

# Beyond Redundancy: Information-Conserving Semantic Packaging for Vision-Language Models

Xin Xiao and Jiang Zhong\*

School of Computer Science, Chongqing University  
{20241401023@stu.cqu.edu.cn, zhongjiang@cqu.edu.cn}

## Abstract

Vision-Language Models (VLMs) achieve strong multimodal understanding but suffer from high inference costs due to redundant vision tokens. Existing hard-pruning methods discard low-scoring tokens directly, inevitably sacrificing fine-grained visual context. We propose **SPack**, a training-free token compression method that reframes pruning as **information-energy redistribution** rather than deletion. Instead of discarding pruned tokens, SPack transfers their semantic mass to retained tokens, enforcing a global principle we term **Information Energy Conservation (IEC)**: the total information contained in redundant tokens is strictly conserved and fully transferred as information gain to the retained components. To realize this conservation, we introduce Optimal Transport (OT) theory, which treats removed energy as mass density and retained tokens as receiver slots. Solving the optimal transport plan under the Wasserstein distance yields a minimum-cost mapping that precisely transfers geometric structure and higher-order moments according to semantic affinity. Concretely, we estimate token importance via relation attention, semantic affinity, and value-space energy, then adaptively route pruned tokens to multiple retained tokens using OT, producing hyperdense tokens that preserve both local semantics and global context. Extensive experiments demonstrate that SPack consistently achieves a superior accuracy-efficiency trade-off compared to state-of-the-art methods.

## 1 Introduction

Vision-language models (VLMs) have achieved impressive progress in visual instruction following, image understanding, and multimodal reasoning (Liu et al., 2023; Tong et al., 2024; Meng et al., 2025; Li et al., 2025b; Tang et al., 2026). A common paradigm is to encode visual inputs into a se-

\*Corresponding author.

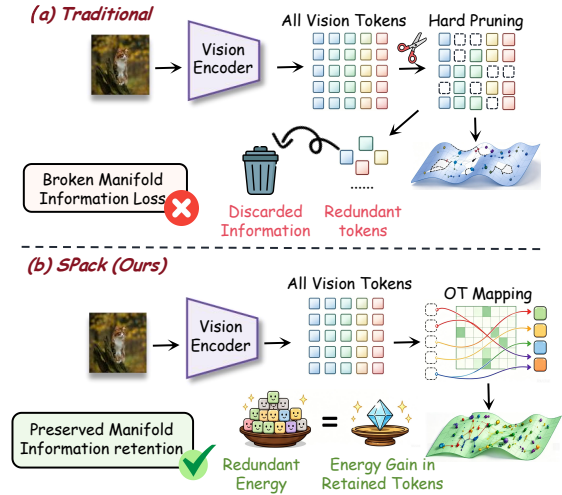


Figure 1: Comparison of traditional token pruning and our SPack. Unlike discard-or-merge strategies that lose or distort information, SPack redistributes redundant information energy to retained tokens via OT-based conservation mapping.

quence of visual tokens and feed them into the language model together with textual instructions (Dai et al., 2023; Liu et al., 2024a; Chen et al., 2024b; Wang et al., 2024; Liu et al., 2026). While effective, this design introduces substantial computational overhead. In particular, high-resolution images inputs often contain hundreds or thousands of visual tokens, resulting in substantial computational and memory cost (Chen et al., 2024a; Dong et al., 2026; Tong et al., 2026). Therefore, reducing redundant visual tokens has become an important direction for efficient VLM inference.

Recent studies have focused on accelerating the inference of VLMs by reducing visual tokens while preserving essential information. For instance, FastV observes that visual tokens can be aggressively reduced after early Large Language Model (LLM) layers (Chen et al., 2024a). LLaVA-PruMerge merges redundant visual tokens through adaptive token reduction (Shang et al., 2025). SparseVLM performs text-guided visual token sparsifi-

cation (Zhang et al., 2024b), while PDrop removes visual redundancy in a pyramid manner (Xing et al., 2025). VisionZip further shows that long visual token sequences are often unnecessary for maintaining strong performance (Yang et al., 2025). More recent methods investigate spatial-temporal token trimming (Zhuang et al., 2025), recoverable token compression guided by text (Chen et al., 2025b), visual token withdrawal (Lin et al., 2025), CLS-attention-based pruning (Zhang et al., 2024a), and multimodal coverage maximization (Dong et al., 2026). These studies demonstrate that visual redundancy is widespread in VLMs and can be exploited for acceleration.

Despite their effectiveness, most existing methods still treat pruning primarily as a *deletion* or *selection* problem. Tokens with low importance scores are removed, while the retained tokens are expected to implicitly preserve the necessary information. However, this deletion-centric view leaves a fundamental question unanswered: *where does the information carried by removed tokens go?* Even tokens regarded as redundant may contain non-zero semantic, spatial, or contextual information. Directly discarding them can break the global information budget of the visual representation, especially under aggressive token reduction. Recent analyses further suggest that visual redundancy emerges dynamically through inter-token interactions and information flow across layers (Tong et al., 2026), indicating that redundancy should be understood as a redistribution phenomenon rather than an isolated token property.

In this paper, we rethink VLM token pruning from the perspective of *Information Energy Conservation*, as illustrated in Fig. 1. Rather than simply deleting redundant visual tokens, we view pruning as a conservation-aware redistribution process, where the information carried by removed tokens is transferred to retained tokens to increase their information density and preserve useful visual semantics. To realize this idea, we formulate pruning as an Optimal Transport (OT) problem (Villani et al., 2009; Cuturi, 2013): removed tokens are treated as source mass, retained tokens as receiving slots, and semantic-geometric distances define the transport cost. The resulting Wasserstein transport plan provides an explicit minimum-cost mapping from removed tokens to retained tokens, preserving the total information energy while maintaining token geometry and reducing destructive information loss. Our contributions are summarized as follows:

- We propose an Information Energy Conservation perspective for VLM token pruning, viewing pruning as information energy redistribution from removed tokens to retained tokens rather than simple token deletion.
- We formulate conservation-aware visual token pruning as an Optimal Transport problem, establishing a minimum-cost mapping from removed tokens to retained tokens.
- Extensive experiments show that the proposed method improves inference efficiency while preserving strong multimodal performance under aggressive token reduction.

