
SparseVLM: Visual Token Sparsification for Efficient Vision-Language Model Inference

Yuan Zhang^{*1} Chun-Kai Fan^{*1} Junpeng Ma^{*2} Wenzhao Zheng³ Tao Huang⁴ Kuan Cheng¹
Denis Gudovskiy⁵ Tomoyuki Okuno⁵ Yohei Nakata⁵ Kurt Keutzer³ Shanghang Zhang¹

Abstract

In vision-language models (VLMs), visual tokens usually bear a significant amount of computational overhead despite sparsity of information in them when compared to text tokens. To address this, most existing methods learn a network to prune redundant visual tokens using certain training data. Differently, we propose a text-guided training-free token optimization mechanism dubbed SparseVLM that eliminates the need of extra parameters or fine-tuning costs. Given that visual tokens complement text tokens in VLM’s linguistic reasoning, we select relevant text tokens to rate the significance of visual tokens using self-attention matrices and, then, prune visual tokens using the proposed strategy to maximize sparsity while retaining information. In particular, we introduce a rank-based strategy to adaptively determine the sparsification ratio for each layer, alongside a token recycling method that compresses pruned tokens into more compact representations. Experimental results show that SparseVLM increases the efficiency of various VLMs in a number of image and video understanding tasks. Our code is available at <https://github.com/Gumpest/SparseVLMs>.

1. Introduction

Benefiting from advancements in large language models (LLMs) (Radford et al., 2019; Brown et al., 2020; Touvron et al., 2023; Peng et al., 2023; Zhang et al., 2024a), the realm of vision-language models (VLMs) has undergone

^{*}Equal contribution ¹State Key Laboratory of Multimedia Information Processing, School of Computer Science, Peking University ²Fudan University ³EECS, UC Berkeley ⁴Shanghai Jiao Tong University ⁵Panasonic Holdings Corporation. Correspondence to: Wenzhao Zheng <wzzheng@berkeley.edu>, Shanghang Zhang <shanghang@pku.edu.cn>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

significant progress. To combine visual signals with textual semantics, the mainstream practice in VLMs (Team et al., 2023; Bai et al., 2023; Chen et al., 2024b; Li et al., 2024c; 2023a) employs sequential visual representation, where images are extracted into visual tokens and sent into an LLM decoder. With modal alignment and instruction fine-tuning (Du et al., 2022; Liu et al., 2024a; Zhu et al., 2024b), recent VLMs successfully adapt LLMs to the vision domain and inherit their perception and reasoning abilities.

Despite the promising performance, further incorporation of visual tokens inevitably introduces a huge memory and computational overhead when compared to LLMs, particularly for high-resolution images (Li et al., 2024c) and long videos (Lin et al., 2024). For instance, a 672×672 image in LLaVA (Liu et al., 2024b) yields 2304 visual tokens that span over half of the context length. However, the information in images is typically more sparse than in natural languages (Marr, 2010), resulting in inefficiency when naively processing both modalities. To address this, existing methods extract more compact image representations by modifying the image encoder or projector (Alayrac et al., 2022; Li et al., 2024b; Dai et al., 2023; Cha et al., 2024). While some recent works further sparsify visual tokens during the decoding (Ye et al., 2025; Chen et al., 2024a; Shang et al., 2024), they still ignore the guidance from the language tokens, which contradicts the multimodality paradigm. We argue that ***visual tokens should be sparsified adaptively based on the question prompt***, as the model might focus on different parts (e.g., foreground or background) when dealing with various questions as shown in Figure 1. Furthermore, current approaches generally train a network to prune redundant visual tokens and require additional training data (Li et al., 2024b; Ye et al., 2025; Cai et al., 2025).

In this paper, we introduce a text-guided training-free framework dubbed **SparseVLM** for efficient vision language model inference. We reuse the self-attention matrix of visual-text tokens directly from the decoder layers without extra training parameters for sparsification. We ascertain that ***not all prompt tokens should be considered*** as some could be less relevant, which leads to inaccurate correlation results and downgrades the performance of sparse

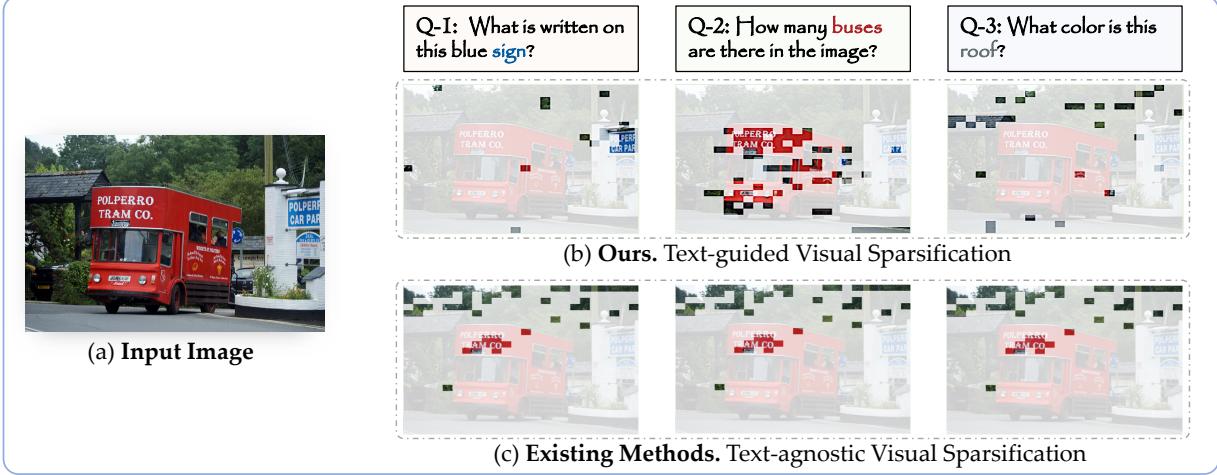


Figure 1. Comparison of visual token sparsification methods. Unlike previous methods with text-agnostic visual sparsification (c) e.g., VocoLLaMA (Ye et al., 2025), our SparseVLM (b) is guided by question prompts to select relevant visual patches from the image (a).

inference. Specifically, our SparseVLM first identifies text tokens strongly correlated with visual signals via cross-attention. Then, we measure the contribution of visual tokens to the selected visual-relevant text tokens (i.e., “*raters*”) and adaptively prune the insignificant visual tokens. Instead of directly discarding the pruned tokens, we further recycle and cluster them to reconstruct more compact tokens to minimize the loss of information. Due to the information density varying for different image inputs, we employ the rank of the attention matrix to indicate the redundancy level and set an adaptive sparsification ratio accordingly.

The proposed method is simple yet practical. It can act as a plug-and-play module to improve the efficiency of VLMs without additional fine-tuning. Extensive experiments demonstrate that our SparseVLM effectively reduces computational overhead of various VLMs without sacrificing their performance in a wide range of image and video understanding tasks. For instance, LLaVA (Liu et al., 2024b) when armed with SparseVLM achieves a $4.5\times$ compression rate while maintaining 97% of its original performance. Alternatively, the CUDA latency can decrease by 37% with only a 0.9% drop in accuracy. To investigate the effectiveness of our method in video tasks, we further apply SparseVLM to VideoLLaVA (Lin et al., 2024) to compress frames with temporal dimension. Without complex design changes, SparseVLM can sparsify video frames into an adaptive number of visual tokens and outperform existing methods in video question-answering benchmarks. Our approach consistently outperforms prior state-of-the-art FastV method (Chen et al., 2024a) by 11.2 – 17.3% on LLaVA, 9.2 – 20.4% on MiniGemini, and 14.7% on VideoLLaVA when both have similar latencies.

Our main contributions are summarized as follows:

- We introduce a novel sparsification framework dubbed SparseVLM. To the best of our knowledge, it is the first training-free approach that explores text-aware guidance for efficient VLM inference.
- Particularly, we propose a strategy to select relevant text tokens as raters of visual tokens, a method to assess the significance of visual tokens followed by pruning of redundant visual tokens with a recycling mechanism to minimize the loss of information.
- When applied to a number of VLMs, SparseVLM consistently outperforms prior state-of-the-art methods in various image and video understanding benchmarks.

2. Related Work

Vision-Language Models. Recent works on vision-language models (Liu et al., 2024a; Chen et al., 2024b; Li et al., 2024c) improve multimodal comprehension and generation by processing longer visual token sequences. Moreover, the usage of higher-resolution images inevitably entails an exponential growth in the length of visual sequences. For example, LLaVA typically encodes 336×336 images into 576 tokens (Liu et al., 2024b) with up to 672×672 maximum resolution using 2880 token sequences (Liu et al., 2024a). Similarly, mini-Gemini-HD (Li et al., 2024c) converts 1536×1536 high resolution and 672×672 low resolution images into 2880 visual tokens. Moreover, comprehending videos or multiple images leads to increased token allocations for visual signals. For instance, the VideoLLaVA (Lin et al., 2024) and VideoPoet (Kondratyuk et al., 2024) use thousands of tokens to encode multiple image frames. However, large number of visual tokens results in a computational bottleneck. Further research on sparsification is

urged to further unleash VLM capabilities.

Visual Compression for VLMs. Compression of visual tokens is necessary because, on the one hand, their quantity is usually tens to hundreds of times that of language tokens. On the other hand, visual signals are inherently more sparse in information when compared to texts that have been produced by humans (Marr, 2010). Past efforts to address the above problem can be categorized into two directions. The first one centers on the compression of a vision tower or an efficient projection of vision modality. For instance, LLaMA-VID (Li et al., 2024b) exploits the Q-Former with the context token while DeCo (Yao et al., 2024) employs an adaptive pooling to downsample the visual tokens at the patch level. Methods that belong to the second direction (Ye et al., 2025; Chen et al., 2024a; Wu et al., 2024) go deeper into the text modality and sparsify visual tokens during the LLM decoding stage, but they still lack guidance from the text tokens. In this paper, SparseVLM takes note of this limitation and improves performance upon it.

3. Method

In this section, we present our SparseVLM for efficient VLM inference. We first review the attention mechanism in VLMs and then introduce the detailed strategies for our visual sparsification including visual significance estimation, relevant text token selection, and sparsification level adaptation. We further propose token recycling to reduce information loss and provide a theoretical analysis of computation savings. The pipeline is shown in Figure 2.

