Object-Centric Vision Token Pruning for Vision Language Models

Guangyuan Li[1*] Rongzhen Zhao[1†] Jinhong Deng[2] Yanbo Wang[3] Joni Pajarinen[1]
[1]Aalto University
[2]University of Electronic Science and Technology of China
[3]Delft University of Technology
{guangyuan.li, rongzhen.zhao, joni.pajarinen}@aalto.fi
jhdengvision@gmail.com y.wang-27@tudelft.nl

Abstract

In Vision Language Models (VLMs), vision tokens are quantity-heavy yet information-dispersed compared with language tokens, thus consume too much unnecessary computation. Pruning redundant vision tokens for high VLM inference efficiency has been continuously studied but all existing methods resort to indirect and non-guaranteed ways. We propose OC-VTP, a direct and guaranteed approach to select the most representative vision tokens for high-efficiency yet accuracy-preserving VLM inference. Our OC-VTP requires merely light-weight pre-training of a small object-centric vision token pruner, which can then be inserted into existing VLMs, without fine-tuning of any models on any datasets. It is gauranteed that the most representative vision tokens are kept by minimizing the error in reconstructing the original unpruned tokens from the selected ones. Across any vision pruning ratios, i.e., inference efficiency, our OC-VTP consistently helps mainstream VLMs to preserve the highest inference accuracy. Our pruning also demonstrates interesting interpretability. Our codes are available at https://github.com/GarryLarry010131/OC-VTP.

1. Introduction

Recent Vision-Language Models (VLMs) have achieved strong performance on challenging multimodal tasks, particularly open-ended visual question understanding and answering [1, 13, 15, 16, 39]. In typical vision-language inference, textual prompts are tokenized into dozens of language tokens, whereas an image is represented by hundreds or even thousands of vision tokens. Unlike language tokens, many vision tokens carry highly redundant information [17], and often a small subset is sufficient to represent the whole. This motivates Vision Token Pruning (VTP) to reduce VLM computation burden and accelerate inference, while incurring little accuracy loss.

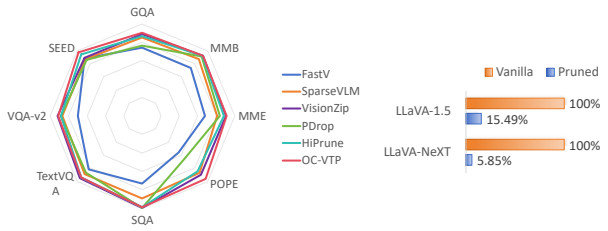


Figure 1. (*left*) Our OC-VTP consistently outperforms prior SotA methods, retaining over **95%** of accuracy with only **11.1%** of visual tokens on LLaVA-1.5. (*right*) Our OC-VTP reduces FLOPs by nearly **85%** on LLaVA-1.5-7B when retaining **11.1%** vision tokens, and by **95%** on LLaVA-NeXT-7B when retaining **5.6%** vision tokens, at a text length of 32, assuming MAC=2.

Given a computation budget, i.e., the number of tokens to be kept, **how to determine** which vision tokens are redundant? ToMe [3] adopts the human intuition that *similar* vision tokens are redundant. FastV [4] defines the importance as the average *attention* score a vision token receives from all other vision tokens. TRIM [25] uses the *similarity* between vision tokens and the pooling of text tokens as the significance. VisionZip [30] also defines the dominance as the average *attention* score or the attention score with the `CLS` token. SparseVLM [34] measure the significance as the *attention* between vision tokens and relevant text tokens. PyramidDrop [29] defines the importance as the *attention* between vision tokens and the last text token. HiPrune [17] uses the overall *attention* scores of different vision layers to measure the importance of vision tokens.

However, all these methods rely on **indirect criteria**. Namely, various attention scores and similarities are meticulously designed as metrics, with the authors' ingenious *intuition*, even based on their insightful *observation*. These are somehow effective, but without any guarantee that the

*Equal contribution.
†Equal contribution; Corresponding author.

most representative vision tokens are selects from all.

We formulate VTP **as an optimal problem** given the budget. The *ideal case* is that we prune the vision tokens that have the least effect on the vision-language inference accuracy. But this requires access to the text tokens, not easy to formalize for optimality guarantee analysis. We focus on the *practical case* that how to prune the least representative vision tokens that minimize the information loss from the original unpruned ones. This is straightforward to formalize for optimality guarantee analysis.

We propose Object-Centric Vision Token Pruning (OC-VTP). It is the **first guaranteed method** that prunes the *least representative* vision tokens, for hight-efficiency yet accuracy-preserving VLM inference. This is realized by pre-training a lightweight Object-Centric pruner (OC-pruner) on the vision tokens of a small subset of images to keep the most representative vision tokens that lose the least information. Then it can plug and play in various VLM inference. As shown in Figure 1, our OC-VTP outperforms state-of-the-art (SotA) VTP methods on various datasets in average accuracy given any budget. Note that all our superiority is achieved without accessing the text tokens for aiding the vision token pruning.

In summary, our contributions are: (*i*) We realize guaranteed VTP for VLMs for the first time that can select the most representative tokens; (*ii*) Our method requires lightweight pretraining of a lightweight module, which can be plugged into various VLMs without fine-tuning; (*iii*) Our method achieves roughly new SotA in terms of VTP for VLM, with intuitive object-level interpretability.

2. Related Work

2.1. Vision Language Models

Mainstream VLMs consist of a vision encoder, a text tokenizer, a multimodal token projector, and a Large Language model (LLM) that infers upon such tokens. LLaVA-1.5 [15] uses the CLIP [22] vision encoder for image tokenization, which produces 576 vision tokens. LLaVA-NeXT [16] adopts the same architecture, but targets high-resolution and multi-image inputs. Its AnyRes technique splits a high-resolution image into several sub-images, all are tokenized and then concatenated, producing 2880 (5×576) tokens in total. Qwen2.5-VL [1, 26, 27] trains a native dynamic-resolution Vision Transformer (ViT) that keeps images at their original size while controlling cost.

These VLMs achieve strong results on benchmarks such as GQA [8], MMBench [18], POPE [12], TextVQA [24] and ScienceQA [20]. However, the inference cost is dominated by the number of vision tokens. High-resolution tiling and multi-image inputs further enlarge the inference computation and latency. This motivates pruning vision tokens, as text tokens are more information intensive, to reduce computation burden while preserving accuracy.