## 2 Related Work

### 2.1 Vision-language Models.

Recent VLMs typically connect visual encoders with large language models by converting images into sequences of visual tokens. LLaVA demonstrates that CLIP visual features can be effectively aligned with an LLM through a lightweight projector and visual instruction tuning (Liu et al., 2023, 2024a). InstructBLIP improves instruction-following ability by using instruction-aware Q-Former queries to extract task-relevant visual features from frozen image encoders (Dai et al., 2023). InternVL scales open-source multimodal models with strong visual encoders and progressive vision-language alignment, while InternVL3 further advances this line through native multimodal pre-training and improved test-time recipes (Chen et al., 2024c; Zhu et al., 2025). Qwen2-VL and Qwen2.5-VL enhance dynamic-resolution perception, document understanding, object localization, and long-video comprehension, making them strong general-purpose VLM backbones (Wang et al., 2024; Bai et al., 2025). Despite their strong performance, these models still rely on dense visual token sequences, which makes token efficiency a central challenge for practical deployment.

### 2.2 Vision Token Compression.

The large number of visual tokens in modern VLMs introduces substantial computational overhead, especially during prefilling. Existing methods prune, merge, or sparsify visual tokens at different stages: FastV removes tokens after early LLM layers (Chen et al., 2024a); LLaVA-PruMerge adaptively reduces tokens via pruning and merging (Shang et al., 2025); SparseVLM performs

text-guided sparsification (Zhang et al., 2024b); PyramidDrop progressively removes redundancy across layers (Xing et al., 2024); VisionZip exploits that long visual sequences are often unnecessary (Yang et al., 2025); and recent works further improve adaptability: V<sup>2</sup>Drop removes low-variation tokens during inference (Chen et al., 2025a). MMTOK (Dong et al., 2026) leverage both vision and text tokens to select informative vision tokens by the coverage criterion. FlowCut studies redundancy through layer-wise information flow (Tong et al., 2026). We position SPack within the broader context of efficient VLM inference. Unlike SCBench (Li et al., 2025a) and OmniKV (Hao et al., 2025), which optimize KV caches or dynamically select contexts, SPack operates at the visual-token level prior to cache generation and redistributes information rather than merely selecting tokens. Although these methods reveal substantial visual redundancy, most follow a hard selection paradigm where discarded tokens no longer contribute to subsequent reasoning. In contrast, SPack treats pruned tokens as reusable semantic mass and redistributes their information into retained tokens, yielding hyper-dense visual representations rather than merely shortening the token sequence.

### 3 Methodology

**Overview.** SPack compresses visual tokens by preserving and redistributing useful semantic information rather than simply removing redundant tokens. As shown in Fig. 2, SPack first estimates token energy from global attention, semantic relevance, and representation density, and then transfers information from pruned tokens to retained tokens via an optimal-transport-inspired mapping. To ensure stable and adaptive compression, SPack further combines cumulative energy tracking, entropy-based budget allocation, and semantic packing to enhance both retained patch tokens and the CLS token.

#### 3.1 Problem Formulation

Given an input image  $I$ , a vision encoder first converts it into a sequence of visual tokens  $\mathbf{X}^0 = [\mathbf{x}_{\text{cls}}^0, \mathbf{x}_1^0, \dots, \mathbf{x}_N^0] \in \mathbb{R}^{(N+1) \times d}$ , where  $\mathbf{x}_{\text{cls}}$  denotes the global token and  $N$  is the number of patch tokens. The visual tokens are then projected into the language embedding space and concatenated with textual tokens for multimodal reasoning. Since the self-attention cost grows quadratically with sequence length, reducing  $N$  is essential for

efficient inference.

The goal of SPack is to compress the visual sequence from  $N$  tokens to  $K$  tokens, where  $K \ll N$ , while preserving the information required by downstream reasoning. Different from hard pruning, which directly removes low-scoring tokens, SPack assumes that a removed token may still contain non-zero semantic information. Therefore, the compression objective is not only to select important tokens, but also to transfer useful information from pruned tokens to retained tokens:

$$\begin{aligned} \mathbf{X}^L &= [\mathbf{x}_{\text{cls}}^L, \mathbf{x}_1^L, \dots, \mathbf{x}_N^L], \\ \tilde{\mathbf{X}} &= [\tilde{\mathbf{x}}_{\text{cls}}, \tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_K]. \end{aligned} \quad (1)$$

Here,  $\tilde{\mathbf{X}}$  is expected to be shorter but denser: each retained token should encode both its own information and the redistributed information from removed tokens.

#### 3.2 Information Energy Conservation

We introduce the principle of *Information Energy Conservation*. Let  $\mathcal{K}$  and  $\mathcal{P}$  denote the retained and pruned token sets, respectively. For each token  $\mathbf{x}_i$ , we define an information energy score  $E_i$  that measures its contribution to global perception and multimodal reasoning. A conventional pruning method keeps  $\mathcal{K}$  and discards  $\mathcal{P}$ :

$$\tilde{\mathbf{X}} = \{\mathbf{x}_i \mid i \in \mathcal{K}\}. \quad (2)$$

This implicitly assumes  $E_i \approx 0$  for all  $i \in \mathcal{P}$ , which is often too strong under aggressive compression. SPack instead enforces the following conservation constraint:

$$\sum_{i \in \mathcal{P}} E_i \approx \sum_{j \in \mathcal{K}} \Delta E_j, \quad (3)$$

where  $\Delta E_j$  is the information gain injected into retained token  $\mathbf{x}_j$ . This formulation transforms visual token compression from a deletion problem into an information redistribution problem. The retained tokens are not merely selected tokens; they are semantic carriers that absorb useful information from the removed tokens.