3.1. Preliminary: Attention in VLM Decoders

VLM decoders typically rely on the causal *self-attention* from the original transformer architecture (Vaswani et al., 2017) for token interactions. Without loss of generality, we describe the single-head attention below. Formally, the self-attention matrix with logits $\mathbf{A} \in \mathbb{R}^{L \times L}$, where L denotes the length of a sequence with all kinds of tokens e.g. text and visual, is computed by

$$\mathbf{A} = \text{Attention}(\mathbf{Q}, \mathbf{K}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{D}}\right), \quad (1)$$

where the scalar D represents the matrix dimension, and the $\mathbf{Q} \in \mathbb{R}^{L \times D}$ and $\mathbf{K} \in \mathbb{R}^{L \times D}$ are the query and key matrices, respectively. The keys and queries in a self-attention layer are computed in parallel by using multi-layer perceptrons to transform the input hidden states \mathbf{H} into a common space, where aligned interactions between modalities occur.

Often, the matrix \mathbf{A} cannot be directly accessed due to FlashAttention-type (Dao et al., 2022) optimizations. Therefore, we develop an approach to extract \mathbf{A} while maintaining compatibility with the FlashAttention when applying our

sparsification. Please refer to the Appendix B.

3.2. Sparsification Guidance from Text to Vision

Estimation of Visual Token Significance. For a multi-modal model, we aim to estimate an impact of deleting a single token from one modality to other modalities. In the VLM case, we need to quantify how relevant a visual token is to text tokens in order to determine whether it can be pruned. Therefore, we naturally reuse the self-attention logits from VLM’s transformer layers as a reference since they already contain *language-to-vision* query results.

In particular, we take the interaction between the *query*-dimensional part of the language modality and the *key*-dimensional part of the vision modality as the basis for sparsification priority matrix $\mathbf{P} \in \mathbb{R}^{L_t \times L_v}$, where L_t and L_v are the lengths of text and visual tokens, defined by

$$\mathbf{P} = \mathbf{A}_{i,j}, \text{ and } (i,j) \in \{\mathbb{L}, \mathbb{I}\}, \quad (2)$$

where \mathbb{L} and \mathbb{I} denote the language instruction and image token sets, respectively.

Next, we obtain a vector $\tilde{\mathbf{p}}$ that estimates the significance of all visual tokens w.r.t. the text dimension as

$$\tilde{\mathbf{p}} = [\tilde{p}_1, \tilde{p}_2, \dots, \tilde{p}_{L_v}] = \frac{1}{L_t} \sum_{i=1}^{L_t} \mathbf{P}_i, \quad (3)$$

where we use $\tilde{\mathbf{p}}$ as an indicator for sparsification and a larger value in $\tilde{\mathbf{p}}$ means higher significance of the corresponding visual token. Calculation of (3) costs $L_t \times L_v$ FLOPs only while the access to already computed \mathbf{A} is considered as free, which highlights low complexity of the SparseVLM.

Relevant Text Token Selection. It is not appropriate to use all text tokens as a reference for visual sparsification. Figure 3 shows four representative cases where we compute the correlation between the prompt and the image. Case 3 highlights Tylenol, Advil, ibuprofen, while sticker, fridge in case 4 are significant, where a large proportion of question tokens in light red include little visual relevance. Therefore, it is unreasonable to make insignificant text tokens to rate visual tokens, and we need to select relevant text tokens (i.e., “*raters*”) for guidance.

Specifically, for an input image \mathbf{x}_v , the vision embedding tokens \mathbf{H}_v can be computed as

$$\mathbf{H}_v = \mathbf{W} \mathbf{Z}_v, \quad (4)$$

where \mathbf{Z}_v is the visual feature provided by visual encoder $\mathbf{Z}_v = g(\mathbf{x}_v)$, and \mathbf{W} is the projection matrix to convert \mathbf{Z}_v into vision embedding tokens \mathbf{H}_v . For the language instruction \mathbf{x}_q , it is transformed into text embedding tokens \mathbf{H}_q through the tokenizer. The above tokens both have the same dimensionality as the word embedding space. Then,

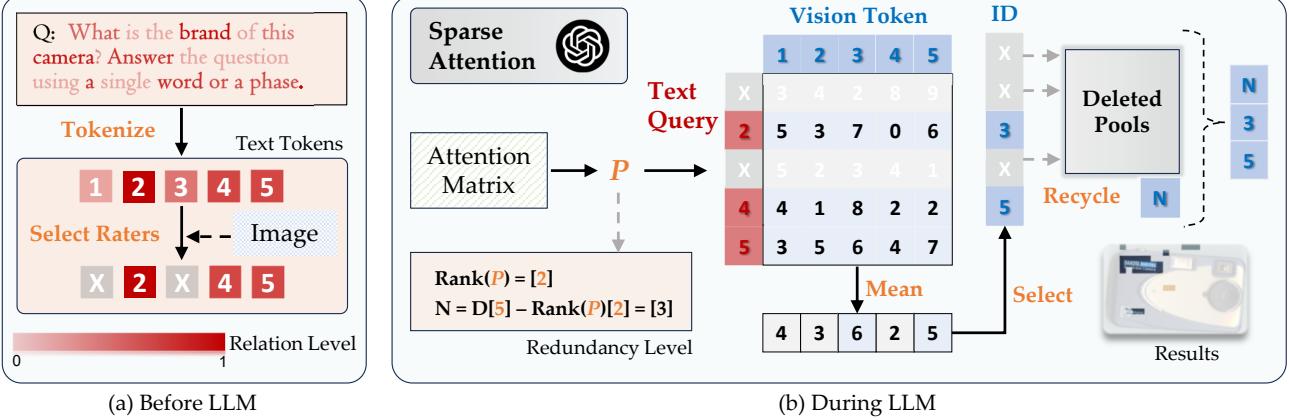


Figure 2. The architecture of SparseVLM. In stage (a), text raters are pre-selected before entering the sparsification LLM. In stage (b), adaptive sparsification is performed on LLM layers, involving computing redundancy and the recycling of reconstructed tokens.

we start to recognize which characters in the prompt are visually relevant and assign them the role of raters, which can be formulated as

$$s = \{ i \mid r_i \geq m \}, \quad i \in \{1, 2, \dots, L_t\}, \quad (5)$$

$$r = \frac{1}{L_v} \sum_{j=1}^{L_v} \left(\text{Softmax} \left(\mathbf{H}_v \mathbf{H}_q^T \right) \right)_j, \quad (6)$$

where $m = \text{mean}(r)$ and only candidates that exceed the m threshold become raters. The strategy s contains the indices of selected raters from the candidate list of L_t tokens. The (6) costs $L_t \times L_v \times 2D$ FLOPs that is only computed once before the decoder layer processing.

Sparsification Level Adaptation. Having obtained the token significance, we further propose a rank-based strategy to adaptively determine the level of vision sparsification at each decoder layer. Considering that *a full-rank matrix implies that all its rows or columns are linearly independent*, we use the rank of P to demonstrate the redundancy of the visual tokens. We argue that the difference between the dimension and rank of P reflects its redundancy and utilize a scaling factor λ to determine the number of deletions as

$$N = \lambda \times (L_v - \text{rank}(P)). \quad (7)$$

We then remove N visual tokens with the smallest values in P . Notably, if the result of N in a decoder layer is 0, we skip the layer without sparsification. This stage requires $L_t \times L_v \times \min(L_t, L_v)$ FLOPs for rank computation.

3.3. Visual Token Recycling

We progressively sparsify visual tokens in each layer in the decoder, which results in more discarded tokens at later stages. Despite being less significant, the pruned visual tokens with relatively large values in P still contain certain

information. To efficiently preserve more visual details with fewer tokens, we propose a token recycling strategy to aggregate and reconstruct tokens to be pruned.

Token Aggregation. We first recycle the pruned visual tokens \bar{h}_v with the top- τ (%) highest values in P from the deleted pool. Then, we group \bar{h}_v tokens with k -nearest neighbor density peak aggregation algorithm (Rodriguez, 2014) for adaptive token aggregation.

In particular, we first compute the local density ρ_i of the i th token of total $\tau \times N$ recycled tokens according to its k -nearest neighbors $\mathcal{K}(\bar{h}_v^i)$ as

$$\rho_i = \exp \left(-\frac{1}{k} \sum_{\bar{h}_v^j \in \mathcal{K}(\bar{h}_v^i)}^{i,j} \|\bar{h}_v^i - \bar{h}_v^j\|_2^2 \right). \quad (8)$$

Then, we compute the minimum distance between the recycled token \bar{h}_v^i and any other token with higher density (denoted as the distance indicator δ_i) that is defined by

$$\delta_i = \begin{cases} \min & \left\| \bar{h}_v^i - \bar{h}_v^j \right\|_2, \text{ if } \exists j \text{ s.t. } \rho_j > \rho_i, \\ \max & \left\| \bar{h}_v^i - \bar{h}_v^j \right\|_2, \text{ otherwise.} \end{cases} \quad (9)$$

We use $\rho_i \times \delta_i$ to indicate the score of each token, where the tokens with higher scores are likely to be cluster centers. Other tokens are then assigned to the nearest cluster center via cosine similarity. The FLOPs cost in this stage is $L_r \times (3L_r - 1) \times 2D + L_r$, where $L_r = \tau \times N$ is the length of recycled tokens, $C = \theta \times L_r$ is the number of cluster centers, and τ and θ are hyperparameters.