2.2. Vision Token Pruning

FastV [4] uses the attention average of ViT layers to identify important token progressively, while merging the remaining into the important ones. Its VTP happens inside the vision encoder. TRIM [25] uses the similarity between vision tokens and text tokens' pooling to select tokens, while merging the remaining. Its VTP happens after the vision encoder and before the LLM decoder. VisionZip [30] uses the average attention or the attention with the `CLS` token to choose tokens, while group merging the remaining. Its VTP applies after the vision encoder and before the LLM decoder, requiring finetuning on the VLMs. SparseVLM [34] relies on the attention between vision tokens and relevant text tokens to choose important tokens, while recycling the pruned tokens by group merging. Its VTP works inside the LLM decoder layers progressively. PyramidDrop [29] keep tokens according to the attention between vision tokens and the last text token, while removing the remaining. Its VTP happens every several layers inside the LLM decoder progressively. HiPrune [17] utilizes the overall attention of different vision layers for pruning, while removing the remaining, after the vision encoder and before the LLM decoder.

Instead of relying on these handcrafted importance signal, our method learns a general optimality-guaranteed vision token pruner. Its VTP happens between the vision encoder and LLM decoder, no finetuning on VLMs needed.

2.3. Object-Centric Learning

Object-Centric Learning (OCL) [19] represents a scene as a few object-level feature vectors, i.e., *slots*. OCL models mostly adopt the encoding-aggregation-decoding architecture and are trained in self-supervision that minimizes the error in reconstructing the original input from these a few slots. This aligns with the VTP objective.

For aggregation, Slot Attention [19] is the core module of mainstream OCL, either on images like DINOSAUR [23], SlotDiffusion [28], SPOT [10] and VVO [36], or on videos like VideoSAUR [32], SlotContrast [21] and RandSF.Q [35]. Slot Attention and its variants, like BO-QSA [9], which predefines the number of slots, ISA [2], which learns geometric invariance, and SmoothSA [37], which introduces a preheater, all ensure *exclusiveness* and *completeness* on the slots, i.e., least information redundancy or loss. We choose the original version to build our OC-pruner, as it supports variant number of slots, does not change the feature distribution and is highly efficient.

For decoding, there are mixture decoders, like broadcast CNN [19] and MLP [23], auto-regressive deocders, like the Transformer [10] and random auto-regressive Transformer [38], and denoising decoders, like the conditional Diffusion [28]. We choose the random auto-regressive Transformer

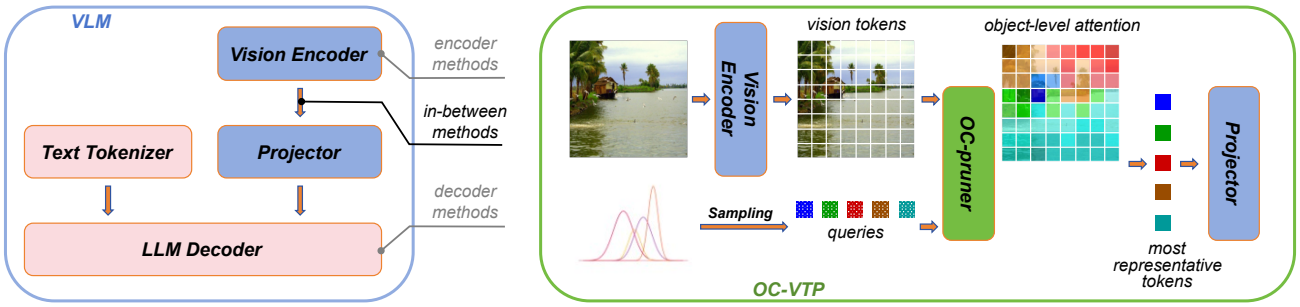


Figure 2. (*left*) Structure of typical Vision-Language Models (VLMs), and three Vision Token Pruning (VTP) places: *encoder* methods like ToMe and FastV [3, 4]; *decoder* methods like SparseVLM and PyramidDrop [29, 34]; and *in-between* methods like TRIM, VisionZip and HiPrune [17, 25, 30]. Our OC-VTP operates in-between. (*right*) OC-pruner is the core of our OC-VTP. It takes the middle layer tokens from the vision encoder as input, and employs queries sampled from a learned Gaussian distribution to locate the most representative token of each object or object part. The corresponding indexes are used to prune the vision tokens.

decoder to build our OC-pruner, as it is adopted by OCL SotA and also is the most lightweight solution.

3. Proposed Method

3.1. Problem Formulation

A vision-language inference task via a VLM has the form:

$$\phi_{\mathrm{VLM}} : (\boldsymbol{V}, \boldsymbol{L}) \rightarrow \boldsymbol{O} \tag{1}$$

Here $\phi_{\mathbf{VLM}}$ is the inference model. $\boldsymbol{V}$ and $\boldsymbol{L}$ are the vision and language parts of the inference task sample, respectively, with $\boldsymbol{L}$ mainly defining the task while $\boldsymbol{V}$ mainly providing the required information. $\boldsymbol{O}$ is the inference output and let us denote $\boldsymbol{Y}$ as the ground-truth answer.

Basically, the **ideal objective** of VTP for VLMs is to remove the vision tokens that have least impact on the inference task given the budget. This requires us to develop an pruner that can prune vision tokens that cause the least average accuracy drop in all inference tasks:

$$\min \mathbb{E}_{(\boldsymbol{V},\boldsymbol{L})}\big[\mathbb{1}[\boldsymbol{O}=\boldsymbol{Y}]\big] - \mathbb{E}_{(\boldsymbol{V}_{\mathrm{p}}^{*},\boldsymbol{L})}\big[\mathbb{1}[\boldsymbol{O}_{\mathrm{p}}^{*}=\boldsymbol{Y}]\big] \tag{2}$$

$$\text{where} \quad \phi_{\mathrm{p}}^{*} : (\boldsymbol{V}; \boldsymbol{L}, \phi_{\mathrm{VLM}}) \rightarrow \boldsymbol{V}_{\mathrm{p}}^{*} \tag{3}$$

Here ϕ_{p}^{*} is the ideal pruner, keeping the most *important* vision tokens $\boldsymbol{V}_{\mathrm{p}}^{*}$, according to the vision input $\boldsymbol{V}$ and language input $\boldsymbol{L}$, as well as the inference model ϕ_{VLM}.

However, finding a solution for the ideal pruner ϕ_{p}^{*} submit to the ideal objective Equation (2) is hardly possible. *This is also why all existing VTP methods rely on handcrafted vision token importance metrics*.