#### 3.3 Multi-Criteria Energy Estimation

SPack estimates token energy using three complementary signals: global attention, semantic relevance, and representation density. At a vision encoder layer  $\ell$ , let  $\mathbf{A}^\ell$  be the multi-head self-attention matrix. We average attention over heads

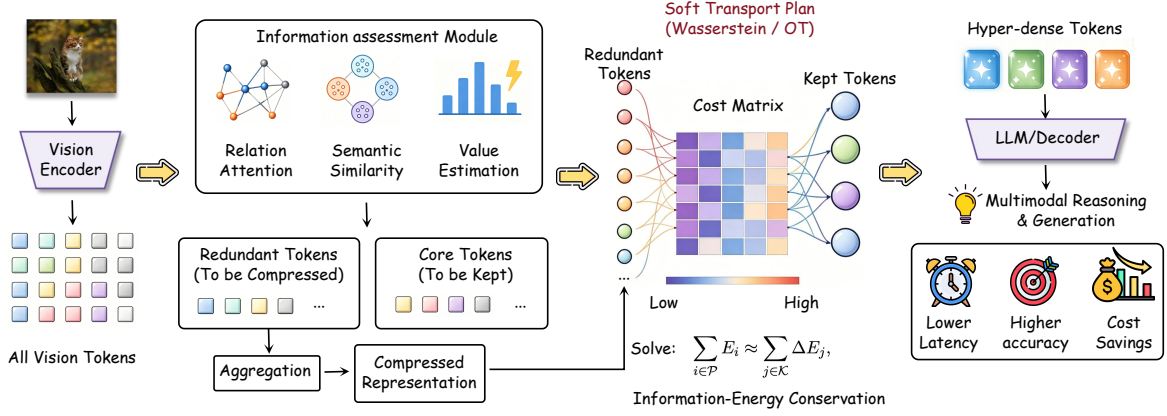


Figure 2: Overview of SPack, a training-free token compression framework for VLMs that scores visual tokens, adaptively separates retained and redundant tokens, and uses optimal transport to redistribute redundant semantic information into compact retained tokens. By conserving information-energy across token-to-token transfer and CLS enhancement, SPack achieves higher inference efficiency while preserving visual-semantic structure and downstream accuracy.

and use the CLS token as a global information probe:

$$a_i^\ell = \frac{1}{H} \sum_{h=1}^H \mathbf{A}_{\text{cls},i}^{\ell,h}, \quad (4)$$

where  $H$  is the number of attention heads. The attention term measures how much each patch token contributes to the global visual representation.

Attention alone can be noisy, so we further compute semantic relevance in the value space. Let  $\mathbf{v}_{\text{cls}}^\ell$  and  $\mathbf{v}_i^\ell$  denote the value vectors of the CLS token and patch token  $i$ . The semantic relevance is computed as

$$s_i^\ell = \text{Softmax} \left( \frac{(\mathbf{v}_{\text{cls}}^\ell)^\top \mathbf{v}_i^\ell}{\sqrt{d}} \right). \quad (5)$$

Finally, we use the value norm to estimate representation density:

$$e_i^\ell = \|\mathbf{v}_i^\ell\|_2. \quad (6)$$

The intuition is that a token with a larger value norm can contribute more strongly to the attention output and thus carries richer representational energy.

We normalize these terms into distributions:

$$p_i^t = \frac{t_i}{\sum_j t_j + \epsilon}, \quad t \in \{a, s, e\}. \quad (7)$$

These normalized distributions provide different views of token importance:  $p^a$  captures global dependency,  $p^s$  captures semantic alignment, and  $p^e$  captures information density.

### 3.4 Optimal-Transport Energy Mapping

SPack redistributes token energy with a Wasserstein optimal-transport mapping before pruning. We treat the normalized value norm  $p^e$  as information mass, and define the ground cost between source token  $j$  and receiver token  $i$  using global attention and semantic relevance:

$$C_{ij} = |p_i^a - p_j^s|. \quad (8)$$

The corresponding Wasserstein objective is

$$W_C(\mathbf{u}, \mathbf{v}) = \min_{\Pi \in \mathcal{U}(\mathbf{u}, \mathbf{v})} \sum_{i,j} \Pi_{ij} C_{ij}, \quad (9)$$

where  $\mathcal{U}(\mathbf{u}, \mathbf{v})$  is the set of couplings with prescribed marginals. For efficient inference, we use an entropy-regularized, row-normalized approximation:

$$T_{ij} = \frac{\exp(-\lambda C_{ij})}{\sum_k \exp(-\lambda C_{ik}) + \epsilon}, \quad (10)$$

where  $\lambda = 5$  in our implementation. This kernel assigns larger transport weights to token pairs with lower semantic-attention cost, approximating the minimum-cost Wasserstein flow without iterative solving. The redistributed value energy is computed as

$$\hat{p}_i^e = \sum_j T_{ij} p_j^e. \quad (11)$$

We combine this transported mass with the original value energy:

$$q_i = \frac{p_i^e + \hat{p}_i^e}{2}, \quad q_i \leftarrow \frac{q_i}{\sum_j q_j + \epsilon}. \quad (12)$$

The final conservation-aware score is then

$$r_i = (p_i^a + p_i^s)q_i, \quad r_i \leftarrow \frac{r_i}{\sum_j r_j + \epsilon}. \quad (13)$$

This score favors tokens that are globally attended, semantically aligned, and capable of carrying re-distributed value energy.

### 3.5 Cumulative Energy Tracking

Layer-wise token scores can fluctuate due to local attention noise. Making pruning decisions from a single layer may therefore remove tokens that become important in later layers. To stabilize the selection, SPack maintains a cumulative energy score:

$$\mathbf{R}^\ell = \mu_\ell \mathbf{R}^{\ell-1} + (1 - \mu_\ell) \mathbf{r}^\ell. \quad (14)$$

Here,  $\mathbf{r}^\ell$  is the current conservation-aware score, and  $\mu_\ell$  is a momentum coefficient. We use a larger momentum in deeper layers because deeper representations are more semantically stable. This cumulative mechanism allows SPack to preserve tokens that are consistently informative across layers rather than only salient at a single layer.

When tokens are pruned, the cumulative score is gathered according to the retained indices:

$$\mathbf{R}^\ell \leftarrow \text{Gather}(\mathbf{R}^\ell, \mathcal{K}^\ell). \quad (15)$$

Thus, score tracking remains aligned with the dynamically shortened token sequence.

### 3.6 Adaptive Token Budget Allocation

Instead of pruning a fixed number of tokens at every layer, SPack adaptively determines the pruning amount according to the concentration of global attention. Given normalized CLS attention  $\mathbf{p}^a$ , we compute its entropy:

$$H(\mathbf{p}^a) = - \sum_i p_i^a \log(p_i^a + \epsilon). \quad (16)$$

The normalized entropy ratio is

$$\rho_H = \frac{H(\mathbf{p}^a)}{\log N}. \quad (17)$$

When  $\rho_H$  is high, attention is diffuse and the model has not yet formed a clear concentration pattern, so SPack prunes conservatively. When  $\rho_H$  is low, attention is concentrated on a smaller set of tokens, indicating stronger redundancy. Given target token

number  $K$  and remaining pruning layers  $L_r$ , the number of tokens to prune is

$$P^\ell = \text{round} \left( \frac{N^\ell - K}{\sqrt{L_r}} (1 - \rho_H^2) \right). \quad (18)$$

The retained token set is selected by top- $K^\ell$  cumulative scores:

$$\mathcal{K}^\ell = \text{TopK}(\mathbf{R}^\ell, N^\ell - P^\ell). \quad (19)$$

The selected indices are sorted to preserve the original spatial order.