Token Reconstruction. Having performed token aggregation, the recycled tokens with similar semantics are classified into the same group. Then, the tokens $\mathbb{T} \in \mathbb{R}^{N_k \times D}$ in the k th group are reconstructed into a new compressed token $\mathbf{T}_k \in \mathbb{R}^{1 \times D}$ via the element-wise sum operation as

$$\mathbf{T}_k = \sum_{i=1}^{N_k} \mathbb{T}[i], \quad k \in \{1, 2, \dots, C\}, \quad (10)$$

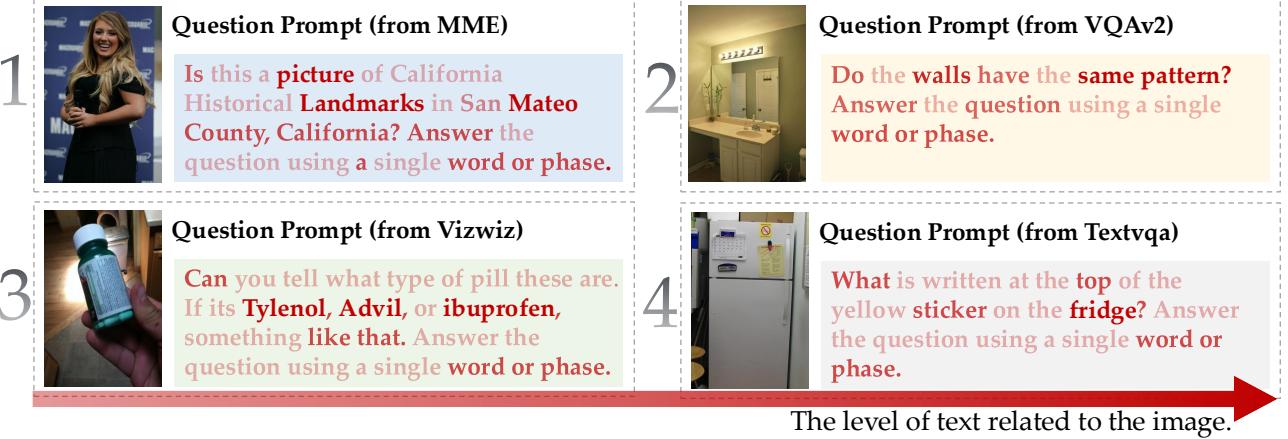


Figure 3. Sample prompts from four representative multimodal benchmarks. The darker the word, the greater its relationship to the image and the more valuable it is for reference. We see that some words are irrelevant to the vision domain (e.g., prepositions and pronouns) and should not be considered for visual sparsification. It is best viewed in color.

where N_k is the token number of the k th group and the operation costs $D \times (L_r - C)$ FLOPs.

3.4. Theoretical Analysis of Computational Complexity

We consider the computation of multi-head attention and feed-forward network (FFN) modules in the FLOPs estimation. Assuming N is the number of pruned tokens, D is the hidden state size, which is the same as the intermediate size in FFN, the FLOPs for one Transformer layer can be reduced by $6(N - C)D^2 + 2(N - C)^2D$. Besides, our partial step introduces minimal computation with the details provided in Appendix C. Thus, we estimate the FLOPs savings as the reduction part minus the additional overhead:

$$\begin{aligned} & \underbrace{\sum_i 6(N_i - C_i)D^2 + 2(N_i - C_i)^2D}_{\text{reduction part}} - \\ & \underbrace{2L_t L_v D - \sum_i L_t^i L_v^i (1 + \min(L_t^i, L_v^i)) - (6L_r^{i^2} + 2L_r^i)D - L_r^i}_{\text{overhead part}} \\ & \approx -2L_t L_v D + \sum_i D(6DN_i(1-x) + N_i^2(2+2x^2-4x-6(\tau)^2)) - L_t^{i^2} L_v^i \\ & \approx -2L_t L_v D + \sum_i DN_i(6D+2N_i) - L_t^{i^2} L_v^i, \end{aligned} \quad (11)$$

where $i \in \{1, 2, \dots, \Omega\}$ and Ω is the number of total layers, and $x = \tau \times \theta$ is a very small decimal that can be ignored.

4. Experiments

In this section, we validate our method within various vision-language architectures on comprehensive multimodal benchmarks, including image and video understanding tasks, to assess its generality, effectiveness, and efficiency.

4.1. Image Understanding Tasks

Datasets. For image-based multimodal evaluation, we conduct experiments on eight widely adopted benchmarks,

including GQA (Hudson & Manning, 2019), MMBench (MMB) (Liu et al., 2024c), MME (Fu et al., 2023), POPE (Li et al., 2023b), SQA (Lu et al., 2022), SEED-Bench (SEED) (Li et al., 2024a), VQA^{Text} (TextVQA) (Singh et al., 2019), and MMVet (Yu et al., 2024).

Implementation Details. We verify SparseVLM on three VLM frameworks: LLaVA (Liu et al., 2024b), Mini-Gemini (MGM) (Li et al., 2024c), and Qwen2-VL (Bai et al., 2023). LLaVA-1.5 employs CLIP-pretrained ViT-L as the visual tower, MGM further introduces a LAION-pretrained ConvNeXt-L (Liu et al., 2022) for high-resolution refinement, while Qwen2-VL owns dynamic resolution encoder.

Main Results. In Table 1, we present the performance of SparseLLaVA (LLaVA equipped with SparseVLM) on image understanding benchmarks. To intuitively assess the performance, we provide the results by percentage format for comparative analysis, and the accuracy of the vanilla model with the 100% upper limit. We set 3 vision token count configurations (192, 128, and 64) to check the advantages of SparseVLM comprehensively. When pruning from 576 to 192 tokens, the SparseLLaVA only decreases the average accuracy by 0.9% without additional training and exceeds ToMe (Bolya et al., 2023) 10.2%. When only 64 tokens are kept, our method outperforms FastV (Chen et al., 2024a) by a significant margin of 17.3%, while ToMe performs worst due to its direct merging. Furthermore, we also compare the recent method PDrop (Xing et al., 2025) training-free version, which has lower FLOPs computation. However, our method outperforms it in accuracy and latency, which are the most crucial metrics for practical deployment.

Figure 4 visualizes the performance of SparseMGM on POPE, TextVQA, and GQA. We find that our framework has an obvious advantage over FastV and ToMe. With the

Table 1. Performance of SparseLLaVA under different vision token configurations. The vanilla number of vision tokens is 576. The first line of each method is the raw accuracy of benchmarks, and the second line is the proportion relative to the upper limit.

| Method | GQA | MMB | MME | POPE | SQA | SEED | VQA ^{Text} | MMVet | Acc. (%) | FLOPs (T) | Latency (ms) |
|---------------------------------------|---------------|---------------|---------------|---------------|----------------|---------------|---------------------|----------------|---------------|-----------|--------------|
| <i>Upper Bound, 576 Tokens (100%)</i> | | | | | | | | | | | |
| Vanilla | 61.9 100% | 64.6 100% | 1864 100% | 85.9 100% | 69.5 100% | 60.3 100% | 58.3 100% | 30.9 100% | 100 | 4.62 | 57.82 |
| <i>Retain 192 Tokens (↓ 66.7%)</i> | | | | | | | | | | | |
| ToMe (ICLR23) | 54.3 87.7% | 60.5 93.5% | 1563 83.9% | 72.4 84.3% | 65.2 93.8% | 53.1 88.1% | 52.1 89.5% | 27.9 90.3% | 88.9 (↓ 11.1) | 2.05 | 34.06 |
| FastV (ECCV24) | 52.6 85.0% | 61.0 94.4% | 1605 86.1% | 64.8 75.4% | 69.1 99.4% | 52.1 86.4% | 52.5 90.1% | 26.7 86.4% | 87.9 (↓ 12.1) | 2.11 | 34.87 |
| PDrop (CVPR25) | 57.1 92.2% | 63.2 97.8% | 1766 94.7% | 82.3 95.8% | 70.2 101.0% | 54.7 90.7% | 56.1 96.2% | 30.5 98.7% | 95.9 (↓ 4.1) | 2.03 | 36.74 |
| SparseVLM | 59.5 96.1% | 64.1 99.2% | 1787 95.9% | 85.3 99.3% | 68.7 98.8% | 58.7 97.3% | 57.8 99.1% | 33.1 107.1% | 99.1 (↓ 0.9) | 2.14 | 36.50 |
| <i>Retain 128 Tokens (↓ 77.8%)</i> | | | | | | | | | | | |
| ToMe (ICLR23) | 52.4 84.7% | 53.3 82.4% | 1343 72.1% | 62.8 73.1% | 59.6 85.8% | 50.9 84.4% | 49.1 84.4% | 27.2 88.0% | 81.9 (↓ 18.1) | 1.62 | 30.00 |
| FastV (ECCV24) | 49.6 80.1% | 56.1 86.8% | 1490 79.9% | 53.4 62.2% | 68.6 98.7% | 48.1 79.8% | 50.5 86.6% | 26.3 85.1% | 82.4 (↓ 17.6) | 1.70 | 30.70 |
| PDrop (CVPR25) | 56.0 90.5% | 61.1 95.4% | 1664 89.3% | 82.3 95.8% | 69.9 100.6% | 53.3 88.4% | 55.1 94.5% | 30.8 99.7% | 94.3 (↓ 5.7) | 1.62 | 37.77 |
| SparseVLM | 58.4 94.3% | 64.5 99.8% | 1746 93.7% | 85.0 99.0% | 68.6 98.7% | 58.2 96.5% | 56.7 97.3% | 29.0 93.9% | 96.7 (↓ 3.3) | 1.72 | 33.28 |
| <i>Retain 64 Tokens (↓ 88.9%)</i> | | | | | | | | | | | |
| ToMe (ICLR23) | 48.6 78.5% | 43.7 67.5% | 1138 61.1% | 52.5 61.1% | 50.0 71.9% | 44.0 73.0% | 45.3 77.8% | 24.1 78.0% | 71.1 (↓ 28.9) | 1.19 | 26.52 |
| FastV (ECCV24) | 46.1 74.5% | 47.2 73.1% | 1255 67.3% | 38.2 44.5% | 68.7 98.8% | 43.7 72.5% | 47.8 82.0% | 19.6 63.4% | 72.0 (↓ 28.0) | 1.29 | 27.30 |
| PDrop (CVPR25) | 41.9 67.7% | 33.3 51.6% | 1092 58.6% | 55.9 65.1% | 69.2 99.6% | 40.0 66.3% | 45.9 78.7% | 30.7 99.4% | 73.4 (↓ 26.6) | 1.18 | 43.41 |
| SparseVLM | 53.8 86.9% | 60.1 93.0% | 1589 85.2% | 77.5 90.2% | 69.8 100.4% | 52.2 86.6% | 53.4 91.6% | 24.9 80.6% | 89.3 (↓ 10.7) | 1.30 | 29.89 |

Table 2. Performance of SparseVLM on Qwen2-VL.

| Tokens | MMB | POPE | VQA ^{Text} | Avg. |
|---------|-------------|-------------|---------------------|------|
| Dynamic | 80.5 (1323) | 86.4 (1311) | 84.3 (1326) | 83.7 |
| 600 | 79.6 | 86.5 | 80.3 | 82.1 |
| 500 | 78.8 | 86.3 | 79.0 | 81.4 |
| 400 | 79.0 | 85.8 | 77.1 | 80.7 |

reduction of tokens, the gap between FastV and SparseVLM is increasing sharply. The reason is that, compared to FastV and ToMe, the text-aware strategy enables us to accurately locate visual tokens with more details, while the recycling of pruned tokens further reduces information loss.