But we can simplify this optimal problem into a **practical objective**. Assuming no access to the language input and the VLM, we need to develop a practical pruner that can keep vision tokens that preserve the most information of the original unpruned vision tokens:

$$\min \phi_{\mathrm{d}}(\boldsymbol{V}, \boldsymbol{V}_{\mathrm{p}}) \tag{4}$$

$$\text{where} \quad \phi_{\mathrm{p}} : \boldsymbol{V} \rightarrow \boldsymbol{V}_{\mathrm{p}} \tag{5}$$

Here ϕ_{p} is the practical pruner, which identifies the most *representative* vision tokens $\boldsymbol{V}_{\mathrm{p}}$ from the original unpruned ones $\boldsymbol{V}$. ϕ_{d} is some distance metric, which measures the information gap between two token sets.

In this case, the practical pruner ϕ_{p} is agnostic to both language input $\boldsymbol{L}$ and the inference model ϕ_{VLM}. *Finding a solution for it is quite feasible* as we only need to deal with the vision input $\boldsymbol{V}$ itself.

3.2. Building OC-Pruner

Based on Equation (5), we **design** our OC-pruner as follows. Given the budget, i.e., the vision sequence length after pruning, we draw the corresponding number of queries to aggregate the vision tokens into slots through a Slot Attention module, where vision tokens belonging to the same object or object part are aggregated into one slot.

$$\boldsymbol{S}, \boldsymbol{A} = \phi_{\mathrm{SA}}(\boldsymbol{Q}, \boldsymbol{V}) \tag{6}$$

Here $\boldsymbol{Q} \in \mathbb{R}^{s\times c}$ is the queries sampled from a learnt Gaussian distribution, with s as the budget. $\boldsymbol{V} \in \mathbb{R}^{n\times c}$ is the original vision tokens to be pruned. ϕ_{SA} is the trained Slot Attention [19] module, and please refer to Section 2.3 for why not choosing other variants. $\boldsymbol{S} \in \mathbb{R}^{s\times c}$ is the aggragated slots, with $\boldsymbol{A} \in \mathbb{R}^{s\times n}$ as the corresponding attention maps. All these slots together can represent the original unpruned vision tokens with least information loss.

By the attention map of each slot, we can locate the most attended vision token, which is also the most representative token, then we can conduct the pruning:

$$\boldsymbol{I} = \mathrm{argmax}_{n}(\boldsymbol{A}) \tag{7}$$

$$\boldsymbol{V}_{\mathrm{p}} = \boldsymbol{V}_{[\boldsymbol{I},:]} \tag{8}$$

Here the most attended indexes $\boldsymbol{I} \in \mathbb{R}^{s}$ are obtained by argmax $\boldsymbol{A}$ along the dimension n. Besides argmax,

we also tried top-k, which as shown in Table 5 is inferior. The pruned vision tokens $\boldsymbol{V}_{\mathrm{p}} \in \mathbb{R}^{s \times c}$ are obtained by selecting tokens from the original unpruned ones $\boldsymbol{V}$ with indexes $\boldsymbol{I}$. Briefly, we implement the practical pruner ϕ_{p} as $\phi_{\mathrm{SA}} \circ \mathrm{argmax} \circ [\cdot, :]$.

3.3. Training OC-pruner

Based on Equation (4), we **train** our OC-pruner, especially the core module, Slot Attention, by firstly reconstructing the unpruned vision tokens from the slots and then by minimizing the reconstruction error:

$$\boldsymbol{V}' = \phi_{\mathrm{RAR}}(\boldsymbol{V}_{\mathrm{p}}) \tag{9}$$

$$\min \phi_{\mathrm{r}}(\boldsymbol{V}', \boldsymbol{V}) \tag{10}$$

Here $\boldsymbol{V}' \in \mathbb{R}^{n \times c}$ is the reconstruction. ϕ_{RAR} is the random auto-regressive Transformer decoder [38]. As for why we do not choose other decoders, please refer to Section 2.3 for detailed reasons. ϕ_{r} is the reconstruction loss. Briefly, we implement the distance metric ϕ_{d} as $\phi_{\mathrm{RAR}} \circ \phi_{\mathrm{r}}$.

How to support any budget by training once? Mainstream VTP methods for VLMs support various pruning rates or budgets naturally, as they do not rely on learnt pruning criteria like our method. To overcome such an issue, we train OC-pruner with #slots queries s randomly sampled from typical budget values, e.g., $\{32, 64, 128, 192\}$.

How to preserve small yet important contents? Such reconstruction-based training objective Equation (10) minimizes the overall loss, thus vision tokens that are spatially small but contain important information might be ignored in the aggregation and decoding. To address it, we propose our novel Area-Weighted Mean-Squared Error (AW-MSE):

$$\phi_{\mathrm{r}}(\boldsymbol{V}', \boldsymbol{V}) = \frac{1}{snc} \sum_{s_i, n_i, c_i} \frac{1}{\sum\limits_{n_j} \boldsymbol{M}_{[s_i, n_j]}} ||\boldsymbol{V}' - \boldsymbol{V}||^2_{[\boldsymbol{M}_{[s_i, n_i]}, c_i]} \tag{11}$$

$$\text{where} \quad \boldsymbol{M} = \mathbb{1}[\boldsymbol{A} = \max_s(\boldsymbol{A})] \tag{12}$$

Here segmentation masks $\boldsymbol{M} \in \mathbb{R}^{s \times n}$ corresponding to slots $\boldsymbol{S}$ are calculated by $\max$ along the dimension s. And the area of each mask, i.e., $\sum_{n_j} \boldsymbol{M}_{[s_i, n_j]}$, is used to reciprocally weight the corresponding squared error, i.e, $||\boldsymbol{V}' - \boldsymbol{V}||^2_{[\boldsymbol{M}_{[s_i, n_i]}, c_i]}$. Please refer to Table 7 for the effects of MSE vs AW-MSE on the performance.

How to generalize to any datasets by training once? Object-centric representation is good at zero-shot generalization [23]. So we naively pre-train our OC-pruner on **40,000** images randomly sampled from COCO [14], i.e., no direct overlap with the evaluation. For training efficiency, we preprocess these images into vision tokens as the training data via VLM vision encoders. Despite evaluation images could follow quite different distributions, like synthetic vs real-world, even optical characters, our method, as shown in Section 4, performs surprisingly steadily.

Comment. The competition mechanism inside Slot Attention [19] intrinsically ensures the kept vision tokens have least information overlap, i.e., *exclusiveness*, while our training objective Equation (10) ensures the union of the kept tokens preserve the most information from the original unpruned tokens, i.e., *completeness*. These two aspects guarantee the most representative vision tokens are kept. In contrast, as discussed in Section 2.2, all existing methods rely on their handcrafted attention or similarity, which are not designed and optimized for such guarantee.