### 3.7 Semantic Packing of Pruned Tokens

After token selection, SPack performs semantic packing to inject information from pruned tokens into retained ones. Let  $\mathbf{X}_\mathcal{K}^\ell$  and  $\mathbf{X}_\mathcal{P}^\ell$  denote retained and pruned patch tokens. We first compute the average attention mass of pruned tokens:

$$m_\mathcal{P}^\ell = \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} a_i^\ell. \quad (20)$$

Then we compute a soft assignment over retained tokens:

$$\omega_j^\ell = \frac{\exp(\text{Mean}(\mathbf{x}_j^\ell))}{\sum_{k \in \mathcal{K}} \exp(\text{Mean}(\mathbf{x}_k^\ell))}, \quad j \in \mathcal{K}. \quad (21)$$

The retained token is updated by

$$\tilde{\mathbf{x}}_j^\ell = \mathbf{x}_j^\ell \left( 1 + \alpha m_\mathcal{P}^\ell \omega_j^\ell \right), \quad (22)$$

where  $\alpha$  is a small transfer coefficient. This multiplicative update increases the information density of retained tokens while keeping their semantic direction stable.

We also inject a compact summary of pruned tokens into the CLS token:

$$\bar{\mathbf{x}}_\mathcal{P}^\ell = \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} \mathbf{x}_i^\ell, \quad (23)$$

$$\tilde{\mathbf{x}}_{\text{cls}}^\ell = \mathbf{x}_{\text{cls}}^\ell \left( 1 + \beta \sigma(\|\bar{\mathbf{x}}_\mathcal{P}^\ell\|_2) \right), \quad (24)$$

where  $\beta$  controls the CLS enhancement strength. Since the CLS token serves as a global visual summary, this update helps preserve information that is difficult to assign to a single retained patch.

### 3.8 Spatial Order Preservation for High-Resolution Inputs

For high-resolution inputs, VLMs such as LLaVA-NeXT split an image into multiple local patches and later merge them according to their spatial layout. Direct token pruning may disturb this layout. SPack therefore tracks the original indices of retained tokens throughout pruning. After compression, the retained features are restored to their original grid positions, and invalid or padded regions are removed according to the image aspect ratio. This step preserves compatibility with spatial patch merging and prevents the language model from receiving disordered visual evidence.

Formally, for retained indices  $\mathcal{K}$ , we construct a binary mask  $\mathbf{M} \in \{0, 1\}^{H \times W}$  over the original visual grid:

$$M_i = \begin{cases} 1, & i \in \mathcal{K}, \\ 0, & \text{otherwise.} \end{cases} \quad (25)$$

The compressed features are then reordered by scanning the valid grid positions. This ensures that SPack reduces token count without destroying the spatial prior expected by the multimodal projector.

## 4 Experiments

**Experimental Setup.** To validate the effectiveness of SPack, we evaluate it on several public LLMs, including LLaVA-1.5-7B, its high-resolution variant LLaVA-NeXT-7B (Liu et al., 2023, 2024a), and the more advanced Qwen2-VL model (Wang et al., 2024) for image-based reasoning tasks, demonstrating the generality of our approach. These models are evaluated across multiple benchmarks under various token reduction ratios. The datasets we use include GQA (Hudson and Manning, 2019), MMBench (Liu et al., 2024b), MME (Fu et al., 2026), POPE (Li et al., 2023), ScienceQA (Lu et al., 2022), VQAv2 (Goyal et al., 2017), TextVQA (Singh et al., 2019), and SEEDBench (Li et al., 2024).

**Implementation Details.** Our technique can be readily incorporated into existing models in a training-free manner, consistently delivering improved inference speed. For LLaVA-1.5, token pruning is performed at the penultimate layer of the vision encoder. For LLaVA-NeXT and Qwen2-VL, which generate a substantially larger number of tokens, pruning is applied from the vision encoder up to the second layer of the VLM. In these

Table 1: **Performance of SPack on LLaVA-1.5-7B under different visual token settings.** The vanilla number of visual tokens is 576. The last column is the average accuracy proportion relative to the upper limit.