We further investigate our efficacy on Qwen2-VL. In Table 2, when 54.5% of vision tokens are removed, Qwen2-VL maintains an accuracy of 98.0%. Furthermore, for every 100 tokens pruned, the accuracy only drops by approximately 0.8%. This validates the effectiveness of our method at high resolutions and its compatibility with variable resolutions.

4.2. Video Understanding Tasks

Datasets. We test on four common video question answering benchmarks, TGIF-QA (Jang et al., 2017), MSVD-QA (Xu et al., 2017), MSRVTT-QA (Xu et al., 2017), and ActivityNet-QA (Yu et al., 2019). Specifically, following FastV’s (Chen et al., 2024a) setup, we use the first 1000 samples per benchmark and score them using the Video-

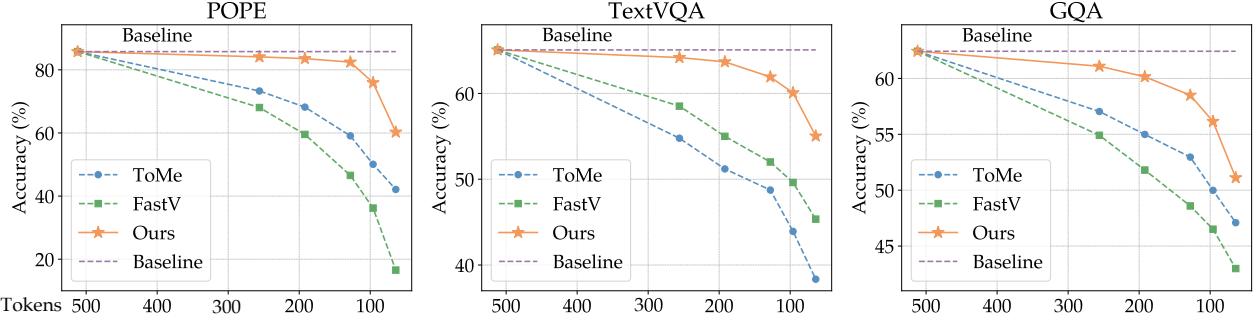


Figure 4. Performance of MGM w/ SparseVLM on three multimodal benchmarks. The horizontal axis represents the remaining number of vision tokens, while the vertical axis means the accuracy after percentage normalization.

Table 3. The results of Video-LLaVA with SparseVLM on video question answering task. The original number of video tokens is 2048, while our experiment collectively prunes it down to 194 tokens. FastV (Chen et al., 2024a) is included for comparison.

| Method | TGIF | | MSVD | | MSRVTT | | ActivityNet | | Avg. |
|----------------|-------|-------|-------|-------|--------|-------|-------------|-------|-----------|
| | Acc. | Score | Acc. | Score | Acc. | Score | Acc. | Score | |
| Video-LLaVA | 18.9 | 2.54 | 72.0 | 3.95 | 57.1 | 3.45 | 43.6 | 3.81 | 47.9 3.44 |
| | 100% | +0.00 | 100% | +0.00 | 100% | +0.00 | 100% | +0.00 | |
| FastV (ECCV24) | 10.2 | 2.29 | 58.3 | 3.62 | 52.3 | 3.42 | 41.3 | 3.76 | 40.5 3.27 |
| | 54.0% | -0.34 | 81.0% | -0.33 | 91.6% | -0.03 | 94.7% | -0.12 | |
| Ours | 14.9 | 2.41 | 71.7 | 3.94 | 56.1 | 3.43 | 45.1 | 3.81 | 47.0 3.40 |
| | 78.8% | -0.13 | 99.6% | -0.01 | 98.3% | -0.02 | 103.4% | -0.00 | |

ChatGPT (Maaz et al., 2024) evaluation tool, acknowledging the characteristic length imbalances in these datasets.

Implementation Details. We directly apply our SparseVLM for Video-LLaVA (Lin et al., 2024), which is composed of several key components, including language bind encoder f_M^v (Zhu et al., 2024a) for extracting features from raw visual inputs (e.g., images or videos), a language decoder model f_L such as Vicuna (Touvron et al., 2023), a visual projection layer f_P , and a word embedding layer f_T .

Main Results. In Table 3, we set the Video-LLaVA with 2048 video tokens as our upper bound for an overall average accuracy of 100.0% and a score of +0.00. To make a fair comparison, we both preserve 194 vision tokens (90.5% pruning ratio) for FastV (Chen et al., 2024a) and SparseVLM. It is clear that our approach consistently outperforms FastV across all benchmarks, both in accuracy (Acc.) and GPT evaluation score. SparseVideoLLaVA achieves a total average accuracy of 95.0%, a significant **14.7%** higher than 80.3% of FastV. (From the GPT score perspective, SparseVLM only loses 0.04 points compared to 0.17 points of FastV.) These improvements suggest that when handling video modality containing temporal features, SparseVLM continues to deliver strong performance, generating accurate responses to diverse questions while utilizing significantly fewer tokens. This achieves an effective trade-off between inference efficiency and model performance.

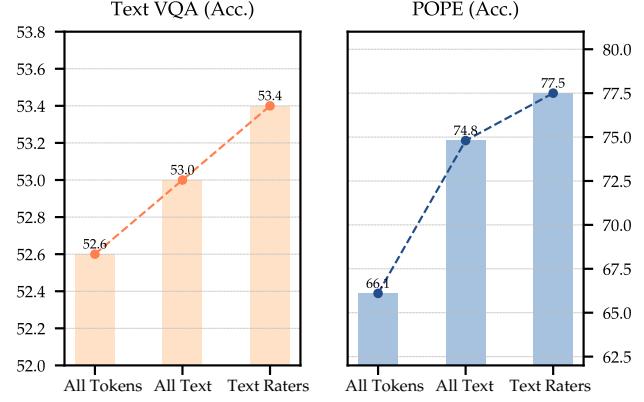


Figure 5. The ablation study of text raters on LLaVA 7B.

5. Analysis

5.1. Relevant Text Token Selection

We propose a selection mechanism to localize visually irrelevant text tokens to limit their negative effects in rating the significance of vision tokens. Here we conduct experiments to analyze the effects of the mechanism in Figure 5. Under the same number of vision tokens (64), we have 3 settings (using all tokens, only text tokens, and only text raters we select) with LLaVA (Liu et al., 2024a) to judge vision token candidates. In TextVQA (Singh et al., 2019), by building upon the text-aware manner, our mechanism improves the baseline (all tokens) by 0.8%, which validates that our extra selection is effective. Besides, we further outperform the vanilla text-aware method (only text tokens) by **2.7%** on POPE (Li et al., 2023b). The huge margin means POPE sparsification is quite sensitive to question prompts, and text guidance is necessary. In summary, text rater selection is general and improves the performance across scenarios.

5.2. Recycling of Pruned Tokens

To validate the effectiveness of our token recycling strategy, we perform ablation experiments on the LLaVA model (Liu et al., 2024a). The results are presented in Table 4. Across multiple sparsity ratios (64, 96, 128, 192), our algorithm achieves a significant average performance improvement of

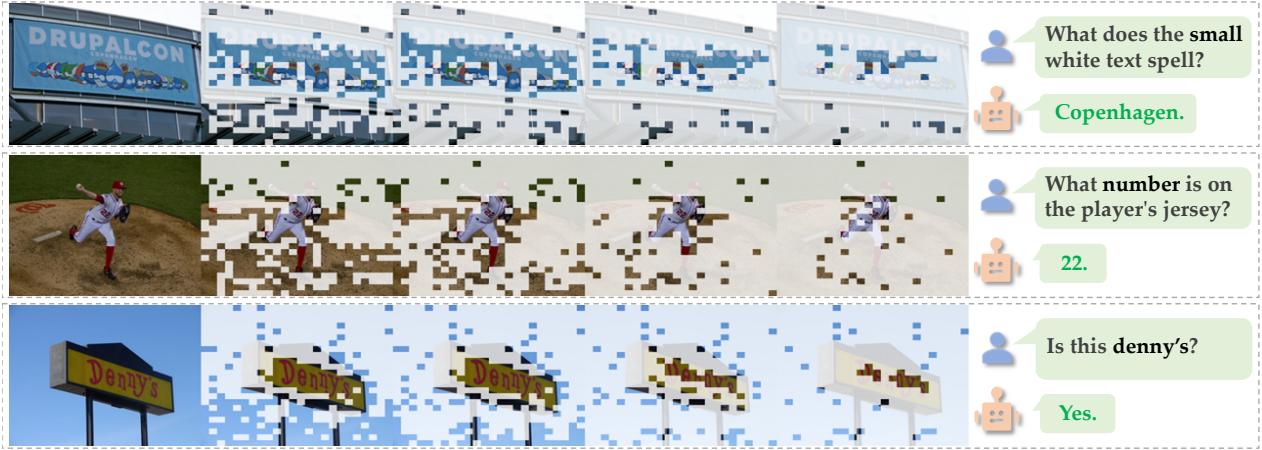


Figure 6. **Visualization of SparseVLM on different VQA prompts.** From left to right, the visual representation becomes increasingly sparse, leaving fewer vision tokens. Best viewed in color.

Table 4. **Ablation study on token reconstruction (TR).** Experiments are conducted on GQA and POPE on LLaVA 7B.

| Benchmark | Tokens | | | | Avg. |
|-------------|-------------|-------------|-------------|-------------|-------------|
| | 64 | 96 | 128 | 192 | |
| GQA | 52.2 | 55.2 | 58.1 | 59.4 | 56.2 |
| + TR | 53.8 | 56.4 | 58.4 | 59.5 | 57.0 |
| POPE | 72.8 | 77.5 | 83.7 | 85.2 | 79.8 |
| + TR | 77.5 | 81.9 | 85.0 | 85.3 | 82.4 |

1.2% and 7.2% on TextVQA (Singh et al., 2019) and POPE (Li et al., 2023b), respectively. Notably, as the number of pruned vision tokens increases, the benefit brought by our recycling method increases. For instance, when pruning from 192 to 64 tokens, the pruned token recycling significantly boosts the accuracy from **1.5%** to **17.7%** on POPE. We argue that when the size of the deleted pool grows, the amount of lost information increases. Our method effectively recycles the lost information and compresses it into few slots using the proposed reconstruction mechanism.