3.4. Instantiating OC-VTP

As shown in Figure 2, we insert the learnt OC-Pruner between the vision encoder and projector. Beside, the vision tokens from some middle layer of the vision encoder should also be fed into OC-pruner as the pruning reference. This is because to prune the vision tokens (from the layer), taking the middle layer vision tokens as the reference is better than taking themselves as the reference. Namely, the input $\boldsymbol{V}$ to the Slot Attention module ϕ_{SA} in Equation (6) should be replace with some middle layer vision tokens, while the input $\boldsymbol{V}$ to the selection operation in Equation (8) should still be the last layer vision tokens. Please refer to Table 6 for the effects of difference middle layers as reference.

Comment. Our OC-VTP operates between the vision encoder and projector. In contrast, as discussed in Section 2.2, some SotA methods [3, 4] take first-mover advantage by pruning vision tokens inside the vision encoder to save computation earlier and more; some SotA methods [29, 34] have access to the language tokens inside the LLM decoder for more task-aware, rather than task-agonistic pruning. Although theoretically possible, our OC-VTP, working without such techniques, still demonstrate consistent superiority, as shown in Section 4.

4. Experiments

We applied the pre-trained OC-Pruner to LLaVA-1.5, LLaVA-Next, and Qwen2.5-VL to test the results [1, 15, 16]. We use LMMs-Eval to run ten official benchmarks [33], including GQA, MMB, MME, POPE, SQA, $\mathrm{VQA}^{\mathrm{Text}}$, $\mathrm{VQA}^{\mathrm{v2}}$, VizWiz, MMMU, and SEED [5–8, 11, 12, 18, 20, 24, 31]. We compare our method with existing SotA approaches, FastV, SparseVLM, VisionZip, PyramidDrop, and HiPrune [4, 17, 29, 30, 34].

4.1. Results on Image Understanding

4.1.1. Results on LLaVA-1.5.

As shown in Table 1, our OC-Pruner on LLaVA-1.5 outperforms SotA methods on most benchmarks and achieves the highest average relative accuracy to the vanilla model. With only 11.1% vision tokens, OC-VTP reaches 95.5% of the vanilla accuracy, exceeding the second-best method

Method	GQA	MMB	MME	POPE	SQA	VQA^{Text}	VQA^{v2}	VizWiz	MMMU	SEED	Avg
Vanilla, 576 Tokens (**100%**)											
Vanilla$^{CVPR'24}$	61.9 100%	64.7 100%	1862 100%	85.9 100%	69.5 100%	58.2 100%	78.5 100%	55.8 100%	36.3 100%	58.6 100%	100%
Retain 192 Tokens (**33.3%**)											
FastV$^{ECCV'24}$	52.7 85.1%	61.2 94.6%	1612 86.6%	64.8 75.4%	67.3 96.8%	52.5 90.2%	67.1 85.5%	50.8 91.0%	34.3 94.5%	57.1 97.4%	89.7%
SparseVLM$^{ICML'25}$	57.6 93.1%	62.5 96.6%	1721 92.4%	83.6 97.3%	**69.1** **99.4%**	56.1 96.4%	**77.0** **98.1%**	50.6 90.7%	33.8 93.1%	55.8 95.2%	95.2%
VisionZip$^{CVPR'25}$	59.3 95.8%	63.0 97.4%	1783 95.8%	85.3 99.3%	68.9 99.1%	57.3 98.5%	76.8 97.8%	- -	**36.6** **101%**	56.4 96.2%	97.9%
PyramidDrop$^{CVPR'25}$	57.3 92.6%	63.3 97.8%	1797 96.5%	82.3 95.8%	69.0 99.3%	56.5 97.1%	75.1 95.7%	51.1 91.6%	- -	54.7 93.3%	95.5%
HiPrune$^{2025.08}$	59.2 95.6%	62.8 97.1%	**1814** **97.4%**	86.1 100%	68.9 99.1%	57.6 99.0%	76.7 97.7%	54.5 97.7%	36.4 100%	**63.6** **109%**	99.3%
OC-VTP^{Ours}	**59.5** **96.1%**	**63.4** **98.0%**	1808 97.1%	**86.3** **101%**	68.4 98.4%	**57.8** **99.3%**	76.7 97.7%	**54.5** **97.7%**	36.4 100%	63.4 108%	**99.3%**
OC-VTP$^{Ours}_{▲}$	59.6 96.3%	- -	- -	- -	69.0 99.3%	57.8 99.3%	77.4 98.6%	55.4 99.3%	- -	- -	-
Retain 128 Tokens (**22.2%**)											
FastV$^{ECCV'24}$	49.6 80.1%	56.1 86.7%	1490 80.0%	59.6 69.4%	60.2 86.6%	50.6 86.9%	61.8 78.7%	51.3 91.9%	34.9 96.1%	55.9 95.4%	85.2%
SparseVLM$^{ICML'25}$	56.0 90.5%	60.0 92.7%	1696 91.1%	80.5 93.7%	67.1 96.5%	54.9 94.3%	73.8 94.0%	50.2 90.0%	33.8 93.1%	53.4 91.1%	92.7%
VisionZip$^{CVPR'25}$	57.6 93.1%	62.0 95.8%	1762 94.6%	83.2 96.9%	68.9 99.1%	**56.8** **97.6%**	**75.6** **96.3%**	- -	**37.9** **104%**	54.9 93.7%	96.8%
PyramidDrop$^{CVPR'25}$	57.1 92.2%	61.6 95.2%	1761 94.6%	82.3 95.8%	68.4 98.4%	56.6 97.3%	72.9 92.9%	51.0 91.4%	- -	53.3 91.0%	94.3%
HiPrune$^{2025.08}$	57.3 92.6%	62.2 96.1%	**1782** **95.7%**	82.8 96.4%	68.3 98.3%	56.6 97.3%	74.9 95.4%	54.3 97.3%	36.7 101%	**61.0** **104%**	97.4%
OC-VTP^{Ours}	**58.0** **93.7%**	**62.2** **96.1%**	1761 94.6%	**85.3** **99.3%**	**69.0** **99.3%**	56.2 96.6%	75.1 95.7%	**54.7** **98.0%**	36.1 99.4%	60.9 104%	**97.7%**
OC-VTP$^{Ours}_{▲}$	58.4 94.3%	- -	- -	- -	69.0 99.3%	56.9 97.8%	76.2 97.1%	54.3 97.3%	- -	- -	-
Retain 64 Tokens (**11.1%**)											
FastV$^{ECCV'24}$	46.1 74.5%	48.1 74.3%	1256 67.5%	48.0 55.9%	51.1 73.5%	47.8 82.1%	55.0 70.1%	50.8 91.0%	34.0 93.7%	51.9 88.6%	77.1%
SparseVLM$^{ICML'25}$	52.7 85.1%	56.2 86.9%	1505 80.8%	75.1 87.4%	62.2 89.5%	51.8 89.0%	68.2 86.9%	50.4 90.3%	32.7 90.1%	51.1 87.2%	87.3%
VisionZip$^{CVPR'25}$	55.1 89.0%	**60.1** **92.9%**	1690 90.8%	77.0 89.6%	**69.0** **99.3%**	**55.5** **95.4%**	72.4 92.2%	- -	36.2 99.7%	52.2 89.1%	93.1%
PyramidDrop$^{CVPR'25}$	47.5 76.7%	58.8 90.9%	1561 83.8%	55.9 65.1%	69.0 99.3%	50.6 86.9%	69.2 88.2%	50.7 90.9%	- -	50.4 86.0%	85.3%
HiPrune$^{2025.08}$	53.6 86.6%	59.5 92.0%	1646 88.4%	73.0 85.0%	68.9 99.1%	54.9 94.3%	69.2 88.2%	54.4 97.5%	**36.7** **101%**	55.2 94.2%	92.6%
OC-VTP^{Ours}	**55.7** **90.0%**	59.8 92.4%	**1708** **91.7%**	**83.0** **96.6%**	68.9 99.1%	54.6 93.8%	**73.6** **93.8%**	**56.3** **101%**	35.6 98.1%	**57.9** **98.8%**	**95.5%**
OC-VTP$^{Ours}_{▲}$	56.2 90.8%	- -	- -	- -	69.0 99.3%	55.4 95.2%	75.1 95.7%	55.2 98.9%	- -	- -	-
Retain 50.7 Tokens (**8.8%**)											
OC-VTP^{Ours}	55.3 89.3%	56.8 87.8%	1581 84.9%	81.5 94.9%	68.6 98.7%	51.5 88.5%	67.4 85.9%	56.6 101%	35.4 97.5%	57.1 97.4%	**92.6%**