| Method                                | GQA         | MMB         | MME         | POPE        | SQA         | VizWiz      | SEED        | Avg.          |
|---------------------------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|---------------|
| <i>Upper Bound, 576 Tokens (100%)</i> |             |             |             |             |             |             |             |               |
| Vanilla                               | 61.9        | 64.7        | 1862        | 85.9        | 69.5        | 50.0        | 59.3        | 100.0%        |
| <i>Retain 192 Tokens (↓ 66.7%)</i>    |             |             |             |             |             |             |             |               |
| FastV (ECCV24)                        | 52.7        | 61.2        | 1612        | 64.8        | 67.3        | 50.8        | 57.1        | 90.9%         |
| SparseVLM (ICML25)                    | 57.6        | 62.5        | 1721        | 83.6        | 68.9        | 50.5        | 55.8        | 96.2%         |
| PDrop (CVPR25)                        | 57.3        | 62.9        | 1766        | 82.3        | 68.8        | 51.1        | 54.7        | 96.3%         |
| VisionZip (CVPR25)                    | 59.3        | 63.0        | 1778        | 85.2        | 68.7        | 51.2        | 56.4        | 97.7%         |
| FlowCut (NeurIPS25)                   | 59.1        | 62.9        | 1821        | 85.7        | 68.3        | 51.5        | 57.0        | 98.2%         |
| V2Drop (CVPR26)                       | 58.5        | <b>63.7</b> | 1826        | 85.1        | 69.3        | 52.4        | 57.2        | 98.7%         |
| <b>SPack (Ours)</b>                   | <b>60.2</b> | <b>63.0</b> | <b>1835</b> | <b>86.8</b> | <b>69.7</b> | <b>54.5</b> | <b>58.8</b> | <b>100.4%</b> |
| <i>Retain 128 Tokens (↓ 77.8%)</i>    |             |             |             |             |             |             |             |               |
| FastV (ECCV24)                        | 49.6        | 56.1        | 1490        | 59.6        | 60.2        | 51.3        | 55.9        | 85.7%         |
| SparseVLM (ICML25)                    | 56.0        | 60.0        | 1696        | 80.5        | 67.1        | 51.4        | 53.4        | 93.9%         |
| PDrop (CVPR25)                        | 57.1        | 61.6        | 1664        | 82.3        | 68.3        | 51.0        | 53.3        | 94.7%         |
| VisionZip (CVPR25)                    | 57.6        | 62.0        | 1759        | 83.2        | 68.9        | 51.6        | 54.9        | 96.4%         |
| FlowCut (NeurIPS25)                   | 57.5        | 61.8        | 1778        | 84.8        | 68.3        | 51.9        | 55.7        | 96.9%         |
| V2Drop (CVPR26)                       | 56.3        | 61.8        | 1712        | 80.9        | 68.8        | 51.8        | 54.8        | 95.4%         |
| <b>SPack (Ours)</b>                   | <b>58.5</b> | <b>62.1</b> | <b>1781</b> | <b>85.7</b> | <b>69.7</b> | <b>54.8</b> | <b>57.5</b> | <b>98.9%</b>  |
| <i>Retain 64 Tokens (↓ 88.9%)</i>     |             |             |             |             |             |             |             |               |
| FastV (ECCV24)                        | 46.1        | 48.0        | 1256        | 48.0        | 51.1        | 50.8        | 51.9        | 76.4%         |
| SparseVLM (ICML25)                    | 52.7        | 56.2        | 1505        | 75.1        | 62.2        | 50.1        | 51.1        | 88.0%         |
| PDrop (CVPR25)                        | 47.5        | 58.8        | 1561        | 55.9        | 68.6        | 50.7        | 40.0        | 83.4%         |
| VisionZip (CVPR25)                    | 55.1        | 60.1        | 1687        | 77.0        | 69.0        | 52.8        | 52.2        | 93.6%         |
| FlowCut (NeurIPS25)                   | 54.6        | <b>60.4</b> | <b>1728</b> | 79.7        | 68.8        | 52.9        | 53.0        | 94.5%         |
| V2Drop (CVPR26)                       | 50.5        | 55.2        | 1470        | 75.1        | 68.9        | 51.0        | 51.5        | 88.8%         |
| <b>SPack (Ours)</b>                   | <b>55.6</b> | <b>60.2</b> | <b>1718</b> | <b>82.3</b> | <b>70.1</b> | <b>55.6</b> | <b>54.2</b> | <b>96.4%</b>  |

models, we first prune tokens to twice the desired count within the vision encoder, then further reduce to the final target number at the second LLM layer. All experiments are carried out on NVIDIA A6000-48G GPUs.

### 4.1 Performance on Diverse Tasks

We evaluate the effectiveness of our proposed SPack across multiple benchmarks, comparing it against state-of-the-art token pruning methods including FastV (Chen et al., 2024a), SparseVLM (Zhang et al., 2024b), PDrop (Xing et al., 2024), VisionZip (Yang et al., 2025), FlowCut (Tong et al., 2026), and V2Drop (Chen et al., 2025a). All experiments are conducted on LLaVA-1.5-7B and the high-resolution LLaVA-NeXT-7B under various token reduction ratios.

**Results on LLaVA-1.5-7B.** Table 1 reports the main results on LLaVA-1.5-7B under different visual token budgets. SPack consistently achieves the best average performance across all settings,

Table 2: **Comparison on LLaVA-NeXT-7B under different visual token settings.** The original number of visual tokens is 2880. The last column is the average accuracy proportion relative to the upper bound.

| Method                                 | GQA         | MMB         | MME         | POPE        | VQA <sup>V2</sup> | VQA <sup>Text</sup> | SEED        | Avg.         |
|----------------------------------------|-------------|-------------|-------------|-------------|-------------------|---------------------|-------------|--------------|
| <i>Upper Bound, 2880 Tokens (100%)</i> |             |             |             |             |                   |                     |             |              |
| Vanilla                                | 64.2        | 67.9        | 1846        | 86.4        | 81.8              | 61.3                | 63.6        | 100.0%       |
| <i>Retain 640 Tokens (↓ 77.8%)</i>     |             |             |             |             |                   |                     |             |              |
| SparseVLM                              | 60.3        | 65.8        | 1773        | 84.2        | 77.1              | 57.8                | 59.3        | 95.2%        |
| PDrop                                  | 60.6        | 65.5        | 1781        | 83.7        | 78.3              | 57.4                | 60.5        | 95.5%        |
| VisionZip                              | 61.3        | 66.2        | 1787        | 85.9        | 79.1              | 60.2                | 60.9        | 97.1%        |
| FlowCut                                | 61.9        | 66.7        | <b>1837</b> | 86.1        | <b>79.8</b>       | 60.6                | <b>61.9</b> | 98.2%        |
| <b>SPack (Ours)</b>                    | <b>64.1</b> | <b>67.4</b> | 1763        | <b>88.2</b> | 79.7              | <b>64.0</b>         | 57.5        | <b>98.4%</b> |
| <i>Retain 320 Tokens (↓ 88.9%)</i>     |             |             |             |             |                   |                     |             |              |
| SparseVLM                              | 57.7        | 63.2        | 1685        | 82.2        | 73.4              | 55.9                | 56.9        | 91.4%        |
| PDrop                                  | 58.3        | 63.9        | 1736        | 80.2        | 75.2              | 55.3                | 57.8        | 92.1%        |
| VisionZip                              | 59.2        | 63.1        | 1702        | 82.1        | 76.2              | 58.9                | 58.2        | 93.3%        |
| FlowCut                                | 59.8        | 65.3        | <b>1791</b> | 83.4        | 77.8              | 60.1                | 59.5        | 95.7%        |
| <b>SPack (Ours)</b>                    | <b>63.1</b> | <b>66.7</b> | 1774        | <b>88.1</b> | <b>78.5</b>       | <b>62.5</b>         | <b>61.7</b> | <b>98.5%</b> |
| <i>Retain 160 Tokens (↓ 94.4%)</i>     |             |             |             |             |                   |                     |             |              |
| SparseVLM                              | 51.2        | 52.1        | 1542        | 72.7        | 66.3              | 46.4                | 49.2        | 79.8%        |
| PDrop                                  | 54.9        | 61.8        | 1513        | 72.3        | 70.2              | 52.7                | 53.9        | 85.5%        |
| VisionZip                              | 55.5        | 60.1        | 1628        | 74.8        | 71.4              | 56.2                | 54.2        | 87.7%        |
| FlowCut                                | 57.6        | 62.8        | 1746        | 79.9        | 74.6              | 57.6                | 56.8        | 92.0%        |
| <b>SPack (Ours)</b>                    | <b>61.5</b> | <b>64.4</b> | <b>1762</b> | <b>86.8</b> | <b>76.5</b>       | <b>58.8</b>         | <b>62.3</b> | <b>96.3%</b> |