5.3. Computational Efficiency

SparseVLM affords significant efficiency and storage gains for the inference process. We conduct a comparative analysis of CUDA time, and FLOPs on LLaVA-7B, and compare our method with the baseline method and FastV (Chen et al., 2024a). As displayed in Table 1, we conduct an inference efficiency analysis on a single NVIDIA A100-80GB with identical lengths of text prompts and single-image inputs. Compared to the baseline model, SparseVLM achieves a significant reduction of 43.1% in CUDA time and 62.8% in FLOPs while keeping 96.7% accuracy. Despite SparseVLM has a minimal overhead to calculate text raters and cluster-pruned vision tokens, it leads to fewer than FastV tokens with comparable accuracy. Additionally, SparseVLM saves

67% cache memory compared to vanilla LLaVA (where 302.4MB is reduced to 100.8MB), while keeping 99.1% accuracy. More efficiency visualization (e.g., efficiency on VideoLLaVA) can be found in the Appendix G.

5.4. Qualitative Visualization

As shown in Figure 6, we visualize SparseVLM on various VQA questions. From left to right, we visualize the results after we apply token pruning to different layers. As the number of layers increases, more tokens are pruned and the Region of Interest (ROI) is gradually refined. The model systematically reduces less relevant image information while retaining key tokens closely tied to the question. The visualization reveals that SparseVLM, although discarding some overall image details, effectively retains essential visual tokens. These preserved tokens encapsulate the features necessary for answering the question, focusing on more relevant visual regions through their interaction with the question. More cases are in the Appendix H.

6. Conclusion

This paper introduced a text-aware training-free token optimization approach called SparseVLM which significantly decreased the test-time computations of various VLMs. Unlike prior methods, SparseVLM optimized VLMs without introducing extra parameters and fine-tuning costs. We achieved a more compact visual representation by employing the rank of attention matrices to determine pruning ratios and by recycling the pruned tokens via the reconstruction mechanism to reduce the information loss. Experiments demonstrated that e.g. the LLaVA when equipped with SparseVLM achieved 37.0% reduction in latency with a compression ratio of 77.8% while maintaining 97% of the original accuracy. Moreover, our method exceeded FastV accuracy by 14.7% in video understanding tasks. Our SparseVLM can provide practical benefits for deploying off-the-shelf VLMs on edge devices and in the cloud setting.

Acknowledgments

This work was supported by the National Science and Technology Major Project (No. 2022ZD0117800) and by the National Natural Science Foundation of China under Grant 62472008.

Impact Statement

Our SparseVLM provides practical advantages for deploying off-the-shelf large vision-language models on edge devices and cloud platforms. While our work does not present any evident societal implications, we believe it is unnecessary to emphasize this aspect in the current context.

References

- Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 2022.
- Bai, J., Bai, S., Yang, S., Wang, S., Tan, S., Wang, P., Lin, J., Zhou, C., and Zhou, J. Qwen-VL: A frontier large vision-language model with versatile abilities. *arXiv:2308.12966*, 2023.
- Bolya, D., Fu, C.-Y., Dai, X., Zhang, P., Feichtenhofer, C., and Hoffman, J. Token merging: Your vit but faster. In *International Conference on Learning Representations*, 2023.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 2020.
- Cai, M., Yang, J., Gao, J., and Lee, Y. J. Matryoshka multimodal models. In *International Conference on Learning Representations*, 2025.
- Cha, J., Kang, W., Mun, J., and Roh, B. Honeybee: Locality-enhanced projector for multimodal llm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2024.
- Chen, L., Zhao, H., Liu, T., Bai, S., Lin, J., Zhou, C., and Chang, B. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *Proceedings of the European Conference on Computer Vision*, 2024a.
- Chen, Z., Wu, J., Wang, W., Su, W., Chen, G., Xing, S., Zhong, M., Zhang, Q., Zhu, X., Lu, L., et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2024b.
- Dai, W., Li, J., Li, D., Tiong, A., Zhao, J., Wang, W., Li, B., Fung, P., and Hoi, S. InstructBLIP: Towards general-purpose vision-language models with instruction tuning. *Advances in Neural Information Processing Systems*, 2023.
- Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. FlashAttention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 2022.
- Du, Z., Qian, Y., Liu, X., Ding, M., Qiu, J., Yang, Z., and Tang, J. Glm: General language model pretraining with autoregressive blank infilling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2022.
- Fu, C., Chen, P., Shen, Y., Qin, Y., Zhang, M., Lin, X., Yang, J., Zheng, X., Li, K., Sun, X., et al. MME: A comprehensive evaluation benchmark for multimodal large language models. *arXiv:2306.13394*, 2023.
- Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., and Parikh, D. 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*, 2017.
- Hudson, D. A. and Manning, C. D. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- Jang, Y., Song, Y., Yu, Y., Kim, Y., and Kim, G. Tgiff-qa: Toward spatio-temporal reasoning in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- Kondratyuk, D., Yu, L., Gu, X., Lezama, J., Huang, J., Hornung, R., Adam, H., Akbari, H., Alon, Y., Birodkar, V., et al. Videopoet: A large language model for zero-shot video generation. In *International Conference on Machine Learning*, 2024.
- Li, B., Ge, Y., Ge, Y., Wang, G., Wang, R., Zhang, R., and Shan, Y. Seed-bench: Benchmarking multimodal large language models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2024a.
- Li, J., Li, D., Savarese, S., and Hoi, S. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International Conference on Machine Learning*, 2023a.

- Li, Y., Du, Y., Zhou, K., Wang, J., Zhao, W. X., and Wen, J.-R. Evaluating object hallucination in large vision-language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2023b.
- Li, Y., Wang, C., and Jia, J. LLaMA-VID: An image is worth 2 tokens in large language models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2024b.
- Li, Y., Zhang, Y., Wang, C., Zhong, Z., Chen, Y., Chu, R., Liu, S., and Jia, J. Mini-gemini: Mining the potential of multi-modality vision language models. *arXiv:2403.18814*, 2024c.
- Lin, B., Ye, Y., Zhu, B., Cui, J., Ning, M., Jin, P., and Yuan, L. Video-llava: Learning united visual representation by alignment before projection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2024.
- Liu, H., Li, C., Li, Y., and Lee, Y. J. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2024a.
- Liu, H., Li, C., Wu, Q., and Lee, Y. J. Visual instruction tuning. *Advances in Neural Information Processing Systems*, 2024b.
- Liu, Y., Duan, H., Zhang, Y., Li, B., Zhang, S., Zhao, W., Yuan, Y., Wang, J., He, C., Liu, Z., et al. Mmbench: Is your multi-modal model an all-around player? In *Proceedings of the European Conference on Computer Vision*, 2024c.
- Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A convnet for the 2020s. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022.
- Lu, P., Mishra, S., Xia, T., Qiu, L., Chang, K.-W., Zhu, S.-C., Tafjord, O., Clark, P., and Kalyan, A. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 2022.
- Maaz, M., Rasheed, H., Khan, S., and Khan, F. Videochatgpt: Towards detailed video understanding via large vision and language models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2024.
- Marr, D. *Vision: A computational investigation into the human representation and processing of visual information*. MIT press, 2010.
- Peng, B., Li, C., He, P., Galley, M., and Gao, J. Instruction tuning with gpt-4. *arXiv:2304.03277*, 2023.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.
- Rodriguez, A. Clustering by fast search and find of density peaks. *Science*, 2014.
- Shang, Y., Cai, M., Xu, B., Lee, Y. J., and Yan, Y. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. *arXiv preprint arXiv:2403.15388*, 2024.
- Singh, A., Natarjan, V., Shah, M., Jiang, Y., Chen, X., Parikh, D., and Rohrbach, M. Towards VQA models that can read. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- Stewart, G. W. On the early history of the singular value decomposition. *SIAM review*, 1993.
- Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. Gemini: a family of highly capable multimodal models. *arXiv:2312.11805*, 2023.
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv:2302.13971*, 2023.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.
- Wu, S., Chen, J., Lin, K. Q., Wang, Q., Gao, Y., Xu, Q., Xu, T., Hu, Y., Chen, E., and Shou, M. Z. Videollm-mod: Efficient video-language streaming with mixture-of-depths vision computation. *Advances in Neural Information Processing Systems*, 2024.
- Xing, L., Huang, Q., Dong, X., Lu, J., Zhang, P., Zang, Y., Cao, Y., He, C., Wang, J., Wu, F., et al. Pyramiddrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2025.
- Xu, D., Zhao, Z., Xiao, J., Wu, F., Zhang, H., He, X., and Zhuang, Y. Video question answering via gradually refined attention over appearance and motion. In *Proceedings of the ACM international conference on Multimedia*, 2017.

Yao, L., Li, L., Ren, S., Wang, L., Liu, Y., Sun, X., and Hou, L. DeCo: Decoupling token compression from semantic abstraction in multimodal large language models. *arXiv:2405.20985*, 2024.

Ye, X., Gan, Y., Huang, X., Ge, Y., Shan, Y., and Tang, Y. VoCo-LLaMA: Towards vision compression with large language models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2025.

Yu, W., Yang, Z., Li, L., Wang, J., Lin, K., Liu, Z., Wang, X., and Wang, L. Mm-vet: Evaluating large multimodal models for integrated capabilities. In *International Conference on Machine Learning*, 2024.

Yu, Z., Xu, D., Yu, J., Yu, T., Zhao, Z., Zhuang, Y., and Tao, D. Activitynet-qa: A dataset for understanding complex web videos via question answering. In *AAAI*, 2019.

Zhang, Y., Huang, T., Fan, C.-K., Dong, H., Li, J., Wang, J., Cheng, K., Zhang, S., Guo, H., et al. Unveiling the tapestry of consistency in large vision-language models. *Advances in Neural Information Processing Systems*, 2024a.

Zhang, Y., Huang, T., Liu, J., Jiang, T., Cheng, K., and Zhang, S. Freekd: Knowledge distillation via semantic frequency prompt. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2024b.

Zhu, B., Lin, B., Ning, M., Yan, Y., Cui, J., HongFa, W., Pang, Y., Jiang, W., Zhang, J., Li, Z., et al. Languagebind: Extending video-language pretraining to n-modality by language-based semantic alignment. In *International Conference on Learning Representations*, 2024a.