Table 1. Performance of OC-VTP on LLaVA-1.5. The vanilla number of vision tokens is 576. The first line of each method shows the benchmark accuracy, and the second line is the relative accuracy to the vanilla performance. The last column is the average proportion across all benchmarks. ▲ means that OC-VTP is trained on each individual benchmark's training set, ensuring no overlap with the scoring splits. "-" means the benchmark does not have a training set or the method does not have a published result on the benchmark. The 8.8% retained tokens is the average number of remained tokens across all benchmarks if the pruned tokens are not padded to 64 tokens.

VisionZip at 93.1%. At 22.2% tokens, the gap narrows but remains the best at 97.7% compared to the second-best HiPrune at 97.4%. **Given the fact that our method is only trained on COCO partial images, which has no direct overlap with these benchmarks, it is surprising that our method demonstrates such steady superiority**.

OC-VTP is designed as a train-once method, but it can also be trained individually on each benchmark that provides a training set. For fairness, we remove potential overlaps with the evaluation images. As MMB, MME, POPE, MMMU, and SEED are evaluation only, we do not train on these benchmarks. As expected, the performance on each benchmark yields additional improvement compared with our train-once OC-Pruner.

We also compare the results that do not pad the pruned tokens to the target budget s. When there are much less objects or object parts than #slots, some slots can be empty, thus the retained #tokens can be smaller than s. Across the ten benchmarks, the average number of retained tokens is 50.7 (8.8%) and the average performance proportion is 92.6%, which is even competitive with some SotA baselines that use the target number of tokens.

On VizWiz and SEED, pruned models can outperform the vanilla model. A possible reason is that redundant vision tokens somehow act as noise to image understanding.

4.1.2. Results on LLaVA-NeXT

OC-VTP achieves comparable results on LLaVA-NeXT with those on LLaVA-1.5 as shown in Table 2. When retaining only 5.6% vision tokens, our pre-trained OC-VTP reduces the performance by merely 5.0%. The performance gap between our method and the second-best SotA approach shows a consistent trend that at the 5.6% retaining ratio, OC-VTP is 1.8% better than VisionZip, but the gap narrows to 0.1% and 1.0% while retaining ratios increase, compared to HiPrune. **Thus our method works better than the baselines in heavy pruning cases**.

On SQA, the results first decrease then increase with increasing pruning ratio. At lower pruning ratios, tokens within the same cluster are pruned first, since these clusters represent the main object in the image and correspond to informative regions, performance initially drops. In this circumstance, the influence of noisy tokens becomes relatively stronger. While pruning continues, tokens from less important clusters (e.g., noise objects) are discarded, allowing the remaining representative object tokens to contribute more effectively, leading to a recovery of performance.

4.1.3. Results on Qwen2.5-VL

Qwen-2.5-VL uses a dynamic-resolution vision encoder with an adaptive token count [1], so we report the results by the retaining ratio rather than an exact number of retained tokens. As shown in Table 3, the performance gap between OC-VTP and the second-best SotA method narrows faster.

Method	GQA	MMB	POPE	SQA	VQA^{Text}	VizWiz	Avg
Vanilla, 2880 Tokens (**100%**)							
Vanilla$^{CVPR'24}$	64.2 100%	66.4 100%	86.2 100%	67.5 100%	61.3 100%	55.2 100%	100%
Retain 640 Tokens (**22.2%**)							
VisionZip$^{CVPR'25}$	**61.3** **95.5%**	66.3 99.8%	86.3 100%	68.1 101%	**60.2** **98.2%**	57.1 103%	99.7%
HiPrune$^{2025.08}$	60.6 94.4%	**67.0** **101%**	85.3 99.0%	68.0 101%	60.0 97.9%	59.9 109%	100%
OC-VTP^{Ours}	61.2 95.3%	64.8 97.6%	**86.9** 101%	**69.3** 103%	60.1 98.0%	**60.3** 109%	**101%**
OC-VTP$^{Ours}_{▲}$	62.8 97.8%	- -	- -	69.3 103%	61.1 99.7%	60.1 109%	-
Retain 320 Tokens (**11.1%**)							
VisionZip$^{CVPR'25}$	**59.3** **92.4%**	63.1 95.0%	82.1 95.2%	67.3 99.7%	**58.9** **96.1%**	56.2 102%	96.7%
HiPrune$^{2025.08}$	57.4 89.4%	**65.3** **98.3%**	78.9 91.5%	67.3 99.7%	57.6 94.0%	59.9 109%	96.9%
OC-VTP^{Ours}	58.1 90.5%	61.7 92.9%	**82.3** **95.5%**	**67.4** **99.9%**	58.2 94.9%	**59.9** **109%**	**97.0%**
OC-VTP$^{Ours}_{▲}$	59.6 92.8%	- -	- -	67.3 99.7%	58.4 95.3%	59.6 108%	-
Retain 160 Tokens (**5.6%**)							
VisionZip$^{CVPR'25}$	55.1 85.8%	**60.1** **90.5%**	77.0 89.3%	69.0 102%	55.5 90.5%	55.5 101%	93.1%
HiPrune$^{2025.08}$	52.5 81.8%	59.8 90.1%	67.7 78.5%	68.7 102%	51.2 83.5%	57.2 104%	89.9%
OC-VTP^{Ours}	**56.1** **87.4%**	60.0 90.4%	**78.9** **91.5%**	**69.2** **103%**	**55.7** **90.9%**	**59.0** **107%**	**94.9%**
OC-VTP$^{Ours}_{▲}$	57.6 89.7%	- -	- -	69.0 102%	56.8 92.7%	59.2 107%	-

