qwen-series are in Table 8 and 10. When implemented on a model with a different size, HiPrune maintains its overall performance and shows a trend similar to a smaller one reported in our paper. Notably, for SQA and VizWiz, HiPrune acquires results even slightly better than baseline under some settings. HiPrune is model-agnostic and easy to deploy on other VLMs, which is among our future works.

**Text-Dominant Results** To evaluate the accuracy performance under text-dominant tasks, we test our method with DocVQA (Mathew et al., 2021) and present the results in Fig. 9. HiPrune and HiPrune++ surpass most comparisons under most

Method	MMB	MMB <sup>CN</sup>	POPE	SQA <sup>IMG</sup>	VizWiz	Avg
Vanilla, 100% Tokens						
Qwen	86.3	84.0	84.2	91.5	64.8	100%
Retain 22.2% Tokens						
FastV	76.4	72.9	64.9	84.4	63.5	88.6%
VisionZip	82.2	79.1	79.2	86.5	64.7	95.6%
<b>HiPrune</b>	82.6	80.0	80.8	87.0	64.2	<b>96.2%</b>
<b>HiPrune++</b>	82.6	79.0	80.1	86.5	64.2	95.7%
Retain 11.1% Tokens						
FastV	67.2	63.5	41.0	82.9	61.9	77.7%
VisionZip	76.0	73.9	71.7	83.5	62.6	89.8%
<b>HiPrune</b>	77.2	74.3	71.7	82.6	63.0	<b>90.1%</b>
<b>HiPrune++</b>	76.6	73.5	72.1	82.9	62.5	89.8%
Retain 5.6% Tokens						
FastV	56.4	53.0	34.3	78.7	60.1	69.6%
VisionZip	69.8	66.6	62.5	81.5	60.9	83.5%
<b>HiPrune</b>	71.3	67.4	63.8	82.6	61.1	<b>84.7%</b>
<b>HiPrune++</b>	70.3	67.4	63.3	81.8	60.8	84.0%

Table 10: **Results on Qwen2.5-VL-32B-Instruct.** All the results are reproduced by us.

Setting	GQA	MME	POPE	VizWiz	Avg
Square(8)	59.2	1817	86.0	54.4	<b>100.0%</b>
Cross(4)*	59.2	1814	86.1	54.5	<b>100.0%</b>
Rot-Cross(4)	59.3	1819	85.7	54.4	<b>100.0%</b>
Row(2)	59.2	1795	85.9	54.3	99.6%
Column(2)	59.1	1805	85.7	54.3	99.7%

Table 11: **Study on selection schemes of buffer tokens.** Each set is evaluated on LLaVA-1.5-7B with 192 tokens and  $\alpha = 0.1$ . The number in denotes buffer token number. ‘\*’ denotes the default setting.

token budgets, demonstrating robustness under text-dominant scenarios.

## D.2 Ablation Studies

**Hyperparameter Sensitivity.** HiPrune depends on two hyperparameters: the object layer  $l$  and the object proportion  $\alpha$ . The former is set by the average pairwise distance of the top-attention token set, while the latter is set manually. As Fig. 11 shows, there does not exist an optimal point regarding  $\alpha$ , and the difference between various  $\alpha$  is trivial, so we simply set it to 0.1 in all our settings. However, for different  $l$ , the optimal results emerge at layer 9, which is exactly the changing point of average pairwise distance in Fig. 8. We hypothesize that this point features the most concentrated information and overlaps well with objects.

**Buffer Selection Scheme.** Buffer tokens are tokens neighbouring anchor tokens. To better depict the positional relation between buffer tokens and anchor tokens, we provide a simple diagram in Fig. 12. Results for these schemes are reported in Ta-

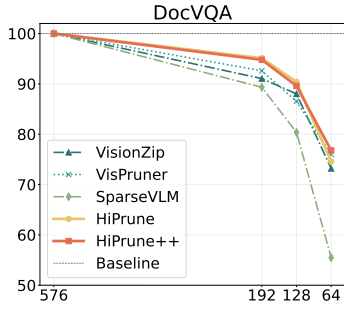


Figure 9: **Results on Text-Dominant Tasks.** We evaluate HiPrune on LLaVA-1.5-7B with the DocVQA dataset. The horizontal axis is the token budget, while the vertical axis is the percentage normalized results.