Zhu, D., Chen, J., Shen, X., Li, X., and Elhoseiny, M. Minigpt-4: Enhancing vision-language understanding with advanced large language models. In *International Conference on Learning Representations*, 2024b.

Appendix

A. The Redundancy of Visual Tokens in VLMs

In non-textual tasks, such as classification or detection, downsampling is commonly used to reduce visual redundancy and enhance model training efficiency (Zhang et al., 2024b). Figure 7 illustrates this process, showing the reduction of tokens from 1166 to 576 in a downsampled image, resulting in a 50% efficiency boost but a 15% information loss (entropy decreased from 7.44 to 6.13). This trade-off is acceptable for such tasks. Conversely, for text-related tasks like visual question answering (VQA), which involve both text and vision modalities, a distinct approach is required. Highlighting the most information-dense text (88% of total text) alongside the region pertinent to the query in the image (38% of total image), we observe that image information is typically sparser than textual data. Hence, our SparseVLM method incrementally prunes visual token redundancy, maintaining crucial information for task accuracy. This strategy enhances model efficiency.

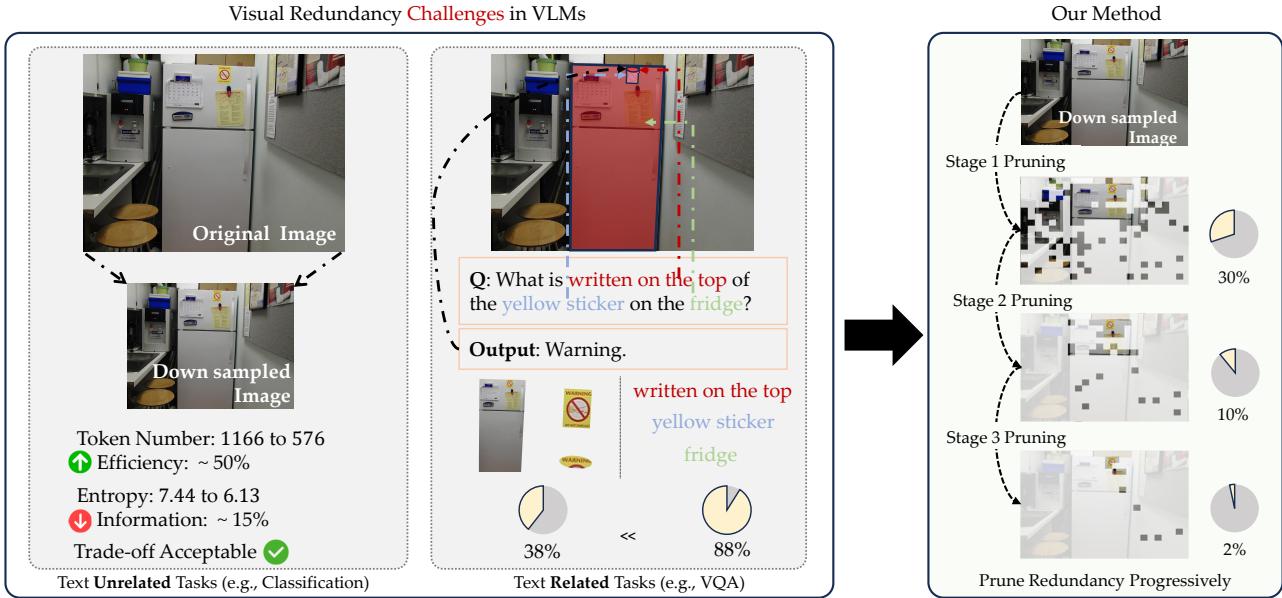


Figure 7. Analysis of visual redundancy in different vision tasks.

B. Compatibility with FlashAttention

To ensure compatibility between SparseVLM and FlashAttention (Dao et al., 2022) when extracting the matrix \mathbf{A} or \mathbf{P} , we devise the **dual-flash attention** operation to directly obtain the average attention scores relative to the text raters. This operation is lightweight and enjoys the efficiency of FlashAttention. Specifically, the first forward pass operates identically to the original FlashAttention, generating the necessary hidden states. In the second forward pass, we introduce a specially designed \mathbf{V} matrix. In this matrix, for the rows corresponding to the text raters we wish to analyze, we set the values to the reciprocal of the number of text raters. This configuration allows the inner product between the attention map and the \mathbf{V} matrix to return the mean value of the attention scores for the selected text raters directly in FlashAttention. With the mean value, we perform a top- k selection to identify the visual tokens to retain. Tokens that are excluded during this process are converted into masks, which are then applied to the hidden states produced by the first FlashAttention pass to complete the pruning operation. This method enables efficient integration of pruning with FlashAttention while preserving compatibility and computational efficiency. The specific principles and calculation of SparseVLM FlashAttention are as follows:

- 1. Attention Score Calculation.** For each block B , compute the scaled dot-product attention scores as

$$\mathbf{S}_B = \frac{\mathbf{Q}_B \mathbf{K}_B^T}{\sqrt{d_k}},$$

where \mathbf{S}_B is the attention score matrix computed within the block.

- 2. Block-wise Softmax.** To ensure numerical stability, the Softmax is computed using the log-sum-exp trick as

(a) Subtract the maximum value for numerical stability:

$$\mathbf{S}'_B = \mathbf{S}_B - \max(\mathbf{S}_B, \text{axis} = 1)$$

(b) Normalize:

$$\mathbf{P}_B = \frac{\exp(\mathbf{S}'_B)}{\sum \exp(\mathbf{S}'_B, \text{axis} = 1)}$$

3. **Special V Matrix.** In order to return the mean value of the attention scores for the selected text raters directly with the FlashAttention, we need to design a special V matrix.

$$\mathbf{V}_{ij} = \begin{cases} 1/n, & \text{if } i \in \{i_1, i_2, \dots, i_k\}, \\ 0, & \text{otherwise.} \end{cases}$$

Here, \mathbf{V} is an $n \times d$ matrix, n is the total number of rows in the matrix, i is the row index, $1 \leq i \leq n$, $s = \{i \mid r_i \geq m\}$, $i \in \{1, 2, \dots, L_t\}$ define the text raters which we selected in Section 3.2.

4. **Incremental Accumulation.** Rather than storing \mathbf{P} explicitly, the result is directly accumulated into the output using:

$$\mathbf{O}_B = \mathbf{P}_B \cdot \mathbf{V}_B$$

The final result is obtained by concatenating all blocks:

$$\mathbf{O} = \text{Concat}(\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_B)$$

5. **Streaming Softmax.** When combining multiple blocks, an incremental softmax computation ensures that normalization is maintained across the entire sequence:

$$\text{Softmax}(\mathbf{S}) = \exp(\mathbf{S}) / \sum \exp(\mathbf{S})$$

This avoids global dependencies and enables efficient block-wise computation.

6. **Top- k Selection for Visual Tokens.** The top- k selection can be expressed as:

$$\mathbf{O}_k = \{x_i \in \mathbf{O}_v \mid \text{rank}(x_i, \mathbf{O}_v) \leq k\},$$

$$\mathbf{O}_v = \{y_j \in \text{mean}(\mathbf{O}) \mid \text{visual tokens start} \leq j \leq \text{visual tokens end}\}.$$

where $\mathbf{O} = \text{Concat}(\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_B)$ is the output array of the second FlashAttention, \mathbf{O}_v is the visual tokens part of \mathbf{O} , $\text{rank}(x_i, \mathbf{O}_v)$ represents the position of x_i in \mathbf{O}_v when sorted in descending order.

The corresponding indices of the top- k elements are:

$$\mathbf{I}_k = \{i \mid x_i \in \mathbf{O}_k\}.$$

7. **Summary of SparseVLM with FlashAttention using Top- k Selection.** The complete process of SparseVLM FlashAttention can be summarized as

$$\begin{aligned} \mathbf{I}_k = \{i \mid x_i \in \{y_j \in \mathbf{O}_v \mid \text{rank}(y_j, \text{mean}(\text{Concat}(\bigcup_B \text{Softmax} \left(\frac{\mathbf{Q}_B \mathbf{K}_B^T}{\sqrt{d_k}} - \max(\mathbf{S}_B) \right) \cdot \mathbf{V}_B \bigcup_B [\text{visual tokens start : visual tokens end}]))\})\}. \end{aligned}$$

Here, each block B is processed independently, and the results are combined using incremental normalization.

C. Computing Budget Detailed Estimation

Estimation of Visual Token Significance. In this stage, only the equation 3 averaging process requires computation. Each visual token undergoes $L_t - 1$ additions and one division. With L_v visual tokens in total, the number of FLOPs for this stage is $(L_t - 1 + 1) \times L_v = L_t \times L_v$.

Relevant Text Selection. In this process, given that official PyTorch implementation for Softmax and Averaging operations, the FLOPs for equation 6 can be approximately simplified to the matrix multiplication between H_v and H_q . The result has a shape of $L_v \times L_t$, where each element undergoes D multiplications and additions. Therefore, the FLOP count can be expressed as $L_t \times L_v \times 2D$.

Sparsification Level Adaptation. The rank of a matrix is typically computed using singular value decomposition (SVD) (Stewart, 1993). With the selected appropriate threshold, the number of above the threshold singular values determines the rank of the matrix. The FLOPs involved in this process can be approximated as $L_t \times L_v \times \min(L_t, L_v)$.

Token Aggregation. At this stage, the first part is to perform a nearest neighbor search for each element in the matrix. With the $L_r \times D$ matrix, this task can be simplified to calculate the distances between L_r elements, resulting in a total of $L_r \times (L_r - 1)/2$ distance calculations. Each distance computation requires sequentially executing subtraction, squaring, addition, and square root operations on D elements. Consequently, the number of FLOPs in the nearest neighbor search is $L_r \times (L_r - 1)/2 \times 4D = L_r \times (L_r - 1) \times 2D$.

The second part is density calculation. Since the operations of averaging and applying the exponential function are implemented by the official PyTorch, this part can be simplified by the matrix squaring. Therefore, the FLOPs for this part are $L_r \times L_r \times 2D$.

The third part is distance indicator calculation. The computation can be approximately simplified to compute $\rho_i \times \delta_i$. Therefore, the FLOPs for this part can be approximated as $L_r \times L_r \times 2D$.