Table 2. Performance of OC-VTP on LLaVA-NeXT. The vanilla number of vision tokens is 2880. ▲ means that OC-VTP is trained on each individual benchmark's training set, ensuring no overlap with the scoring splits. "-" means the benchmark does not have a training set for individual training.

At ratio 33.3%, OC-VTP is weaker than that of HiPrune. The overall performance on Qwen2.5-VL is also slightly weaker than on LLaVA VLMs. We believe that LLaVA produces a fixed number of tokens, so OC-Pruner can learn a stable pattern that maps a target budget to a fixed number of slots during training. In Qwen2.5-VL, the token count varies across images, so a train-once OC-Pruner can be misaligned with the actual budget, which weakens object-level feature aggregation and reduces pruning optimality.

4.2. Results on Efficiency

4.2.1. Results on FLOPs

We evaluate OC-VTP on LLaVA-1.5 and LLaVA-NeXT, and report the computation cost of the prefill FLOPs (MAC=2). The results are summarized in Table 4.

Compared with the vanilla model, our method effectively reduces computational overhead while maintaining performance. Specifically, assuming the prompt token length is 32, for LLaVA-1.5 the pruned model (retaining 64 tokens) requires only 0.97 T FLOPs compared to 6.30 T FLOPs in the original model, achieving a 6.5× reduction. Similarly,

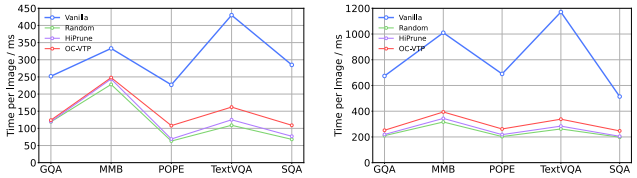
Method	MMB	POPE	SQA	VizWiz	Avg
Vanilla, **100%** Tokens					
Vanilla$^{\text{CVPR'24}}$	77.3 100%	87.0 100%	80.4 100%	68.3 100%	100%
Retain **33.3%** Tokens					
VisionZip$^{\text{CVPR'25}}$	74.9 96.9%	85.4 98.2%	**80.1** **99.6%**	67.1 98.2%	98.2%
HiPrune$^{\text{2025.08}}$	**75.8** **98.1%**	86.0 98.9%	80.0 99.5%	**67.5** **98.8%**	**98.8%**
OC-VTP$^{\text{Ours}}$	75.3 97.4%	**86.4** **99.3%**	79.6 99.0%	67.4 98.7%	98.6%
OC-VTP$^{\text{Ours}}_{\blacktriangle}$	- -	- -	79.8 99.3%	67.5 98.8%	-
Retain **22.2%** Tokens					
VisionZip$^{\text{CVPR'25}}$	73.5 95.1%	84.6 97.2%	80.0 99.5%	66.3 97.1%	97.2%
HiPrune$^{\text{2025.08}}$	74.0 95.7%	84.7 97.4%	**80.3** **99.9%**	66.5 97.4%	97.6%
OC-VTP$^{\text{Ours}}$	**74.6** **96.5%**	**85.0** **97.7%**	79.8 99.3%	**66.8** **97.8%**	**97.8%**
OC-VTP$^{\text{Ours}}_{\blacktriangle}$	- -	- -	80.0 99.5%	67.1 98.2%	-
Retain **11.1%** Tokens					
VisionZip$^{\text{CVPR'25}}$	67.8 87.7%	80.2 92.2%	**79.5** **98.9%**	62.8 91.9%	92.7%
HiPrune$^{\text{2025.08}}$	69.6 90.0%	80.4 92.4%	78.9 98.1%	64.2 94.0%	93.6%
OC-VTP$^{\text{Ours}}$	**70.9** **91.7%**	**81.8** **94.0%**	78.0 97.0%	**64.2** **94.0%**	**94.2%**
OC-VTP$^{\text{Ours}}_{\blacktriangle}$	- -	- -	78.2 97.3%	66.1 96.8%	-

Table 3. Performance of OC-VTP on Qwen2.5-VL. Qwen2.5-VL uses adaptive token counts, retaining ratio is used for evaluation. ▲ means that OC-VTP is trained on each individual benchmark's training set, ensuring no overlap with the scoring splits. "-" means the benchmark does not have a training set for individual training.

Method	Vanilla	Pruned	OC-Pruner
LLaVA-1.5	6.30 T	0.97 T	5.97 G
LLaVA-NeXT	33.76 T	1.95 T	23.82 G

Table 4. Prefill FLOPs (MAC=2). Computed per image for the prefill stage with 32 text tokens. "Vanilla" is the unpruned VLM, "Pruned" is the VLM after token pruning (64 vision tokens for LLaVA-1.5, and 160 for LLaVA-NeXT), and "OC-Pruner" is the additional cost of our pruner. T = teraFLOPs, G = gigaFLOPs.

for LLaVA-NeXT, the FLOPs drop from 33.76 T to 1.95 T, corresponding to a 17× reduction. Meanwhile, our OC-Pruner only needs 5.97 G and 23.82 G FLOPs, which incurs nearly zero additional computational cost compared to the VLM backbone. Overall, the results demonstrate that the proposed OC-Pruner introduces only small overhead while substantially improving efficiency and preserving performance, making it highly scalable to large VLMs.



(a) Inference time per image of LLaVA-1.5 with 64 retrained vision tokens.

(b) Inference time per image of LLaVA-NeXT with 160 tokens.

Figure 3. Inference time per image (ms). Average inference time for each case/image is calculated from total evaluation using LMMs-Eval on a single V100-32G GPU with batch size = 1. Notably, *Random* is a random pruning baseline with no extra cost.