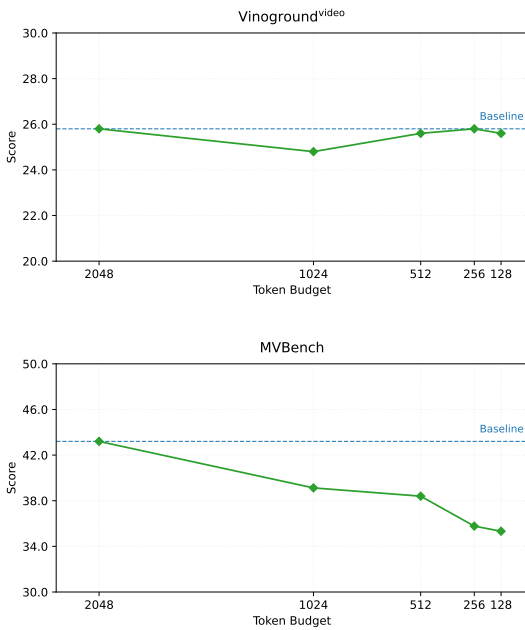


Figure 10: **Results on Video-LLaVA-7B.** We apply HiPrune and set different token budgets. The vanilla model features a token budget of 2048.

ble 11. When the number of buffers around one anchor surpasses 4, the performance stays stable. The buffers are introduced to mitigate misselection caused by noise in the attention map; therefore, theoretically, as long as their coverage size is sufficiently wide, the exact shape does not make a significant difference.

## E Visualization of Attention Evolution

As shown in Fig. 14, the high-attention tokens in the input layer and the output layer (last but one in LLaVA) obey distinct distributions. In these examples, high-attention tokens in the shallow layer distribute uniformly across the embedding space,

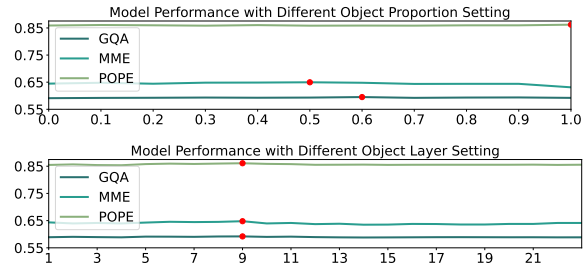


Figure 11: **Hyperparameter sensitivity.** The results are obtained on LLaVA-v1.5-7B with budget  $N' = 192$ . The best performance setting is marked in red.

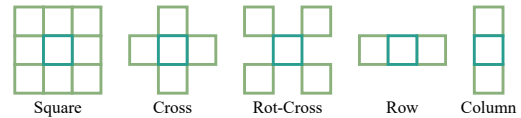


Figure 12: Positional relation between buffer and anchor tokens under different selection schemes. The anchor tokens are in teal while the buffer tokens are in lime.

while they cluster in the output layer. During the shift, the middle layers show a transitional status covering every cluster. These examples indicate that CLIP encodes images in a continuous and gradual way, forming a hierarchical representation inside the vision encoder.

## F Visualizations on Retained Tokens

We provide extended examples on retained tokens in Fig. 13. We can see that the anchor tokens and buffer tokens contain the main body of the image, e.g., the player, the person, and the animal, which are crucial for image understanding. The register tokens seem less image-aligned, but cover the whole image sufficiently, indicating they encode global information. The reason is discussed in the section Experiment, and this visualization further confirms our theory.

## G Visualizations on All Layers' Attention

We provide two examples of all layers' attention distribution in Fig. 15. In the middle layers, the highlighted areas mainly overlap the person, which is the main object of the image, and agrees with the human perception order, as we also first notice the person in the image.



Figure 13: Visualization on tokens retained by HiPrune. The images are randomly chosen from the COCO val2017 set (Lin et al., 2014). Anchor tokens are in yellow, buffer tokens are in red, and register tokens are in cyan.