The last part is clustering. In this part, we need to select C tokens with the highest scores from a total of L_r tokens to serve as cluster centers, and the FLOPs can be approximated as L .

In summary, the total FLOPs for this stage are given by

$$\begin{aligned} \text{FLOPs} &= \underbrace{L_r \times (L_r - 1) \times 2D}_{\text{Nearest Neighbors Search}} + \underbrace{L_r \times L_r \times 2D}_{\text{Density Calculation}} + \underbrace{L_r \times L_r \times 2D}_{\text{Distance Indicator Calculation}} + \underbrace{L}_{\text{Select Cluster Center}} \\ &= L_r \times (3L_r - 1) \times 2D + L. \end{aligned}$$

Token Reconstruction. Token reconstruction involves performing a weighted sum for each group, excluding the cluster center. Thus, there are $L_r - C$ elements to sum where each one has $1 \times D$ dimensions. Consequently, the number of FLOPs for this operation is $D \times (L_r - C)$.

D. Dataset

We conducted experiments on several widely used visual understanding benchmarks.

GQA. (Hudson & Manning, 2019) The GQA is composed of three parts: scene graphs, questions, and images. The image part contains images, as well as the spatial features of images and the features of all objects in images. The questions in GQA are designed to test the understanding of visual scenes and the ability to reason about different aspects of an image.

MMBench. (Liu et al., 2024c) The MMBench benchmark comprehensively evaluates the model's overall performance across multiple dimensions. It includes three levels of ability dimensions. The first level (L-1) consists of two main abilities, perception and reasoning. The second level (L-2) expands based on the first level, including six sub-abilities. The third level (L-3) further refines the second level, encompassing 20 specific ability dimensions. This hierarchical structure enables a granular and comprehensive evaluation of the model's various capabilities.

MME. (Fu et al., 2023) The MME benchmark is also a comprehensive benchmark meticulously designed to thoroughly evaluate various aspects of a model's performance. It consists of 14 subtasks that specifically aim to evaluate both the model's perceptual and cognitive abilities. By utilizing manually constructed instruction-answer pairs and concise instruction design, it effectively mitigates issues such as data leakage and unfair evaluation of model performance.

POPE. (Li et al., 2023b) The POPE benchmark is primarily used to evaluate the degree of Object Hallucination in models. It reformulates hallucination evaluation by requiring the model to answer a series of specific binary questions regarding the presence of objects in images. Accuracy, Recall, Precision, and F1 Score are effectively employed as reliable evaluation metrics to precisely measure the model’s hallucination level under three different sampling strategies.

ScienceQA. (Lu et al., 2022) The ScienceQA benchmark covers a rich diversity of domains, including natural science, language science, and social science. Within each subject, questions are categorized first by the topic, then by the category, and finally by the skill. This hierarchical categorization results in 26 topics, 127 categories, and 379 skills, providing a comprehensive and diverse range of scientific questions. It provides a comprehensive evaluation of a model’s capabilities in multimodal understanding, multi-step reasoning, and interpretability.

VQA-v2. (Goyal et al., 2017) The VQA-v2 benchmark evaluates the model’s visual perception capabilities through open-ended questions. It consists of 265,016 images, covering a wide variety of real-world scenes and objects, providing rich visual contexts for the questions. For each question, there are 10 ground truth answers provided by human annotators, which allows for a comprehensive evaluation of the performance of different models in answering the questions accurately.

TextVQA. (Singh et al., 2019) The TextVQA benchmark focuses on the comprehensive integration of diverse text information within images. It meticulously evaluates the model’s text understanding and reasoning abilities through a series of visual question-answering tasks with rich textual information. Models need to not only understand the visual content of the images but also be able to read and reason about the text within the images to answer the questions accurately.

MMVet. (Yu et al., 2024) The MMVet benchmark is designed based on the insight that the intriguing ability to solve complicated tasks is often achieved by a generalist model being able to integrate different core vision-language capabilities. MM-Vet defines 6 core VL capabilities and examines the 16 integrations of interest derived from the capability combination.

TGIF-QA. (Jang et al., 2017) The TGIF-QA benchmark is an extension of the image question answering (ImageQA) task to the video domain, aiming to promote the development of video question answering techniques. It contains 165,000 question answer pairs in total and requires the model to comprehend the details of GIF videos. Specifically, it introduces three new tasks for VideoQA (repetition count, repeating action, and state transition), which require spatio-temporal reasoning from videos, and frame QA tasks that can be answered from one of the frames.

MSVD-QA. (Xu et al., 2017) The MSVD-QA benchmark is based on the existing Microsoft Research Video Description (MSVD) dataset and contains 1970 video clips and approximately 50.5K QA pairs. The questions and answers are diverse in nature, covering a wide range of topics and aspects related to the video content. Due to its relatively large data size and the diversity of questions, it is widely used for video question answering tasks and video caption tasks. The tasks formed in it are open-ended questions, consisting of five types of questions: what, who, how, when, and where.

MSRVTT-QA. (Xu et al., 2017) The MSRVTT-QA benchmark consists of 10K video clips and 243k question answer pairs. One of the main challenges addressed by the MSRVTT-QA benchmark is the complexity of understanding and reasoning about video content. Videos contain both visual and temporal information, and models need to be able to effectively process and integrate these aspects to answer the questions accurately. The tasks formed in it also consist of five types of questions, similar to the MSVD-QA benchmark.

ActivityNet-QA (Yu et al., 2019) The ActivityNet-QA benchmark contains 58,000 human-annotated QA pairs on 5,800 videos derived from the ActivityNet dataset. The questions are designed to cover a range of types, including motion, spatial relationship, and temporal relationship, which challenge the model to understand and reason about the video content at different levels and evaluate the performance of VideoQA models in long-term spatio-temporal reasoning.

E. Implementation Details

All of our experiments are conducted on a single NVIDIA A100-80G GPU. The implementation is carried out in Python 3.10, utilizing PyTorch 2.1.2, CUDA 11.8, and transformers 4.31.0. The inference follows the evaluation settings established by LLaVA (Liu et al., 2024b). For LLaVA-1.5-7/13B, Mini-Gemini (MGM), and Qwen-VL, we follow the same inference setting as the original paper as it is publicly available^{1 2 3}. For video understanding tasks, we adopt the same inference setup

¹github.com/haotian-liu/LLAVA

²github.com/dvlab-research/MGM

³<https://github.com/QwenLM/Qwen-VL>

as the original Video-LLaVA code base⁴, as it is publicly available.

F. Efficiency Details

We present a comparative efficiency analysis of SparseVLM, the baseline, and FastV during the inference phase in Table 1. In this section, we provide additional details on the CUDA time during the inference phase. Following VoCo-LLaMA (Ye et al., 2025), we primarily consider the following components that contribute to the reported CUDA time: image encoding time (if applicable), KV cache load time (if applicable), and transformers forward time. We exclude other computational times that are not dependent on the model itself and the caching strategy, such as model loading time, from the CUDA time measurement. Specifically, the attention operation is implemented by FlashAttention (Dao et al., 2022).

G. More Detailed Efficiency Analysis

To better validate the efficiency of our method, we provide the latency-vs.-accuracy and FLOPs-vs.-Accuracy trade-offs for SparseVLM applied to LLaVA and MGM across three benchmarks: POPE, TextVQA, and MME, which are shown in Figure 8 and Figure 9. Besides, we also analyze Video-LLaVA matched with SparseVLM in Figure 10 on TGIF and MSVD.

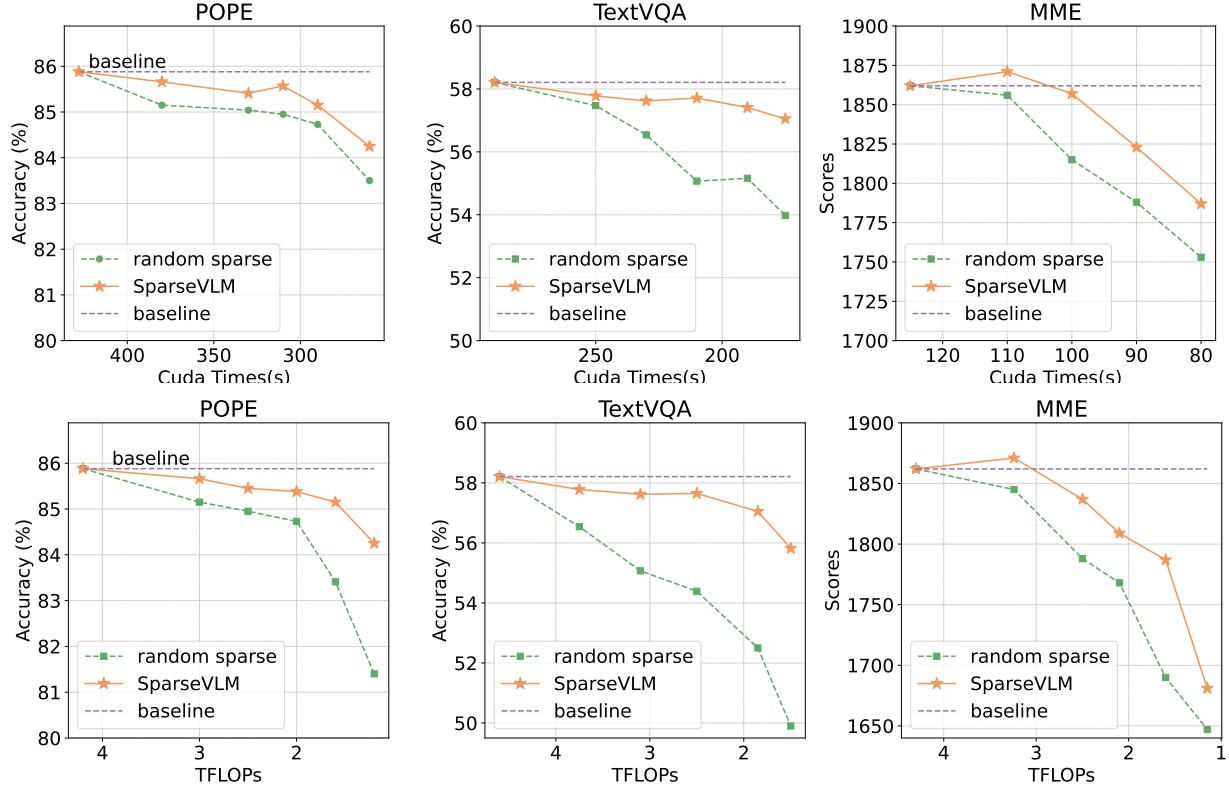


Figure 8. Trade-offs for SparseVLM on LLaVA: (a) Latency vs. Accuracy, and (b) FLOPs vs. Accuracy. Both studies demonstrate comparisons among random sparse, our SparseVLM, and baseline models.