4.2.2. Results on Latency

We measure end-to-end inference latency per image on a single NVIDIA V100-32GB GPU with identical generation settings (batch size = 1), using LLaVA-1.5 with 64 retained vision tokens and LLaVA-NeXT with 160 vision tokens. As shown in Figure 3, in GQA, MMB, POPE, VQA$^{\text{Text}}$ and SQA, pruning significantly reduces latency compared to vanilla VLMs. To better compare the effect of pruning, we also include a **Random pruner** baseline, which involves almost no extra overhead by randomly discarding vision tokens. On LLaVA-1.5 (left figure), the Random pruner and HiPrune achieve very similar latency and reduce the inference time per image by nearly 60% relative to the vanilla model. Our OC-VTP is slightly slower but still comparable, yielding a 55% (305.4 ms to 138.2 ms) reduction in both inference time and latency. On LLaVA-NeXT (right figure), pruning brings even greater benefits that HiPrune and Random pruner reduce the inference time by almost 70%, while OC-VTP achieves a similar 65% (811.8 ms to 287.3 ms) reduction. We expect these gains to be further improved when the pruners are deployed on more powerful GPUs, where computation dominates the system overhead.

4.3. Results on Visualization

With a budget of 64 tokens on LLaVA-1.5, OC-Pruner retains one token per slot. As shown in Figure 4, the selected tokens concentrate on different object centers (e.g., bike, cars, buildings, aircraft, animals, and signs). Large objects receive multiple slots that cover different parts, while small but important objects or background objects (such as the humans in (b) and (e), the mountain in the background in (c) and (f), and birds in (d)) still get at least one token, and backgrounds (sky, water, grass) are largely suppressed in several tokens unless they are important to define the scene. This object diversity and alignment match the expected behavior of OCL characteristics, explaining the reasons why OC-VTP preserving most of the performance.

Typically, failure cases occur when a slot lands near, but not exactly on the instance. We believe this is because the objects are very small or have low contrast, making them

(a) Tokens from the bike, cars, ground, trees, and buildings.

(b) Tokens from the aircraft, humans, the tower, clouds, and the runway.

(c) Tokens from trees, the mountain, the sky, and the path.

(d) Tokens from the house, trees, birds, the sky, clouds and the lake.

(e) Tokens from signs, buildings, the human, the minivan, and lanes.

(f) Tokens from sheep, grasses, the trunk, leaves and the mountain.

Figure 4. **OC-VTP visualization results.** OC-Pruner retains one token each slot, and the slots represent different objects in the scene. The retaining budget is 64, and the test is conducted on LLaVA-1.5.

slots top-k	GQA	MMB	MME	POPE	VQA$^{\text{Text}}$	MMMU	SEED	Avg
Vanilla, 576 Tokens (**100%**)								
Vanilla	61.9 100%	64.7 100%	1862 100%	85.9 100%	58.2 100%	36.3 100%	58.6 100%	100%
Retain 192 Tokens (**33.3%**)								
32 Slots - Top 6	56.4 91.1%	60.9 94.1%	1712 91.9%	83.9 97.7%	55.2 94.8%	33.6 92.6%	58.8 100%	94.6%
64 Slots - Top 3	58.1 93.9%	61.5 95.1%	1738 93.3%	85.6 99.7%	55.6 95.5%	35.2 97.0%	60.0 102%	96.7%
192 Slots - Top 1	**59.5** **96.1%**	**63.4** **98.0%**	**1808** **97.1%**	**86.3** **101%**	**57.8** **99.3%**	**36.4** **100%**	**63.4** **108%**	**99.9%**
Retain 128 Tokens (**22.2%**)								
32 Slots - Top 4	55.2 89.2%	58.1 89.8%	1678 90.1%	82.8 96.4%	54.3 93.3%	35.9 98.9%	54.0 92.2%	92.8%
64 Slots - Top 2	56.9 91.9%	61.3 94.7%	1729 92.9%	84.0 97.8%	54.9 94.3%	35.8 98.6%	59.1 101%	95.9%
128 Slots - Top 1	**58.0** **93.7%**	**62.2** **96.1%**	**1761** **94.6%**	**85.3** **99.3%**	**55.6** **95.5%**	**36.1** **99.4%**	**60.9** **104%**	**97.5%**
Retain 64 Tokens (**11.1%**)								
32 Slots - Top 2	53.6 86.6%	57.8 89.3%	1612 86.6%	81.7 95.1%	52.9 90.9%	35.3 97.2%	54.0 92.2%	91.1%
64 Slots - Top 1	**55.7** **90.0%**	**59.8** **92.4%**	**1708** **91.7%**	**83.0** **96.6%**	**54.6** **93.8%**	**35.6** **98.1%**	**57.9** **98.8%**	**94.5%**

Table 5. Effect of #slots s and top-k on LLaVA-1.5. Different combinations of s - k determine the retained #tokens.

s	Layer 9	Layer 10	Layer -2
192	99.9%	99.9%	98.1%
128	97.5%	97.2%	95.8%
64	94.5%	93.6%	93.5%

Table 6. Performance of OC-VTP at different layers. Relative performance on LLaVA-1.5 averaged over the seven benchmarks (GQA, MMB, MME, POPE, VQA$^{\text{Text}}$, MMMU, and SEED).

s	MSE	AW-MSE
192	99.3%	99.9%
128	96.8%	97.5%
64	92.4%	94.5%

Table 7. Effect of OC-pruner training losses. Relative performance on LLaVA-1.5 averaged over the seven benchmarks (GQA, MMB, MME, POPE, VQA$^{\text{Text}}$, MMMU, and SEED). AW-MSE is always better than MSE at different token budgets.

difficult for the model to distinguish from the background. Another failure case appears in image (f), where the fence is omitted. We believe that this happens because when slot attention computes similarity with vision tokens, the fence is considered less important, and the slot instead selects tokens corresponding to the grass in the region.

4.4. Ablations

Slot count and top-k combinations. We compare a different number of slots s and the top-k for each slot while keeping the same token budget $s \times k$. As shown in Table 5, using smaller k outperforms other combinations across all seven benchmarks. We therefore adopt $k = 1$, which yields the best performance. In one slot cluster, similar tokens represent an object, so selecting more than one token per slot involves redundancy.

Pruner insertion layers. We compare OC-VTP at three insertion positions in the vision encoder that Layer 9, Layer 10, and Layer -2 under the same token budgets. Following the discoveries in HiPrune [17], Layer 9 is recognized as the most **information-dense**, Layer 10 is the most **sparse**, and Layer -2 is near the encoder output. As shown in Table 6, inserting at Layer 9 consistently yields the best results across budgets, so we adopt it as the first choice. Because Layer 9 is the most information-dense, it can represent the most objects with the fewest tokens. In very dense clusters, a single token is enough to represent the whole cluster.