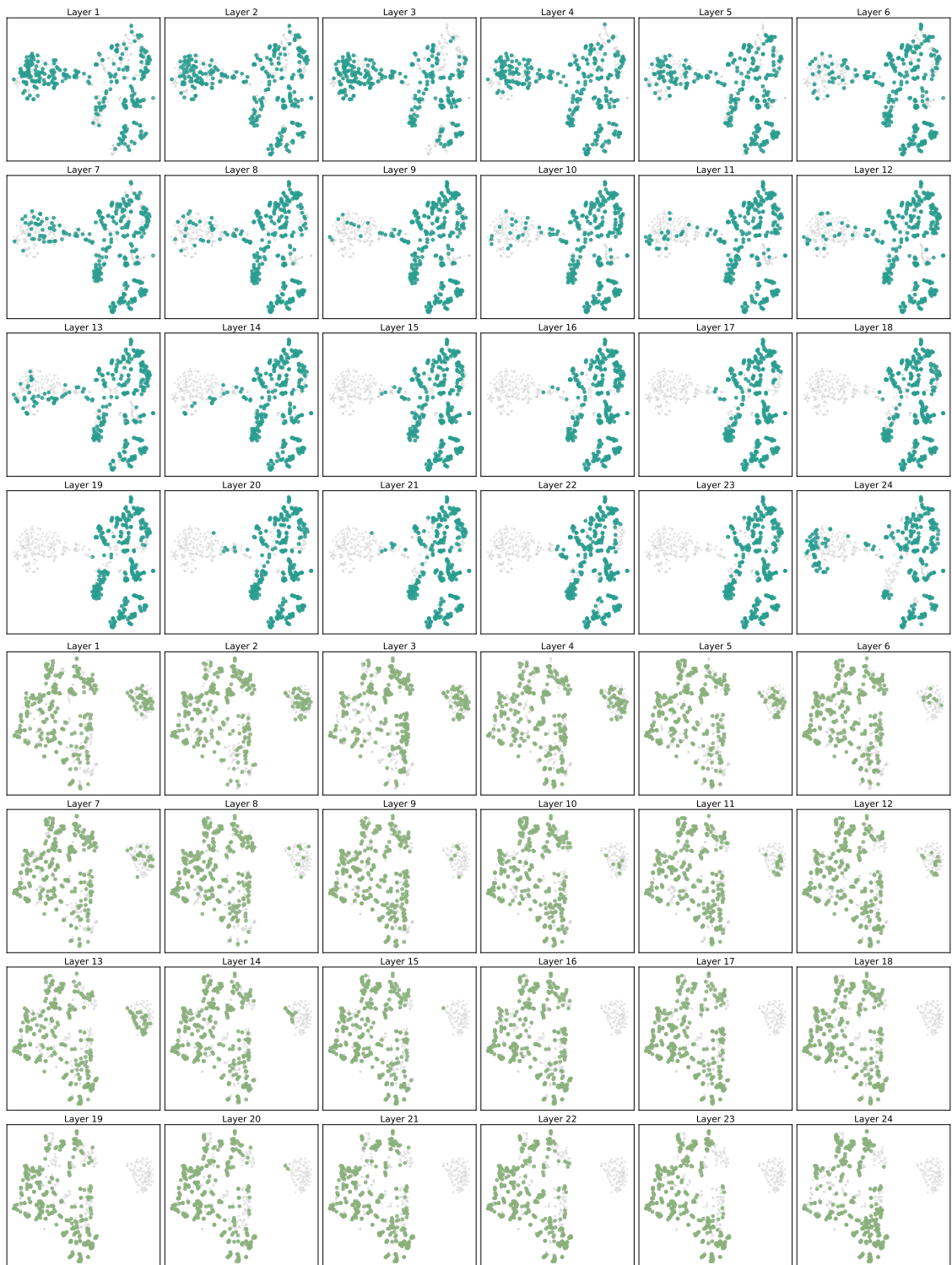


Figure 14: Two examples of t-SNE visualization on visual tokens that receive top 50% attention from different CLIP layers.