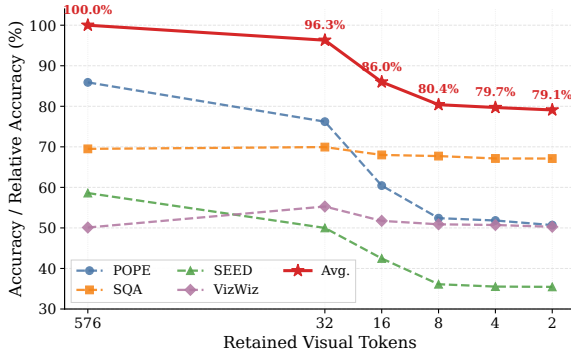


Figure 3: Performance under extremely low visual token budgets.

retaining 100.4%, 98.9%, and 96.4% of the upper-bound performance when keeping 192, 128, and 64 tokens, respectively. SPack remains highly robust under aggressive compression. With only 64 tokens, it outperforms FlowCut by 1.9 percentage points on average and achieves the best results on five of seven benchmarks, including GQA, POPE, SQA, VizWiz, and SEED. These results show that SPack preserves task-relevant visual information while removing redundant tokens, leading to a better accuracy than existing methods.

**Results on LLaVA-NeXT-7B.** Table 2 presents results on the high-resolution LLaVA-NeXT-7B, which processes up to 2,880 visual tokens. Under the 640-token setting with 77.8% reduction, SPack

Table 3: **Comparison on Qwen2-VL-7B under different visual token reduction settings.** The original number of visual tokens is 576. The last column is the average accuracy proportion relative to the upper bound.

| Method                                | GQA         | MMB         | MMB <sup>CN</sup> | MME         | POPE        | SQA         | Avg.         |
|---------------------------------------|-------------|-------------|-------------------|-------------|-------------|-------------|--------------|
| <i>Upper Bound, All Tokens (100%)</i> |             |             |                   |             |             |             |              |
| Vanilla                               | 61.9        | 79.9        | 79.5              | 2338        | 87.2        | 85.1        | 100.0%       |
| <i>Token Reduction (↓ 66.7%)</i>      |             |             |                   |             |             |             |              |
| FastV                                 | 58.0        | 76.1        | 73.9              | 2130        | 82.1        | 80.0        | 93.5%        |
| V2Drop                                | 59.0        | 78.7        | 77.0              | 2224        | 87.2        | 81.6        | 97.0%        |
| FlowCut                               | 60.5        | 79.2        | 78.2              | 2335        | 86.0        | 84.0        | 98.7%        |
| <b>SPack (Ours)</b>                   | <b>61.0</b> | <b>78.4</b> | <b>77.3</b>       | <b>2313</b> | <b>87.9</b> | <b>85.0</b> | <b>98.9%</b> |
| <i>Token Reduction (↓ 77.8%)</i>      |             |             |                   |             |             |             |              |
| FastV                                 | 56.7        | 74.1        | 72.4              | 2031        | 79.2        | 78.3        | 90.9%        |
| V2Drop                                | 57.8        | 75.8        | 74.5              | 2173        | 85.1        | 78.9        | 95.1%        |
| FlowCut                               | 59.2        | 77.8        | 76.9              | 2310        | 84.6        | 80.5        | 96.7%        |
| <b>SPack (Ours)</b>                   | <b>59.7</b> | <b>78.5</b> | <b>77.0</b>       | <b>2271</b> | <b>86.6</b> | <b>84.5</b> | <b>97.9%</b> |
| <i>Token Reduction (↓ 88.9%)</i>      |             |             |                   |             |             |             |              |
| FastV                                 | 51.9        | 70.1        | 63.8              | 1962        | 76.1        | 75.8        | 85.3%        |
| V2Drop                                | 53.5        | 72.0        | 70.5              | 2080        | 80.5        | 74.5        | 89.2%        |
| FlowCut                               | 56.4        | 72.6        | 72.5              | 2252        | 81.8        | 78.2        | 92.5%        |
| <b>SPack (Ours)</b>                   | <b>57.1</b> | <b>74.7</b> | <b>75.2</b>       | <b>2168</b> | <b>83.8</b> | <b>82.7</b> | <b>94.4%</b> |

achieves 98.4% relative accuracy, outperforming FlowCut at 98.2% and VisionZip at 97.1%. Under the 320-token setting with 88.9% reduction, SPack reaches 98.5% relative accuracy, significantly surpassing FlowCut at 95.7% and VisionZip at 93.3%. Even under the most challenging 160-token setting with 94.4% reduction, our method retains 96.3% relative accuracy, substantially outperforming FlowCut at 92.0%, VisionZip at 87.7%, and PDrop at 85.5%. Notably, SPack achieves or exceeds the vanilla upper bound on several benchmarks including POPE and VQA<sup>Text</sup> under moderate reduction ratios.

**Results on Qwen2-VL-7B.** Table 3 evaluates SPack on Qwen2-VL-7B under different token reduction ratios. SPack consistently achieves the best average performance, retaining 98.9%, 97.9%, and 94.4% of the upper-bound accuracy under 66.7%, 77.8%, and 88.9% token reduction, respectively. The advantage becomes more evident under aggressive compression: when reducing 88.9% of visual tokens, SPack surpasses FlowCut by 1.9 percentage points on average and obtains the best results on five of six benchmarks. These results demonstrate that SPack generalizes well beyond the LLaVA family and remains robust on stronger LVLMS.