Loss functions. We compare AW-MSE with plain MSE on LLaVA-1.5, averaging the relative scores on seven benchmarks (GQA, MMB, MME, POPE, VQA$^{\text{Text}}$, MMMU, and SEED). As shown in Table 7, training OC-Pruner with AW-MSE yields the highest accuracy in all token budgets, so we adopt AW-MSE as default. While the OCL module reconstructs encoder features, a plain MSE objective may underweight slots covering small but important regions. The inverse-area re-weighting in AW-MSE can solve this issue and preserve these informative areas.

5. Conclusion

We introduce OC-VTP, an Object-Centric Learning-based Vision Token Pruning method for VLMs. We prune vision tokens by utilizing Object-Centric Learning especially the Slot Attention module to select the most representative vision tokens, with guarantee for the first time. Evaluated on LLaVA-1.5, LLaVA-NeXT, and Qwen2.5-VL, OC-VTP delivers great FLOPs savings and competitive latency compared to existing strong baselines, while preserving accuracy. In future work, we will extend OC-VTP to video and other tasks, and explore pruning in the decoder as well as text-token-assisted vision tokens pruning modules.

References

[1] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025. 1, 2, 4, 6

[2] Ondrej Biza, Sjoerd van Steenkiste, Mehdi SM Sajjadi, Gamaleldin Elsayed, Aravindh Mahendran, and Thomas Kipf. Invariant Slot Attention: Object Discovery with Slot-Centric Reference Frames. In *International Conference on Machine Learning*, pages 2507–2527, 2023. 2

[3] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *ICLR*, 2023. 1, 3, 4

[4] Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 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*, pages 19–35. Springer, 2024. 1, 2, 3, 4

[5] Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, et al. Mme: A comprehensive evaluation benchmark for multimodal large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. 4

[6] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913, 2017.

[7] Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3608–3617, 2018.

[8] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019. 2, 4

[9] Baoxiong Jia, Yu Liu, and Siyuan Huang. Improving Object-centric Learning with Query Optimization. In *The Eleventh International Conference on Learning Representations*, 2023. 2

[10] Ioannis Kakogeorgiou, Spyros Gidaris, Konstantinos Karantzalos, and Nikos Komodakis. Spot: Self-Training with Patch-Order Permutation for Object-Centric Learning with Autoregressive Transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22776–22786, 2024. 2

[11] Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. Seed-bench: Benchmarking multimodal large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13299–13308, 2024. 4

[12] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. *arXiv preprint arXiv:2305.10355*, 2023. 2, 4

[13] Yanwei Li, Yuechen Zhang, Chengyao Wang, Zhisheng Zhong, Yixin Chen, Ruihang Chu, Shaoteng Liu, and Jiaya Jia. Mini-gemini: Mining the potential of multi-modality vision language models. *arXiv preprint arXiv:2403.18814*, 2024. 1

[14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 4

[15] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023. 1, 2, 4

[16] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llavanext: Improved reasoning, ocr, and world knowledge, 2024. 1, 2, 4

[17] Jizhihui Liu, Feiyi Du, Guangdao Zhu, Niu Lian, Jun Li, and Bin Chen. Hiprune: Training-free visual token pruning via hierarchical attention in vision-language models. *arXiv preprint arXiv:2508.00553*, 2025. 1, 2, 3, 4, 8

[18] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pages 216–233. Springer, 2024. 2, 4

[19] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in neural information processing systems*, 33:11525–11538, 2020. 2, 3, 4

[20] Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521, 2022. 2, 4

[21] Anna Manasyan, Maximilian Seitzer, Filip Radovic, Georg Martius, and Andrii Zadaianchuk. Temporally consistent object-centric learning by contrasting slots. In *Proceedings*

of the Computer Vision and Pattern Recognition Conference, pages 5401–5411, 2025. 2

[22] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 2

[23] Maximilian Seitzer, Max Horn, Andrii Zadaianchuk, Dominik Zietlow, Tianjun Xiao, Carl-Johann Simon-Gabriel, Tong He, Zheng Zhang, Bernhard Schölkopf, Thomas Brox, et al. Bridging the gap to real-world object-centric learning. In *The Eleventh International Conference on Learning Representations (ICLR 2023)*. OpenReview, 2023. 2, 4

[24] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8317–8326, 2019. 2, 4

[25] Dingjie Song, Wenjun Wang, Shunian Chen, Xidong Wang, Michael X Guan, and Benyou Wang. Less is more: A simple yet effective token reduction method for efficient multimodal llms. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 7614–7623, 2025. 1, 2, 3

[26] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. 2

[27] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024. 2

[28] Ziyi Wu, Jingyu Hu, Wuyue Lu, Igor Gilitschenski, and Animesh Garg. SlotDiffusion: Object-Centric Generative Modeling with Diffusion Models. *Advances in Neural Information Processing Systems*, 36:50932–50958, 2023. 2

[29] Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, and Dahua Lin. Conical visual concentration for efficient large vision-language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pages 14593–14603, 2025. 1, 2, 3, 4

[30] Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. Visionzip: Longer is better but not necessary in vision language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19792–19802, 2025. 1, 2, 3, 4

[31] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9556–9567, 2024. 4

[32] Andrii Zadaianchuk, Maximilian Seitzer, and Georg Martius. Object-Centric Learning for Real-World Videos by Predicting Temporal Feature Similarities. *Advances in Neural Information Processing Systems*, 36, 2024. 2

[33] Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, et al. Lmms-eval: Reality check on the evaluation of large multimodal models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 881–916, 2025. 4

[34] Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis A Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. Sparsevlm: Visual token sparsification for efficient vision-language model inference. In *Forty-second International Conference on Machine Learning*, 2025. 1, 2, 3, 4

[35] Rongzhen Zhao, Jian Li, Juho Kannala, and Joni Pajarinen. Predicting video slot attention queries from random slot-feature pairs. *arXiv preprint arXiv:2508.22772*, 2025. 2

[36] Rongzhen Zhao, Vivienne Wang, Juho Kannala, and Joni Pajarinen. Vector-Quantized Vision Foundation Model for Object-Centric Learning. In *ACM Multimedia*, 2025. 2

[37] Rongzhen Zhao, Wenyan Yang, Juho Kannala, and Joni Pajarinen. Smoothing Slot Attention Iterations and Recurrences. *arXiv:2508.05417*, 2025. 2

[38] Rongzhen Zhao, Yi Zhao, Juho Kannala, and Joni Pajarinen. Slot Attention with Re-Initialization and Self-Distillation. In *ACM Multimedia*, 2025. 2, 4

[39] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023. 1