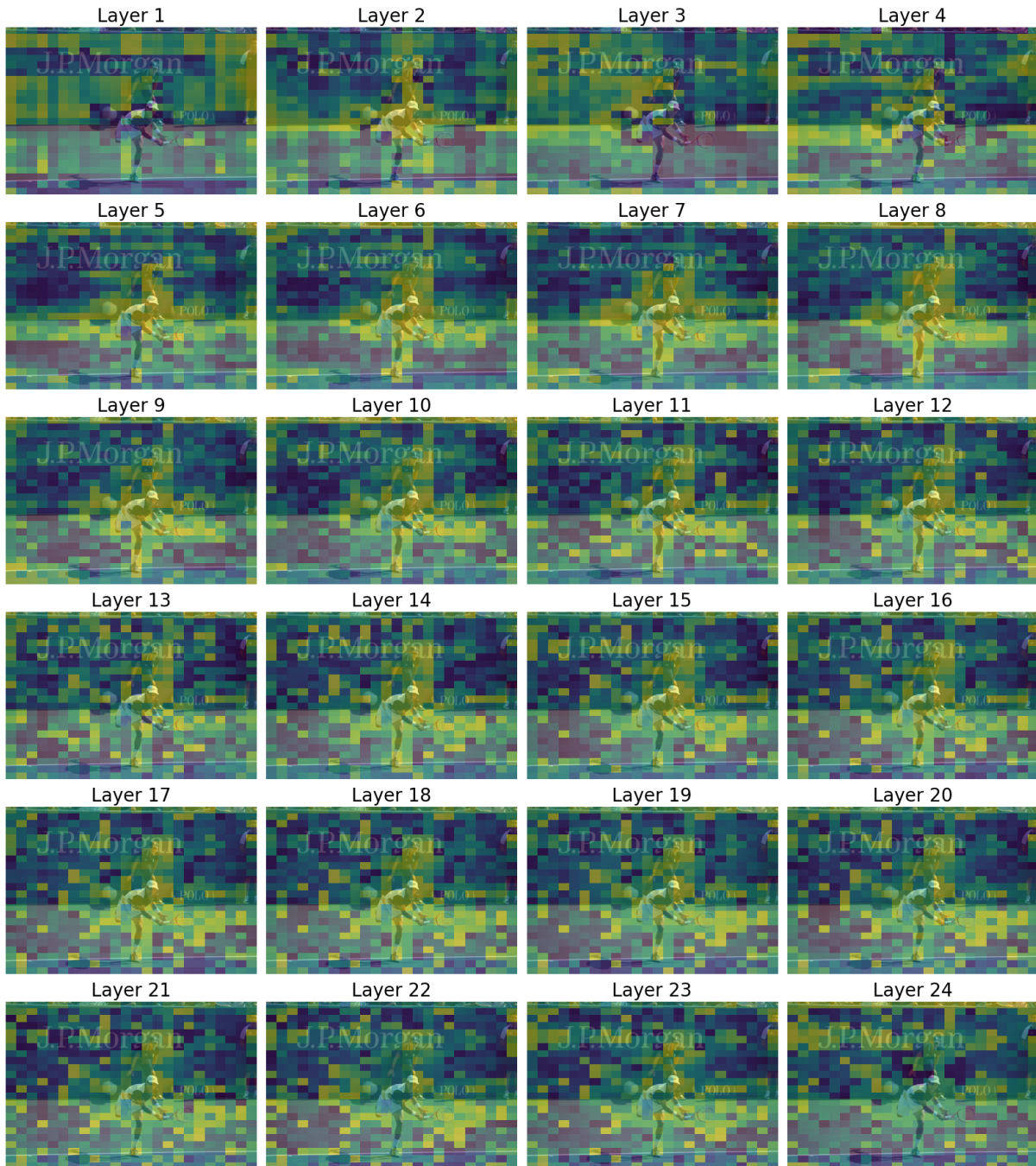


Figure 15: Visualization of all layers' attention in the CLIP model.