**Extremely Aggressive Compression** To further examine the robustness of SPack under highly con-

Table 4: **Efficiency comparison on MME and POPE.** We report the total inference time, prefill time, peak GPU memory, KV cache memory, and throughput. The original setting uses all visual tokens (576), while SPack keeps only 64 visual tokens.

| Metric                 | MME (2374 items) |         |                                | POPE (9000 items) |         |                                |
|------------------------|------------------|---------|--------------------------------|-------------------|---------|--------------------------------|
|                        | Vanilla          | FlowCut | Ours ( $\Delta$ )              | Vanilla           | FlowCut | Ours ( $\Delta$ )              |
| Inference Time (s) ↓   | 356.16           | 230.00  | <b>201.95</b> (1.76 $\times$ ) | 1206.30           | 810.00  | <b>726.77</b> (1.66 $\times$ ) |
| Prefill Time (s) ↓     | 259.30           | 100.00  | <b>81.40</b> (3.19 $\times$ )  | 844.29            | 330.00  | <b>268.55</b> (3.14 $\times$ ) |
| GPU Peak Memory (GB) ↓ | 13.96            | 13.65   | <b>13.43</b> (1.04 $\times$ )  | 13.93             | 13.62   | <b>13.42</b> (1.04 $\times$ )  |
| KV Cache Memory (MB) ↓ | 331.50           | 120.00  | <b>75.50</b> (4.39 $\times$ )  | 320.00            | 110.00  | <b>64.00</b> (5.00 $\times$ )  |
| Throughput (items/s) ↑ | 6.67             | 9.80    | <b>11.76</b> (1.76 $\times$ )  | 7.46              | 9.80    | <b>12.38</b> (1.66 $\times$ )  |

strained visual token budgets, we evaluate its performance when retaining only 32, 16, 8, 4, and even 2 visual tokens. As shown in Figure 3, SPack preserves strong performance with only 32 tokens, achieving 96.3% of the full-token upper bound on average across POPE, SQA, SEED, and VizWiz. Notably, SPack matches or exceeds the vanilla model on SQA (70.0%) and VizWiz (56.0% vs. 50.0%, +6.0%) under the 32-token setting. As the token budget becomes more aggressive, performance gradually degrades, yet SPack still maintains 79.1% of the upper-bound performance with only 2 visual tokens. These results demonstrate that SPack can effectively concentrate visual information into a compact token subset and remains robust even under extreme compression ratios.

## 4.2 Efficiency Analysis

Table 4 shows that SPack consistently improves inference efficiency over both the vanilla setting with 576 visual tokens and the prior FlowCut method. On MME and POPE, SPack achieves 1.76 $\times$  and 1.66 $\times$  end-to-end speedups, respectively. The gain is more significant in the prefilling stage, where SPack reaches over 3 $\times$  acceleration on both datasets, demonstrating the benefit of reducing redundant visual tokens. SPack also substantially reduces KV-cache memory by 4.39 $\times$  on MME and 5.00 $\times$  on POPE, while improving throughput to 11.76 and 12.38 items/s. These results indicate that SPack effectively lowers both computational and memory costs during multimodal inference.

## 4.3 Ablation Studies

Table 5 ablates the key components of SPack. Hard pruning causes severe information loss and performs worst. Replacing it with Packaging brings clear gains, confirming the benefit of retaining semantic mass from pruned tokens. Random and local neighbor injection yield progressive improve-

Table 5: **Ablation study on multimodal benchmarks.** We compare different variants of our method: Baseline (Hard Pruning), Semantic Packaging only (Simple Average Pooling), Packaging with Random Injection, and our full method with Optimal Transport.

| Configuration          | GQA          | MMB          | POPE         | SQA          | VizWiz       | SEED         |
|------------------------|--------------|--------------|--------------|--------------|--------------|--------------|
| 0. Baseline            | 52.10        | 57.30        | 78.50        | 66.80        | 53.20        | 51.60        |
| 1. Packaging Only      | 53.50        | 58.60        | 79.90        | 67.90        | 54.10        | 52.80        |
| 2. + Random Injection  | 54.20        | 59.10        | 80.80        | 68.60        | 54.50        | 53.40        |
| 3. + Local Neighbor    | 55.00        | 59.70        | 81.60        | 69.30        | 55.00        | 53.90        |
| <b>4. SPack (Ours)</b> | <b>55.60</b> | <b>60.22</b> | <b>82.34</b> | <b>70.08</b> | <b>55.60</b> | <b>54.24</b> |

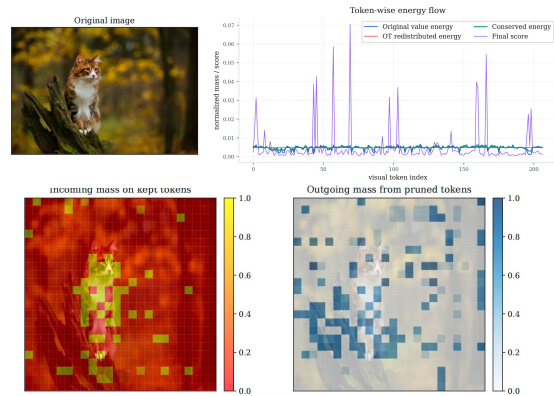


Figure 4: Spatial distribution of retained visual tokens.

ments. Our full SPack with optimal transport achieves the best results, improving GQA by 6.21 over baseline. These results validate that formulating token compression as information-energy redistribution effectively preserves both local and global visual semantics.

## 4.4 Visualization

Figure 4 visualizes the proposed OT-based token pruning and energy redistribution mechanism in vision Transformers. The top-left panel shows the input cat image, while the top-right plot compares original value energy, conserved energy, OT-redistributed energy, and final importance scores. The results show that OT redistribution amplifies semantically critical tokens and suppresses redundant background tokens. The bottom row further illustrates token-level energy flow: incoming mass is concentrated on kept foreground tokens around the cat, while outgoing mass mainly comes from low-information background tokens. The visualization shows that SPack can retain semantically important tokens while removing redundant ones through energy-conserving redistribution.

## 5 Conclusion

In this paper, we propose **SPack**, a training-free vision token compression method that rethinks hard pruning. Instead of directly discarding redundant tokens, SPack treats pruned tokens as semantic mass and redistributes their information to retained tokens via optimal transport. By jointly modeling relation attention, semantic affinity, and value-space energy, SPack compresses visual tokens while preserving contextual information. Extensive experiments show superior accuracy-efficiency trade-offs, indicating that semantic information conservation is crucial for token compression.

## Limitations

Although SPack achieves a favorable accuracy-efficiency trade-off, the compression process still inevitably introduces a certain degree of information loss compared with using full vision tokens, especially under aggressive pruning ratios. This may slightly affect performance on tasks requiring fine-grained visual details. In addition, SPack is designed as a training-free framework and mainly evaluates token redistribution based on feature similarity and importance estimation. Its effectiveness may therefore depend on the quality of intermediate representations produced by the backbone VLM, which could limit generalization across architectures with significantly different token distributions.