H. More Sparsification Visualization

Figure 11 showcases a diverse array of visualization examples that demonstrate the application of SparseVLM across a spectrum of visual question-answering (VQA) prompts. These visualizations offer a deeper insight into how our SparseVLM processes and responds to different types of queries posed in a visual context.

⁴github.com/PKU-YuanGroup/Video-LLaVA.

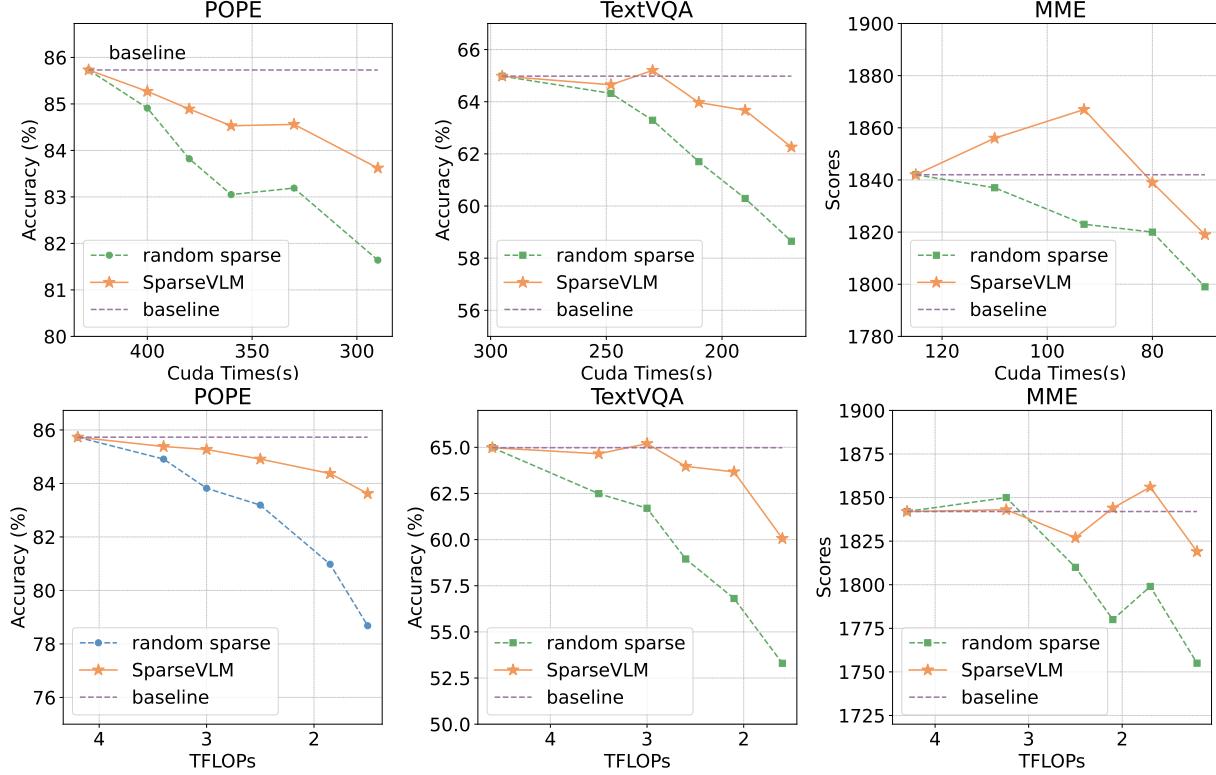


Figure 9. Trade-offs performance for SparseVLM on MGM: (a) Latency vs. Accuracy, and (b) FLOPs vs. Accuracy. Both studies demonstrate comparisons among random sparse, our SparseVLM, and baseline models.

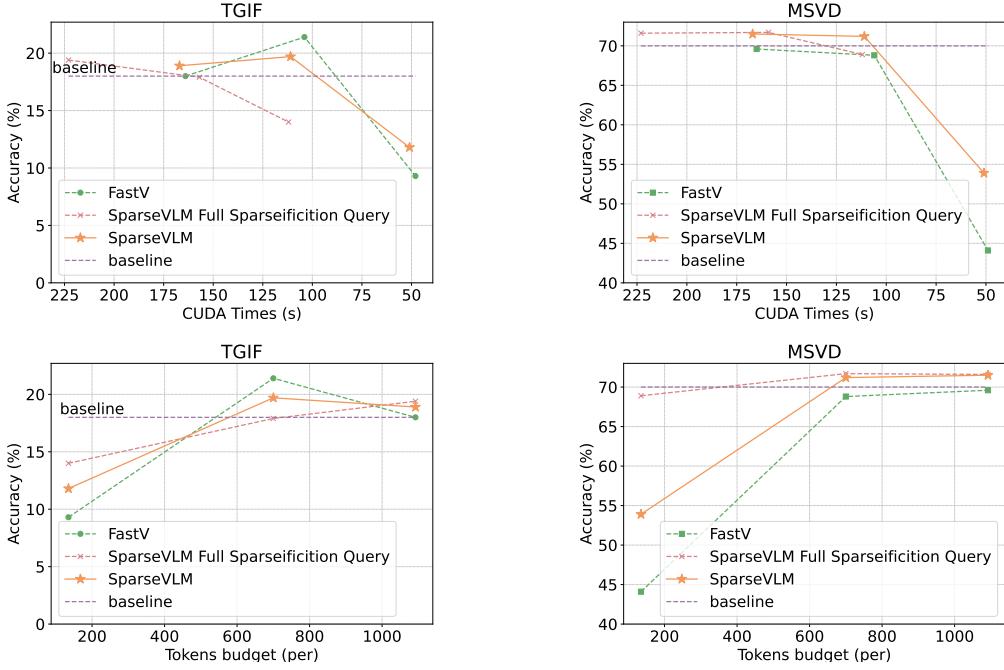


Figure 10. Trade-offs for SparseVLM on Video-LLaVA: (a) Latency vs. Accuracy, and (b) Token budget vs. Accuracy. Both studies demonstrate comparisons among random sparse, our SparseVLM, and baseline models.

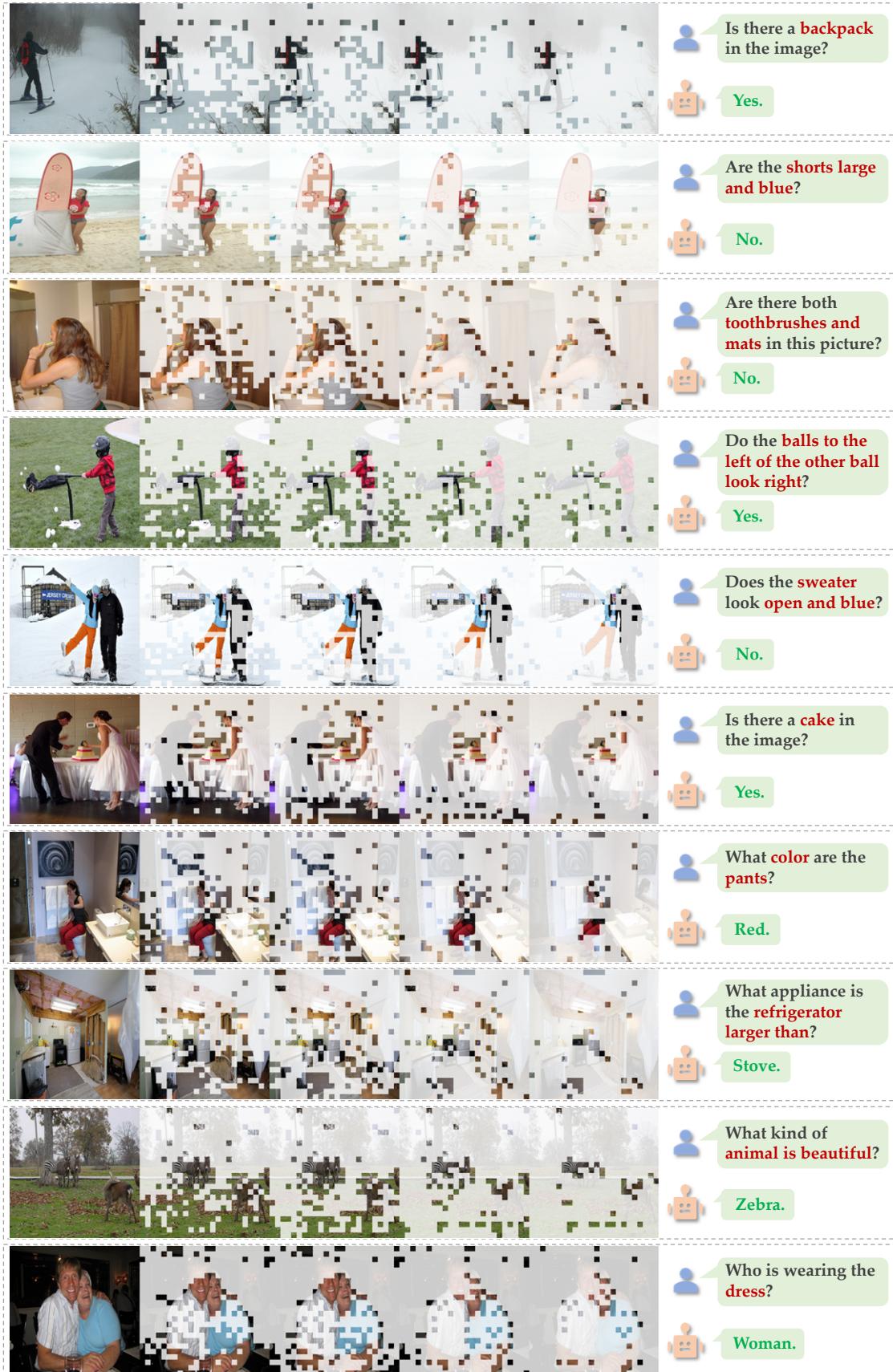


Figure 11. More visualization examples of SparseVLM on different prompts. Best viewed in color.