## Acknowledgements

We gratefully acknowledge the support of the National Key R&D Program of China (No. 2024YFC2707805), the National Natural Science Foundation of China (No. 62506050, No. 62176029), China Postdoctoral Science Foundation Funded Project (2024M763867), the Chongqing Science and Technology Innovation and Application Development projects (No. CSTB2023TIAD-KPX0064, No. CSTB2025TIAD-JM010), and the Open Competition Program of Chongqing Municipal Commission of Economy and Information Technology (No. YJX-202500100200X).

The experimental and computational work in this research runs on the Huawei Cloud AI Compute Service. We appreciate the stable compute supply from this platform.

## References

- Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025. [Qwen2.5-vl technical report](#). *Preprint*, arXiv:2502.13923.
- Junjie Chen, Xuyang Liu, Zichen Wen, Yiyu Wang, Siteng Huang, and Honggang Chen. 2025a. Variation-aware vision token dropping for faster large vision-language models. *arXiv preprint arXiv:2509.01552*.
- 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 *European Conference on Computer Vision*, pages 19–35. Springer.
- Yi Chen, Jian Xu, Xu-Yao Zhang, Wen-Zhuo Liu, Yang-Yang Liu, and Cheng-Lin Liu. 2025b. Recoverable compression: A multimodal vision token recovery mechanism guided by text information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 2293–2301.
- 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. *Science China Information Sciences*, 67(12):220101.
- Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, and 1 others. 2024c. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 24185–24198.
- Marco Cuturi. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26.
- Wenliang Dai, Junnan Li, Dongxu Li, Anthony Tjong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. 2023. Instructblip: Towards general-purpose vision-language models with instruction tuning. *Advances in neural information processing systems*, 36:49250–49267.
- Sixun Dong, Juhua Hu, Mian Zhang, Ming Yin, Yanjie Fu, and Qi Qian. 2026. [MMTok: Multimodal coverage maximization for efficient inference of VLMs](#). In *The Fourteenth International Conference on Learning Representations*.
- Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiwu Zheng, Ke Li, Xing Sun, and 1 others. 2026. Mme: A comprehensive evaluation benchmark for multimodal

- large language models. *Advances in Neural Information Processing Systems*, 38.
- 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913.
- Jitai Hao, Yuke Zhu, Tian Wang, Jun Yu, Xin Xin, Bo Zheng, Zhaochun Ren, and Sheng Guo. 2025. Omniv: Dynamic context selection for efficient long-context llms. In *International Conference on Learning Representations*, volume 2025, pages 87443–87464.
- Drew A Hudson and Christopher D Manning. 2019. 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.
- Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. 2024. Seed-bench: Benchmarking multimodal large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13299–13308.
- Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. 2023. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 292–305.
- Yucheng Li, Huiqiang Jiang, Qianhui Wu, Xufang Luo, Surin Ahn, Chengruidong Zhang, Amir Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, and 1 others. 2025a. Scbench: A kv cache-centric analysis of long-context methods. In *International Conference on Learning Representations*, volume 2025, pages 66063–66093.
- Zongxia Li, Xiyang Wu, Hongyang Du, Fuxiao Liu, Huy Nghiem, and Guangyao Shi. 2025b. A survey of state of the art large vision language models: Alignment, benchmark, evaluations and challenges. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1578–1597. IEEE.
- Zhihang Lin, Mingbao Lin, Luxi Lin, and Rongrong Ji. 2025. Boosting multimodal large language models with visual tokens withdrawal for rapid inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 5334–5342.
- Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. 2024a. Llava-next: Improved reasoning, ocr, and world knowledge.
- Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916.
- Jiaxin Liu, Ding Zhong, Yue Wang, Zhidong Yang, Zhaolu Kang, Guangyuan Dong, Qishi Zhan, Pengcheng Fang, and Aofan Liu. 2026. Dual-pathway circuits of object hallucination in vision-language models. *arXiv preprint arXiv:2605.13156*.
- Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, and 1 others. 2024b. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pages 216–233. Springer.
- Pan Lu, Swaroop Mishra, Tanglin 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. *Advances in neural information processing systems*, 35:2507–2521.
- Fanqing Meng, Jin Wang, Chuanhao Li, Quanfeng Lu, Hao Tian, Tianshuo Yang, Jiaqi Liao, Xizhou Zhu, Jifeng Dai, Yu Qiao, and 1 others. 2025. Mmiu: Multimodal multi-image understanding for evaluating large vision-language models. In *International Conference on Learning Representations*, volume 2025, pages 38405–38453.
- Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. 2025. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22857–22867.
- 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8317–8326.
- Liyan Tang, Grace Kim, Xinyu Zhao, Thom Lake, Wenxuan Ding, Fangcong Yin, Prasann Singhal, Many Wadhwa, Zeyu Liu, Zayne Sprague, and 1 others. 2026. Chartmuseum: Testing visual reasoning capabilities of large vision-language models. *Advances in Neural Information Processing Systems*, 38.
- Jintao Tong, Wenwei Jin, Pengda Qin, Anqi Li, Yixiong Zou, Yuhong Li, Yuhua Li, and Ruixuan Li. 2026. Flowcut: Rethinking redundancy via information flow for efficient vision-language models. *Advances in Neural Information Processing Systems*, 38:94946–94973.
- Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai C Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, and 1 others. 2024. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *Advances in Neural Information Processing Systems*, 37:87310–87356.

- Cédric Villani and 1 others. 2009. *Optimal transport: old and new*, volume 338. Springer.
- Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, and 1 others. 2024. Qwen2-vl: Enhancing vision-language model’s perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.
- Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, and 1 others. 2024. Pyramidrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. *arXiv preprint arXiv:2410.17247*.
- Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, and 1 others. 2025. Conical visual concentration for efficient large vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14593–14603.
- Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. 2025. Visionzip: Longer is better but not necessary in vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19792–19802.
- Qizhe Zhang, Aosong Cheng, Ming Lu, Zhiyong Zhuo, Minqi Wang, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. 2024a. [cls] attention is all you need for training-free visual token pruning: Make vlm inference faster. *arXiv e-prints*, pages arXiv–2412.
- Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and 1 others. 2024b. Sparsevlm: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*.
- Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, and 1 others. 2025. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*.
- Jiedong Zhuang, Lu Lu, Ming Dai, Rui Hu, Jian Chen, Qiang Liu, and Haoji Hu. 2025. St3: Accelerating multimodal large language model by spatial-temporal visual token trimming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 11049–11057.