SwiftVLM: Efficient Vision-Language Model Inference via Cross-Layer Token Bypass

Chen Qian[*1], **Xinran Yu**[*1], **Xinan Wang**[†1], **Danyang Li**[1], **Guoxuan Chi**[2], **Zheng Yang**[†1], **Qiang Ma**[†3], **Xin Miao**[1]

[1]School of Software and BNRist, Tsinghua University; [2]School of Artificial Intelligence, Beijing University of Posts and Telecommunications; [3]QiYuan Lab, Beijing, China.

chen.cronus.qian@gmail.com wxa25@mails.tsinghua.edu.cn
hmilyyz@gmail.com maqiang@qiyuanlab.com

Abstract

Visual token pruning is a promising approach for reducing the computational cost of vision–language models (VLMs), and existing methods often rely on early pruning decisions to improve efficiency. While effective on coarse-grained reasoning tasks, they suffer from significant performance degradation on tasks requiring fine-grained visual details. Through layer-wise analysis, we reveal substantial discrepancies in visual token importance across layers, showing that tokens deemed unimportant at shallow layers can later become highly relevant for text-conditioned reasoning. To avoid irreversible critical information loss caused by premature pruning, we introduce a new pruning paradigm, termed bypass, which preserves unselected visual tokens and forwards them to subsequent pruning stages for re-evaluation. Building on this paradigm, we propose SwiftVLM, a simple and training-free method that performs pruning at model-specific layers with strong visual token selection capability, while enabling independent pruning decisions across layers. Experiments across multiple VLMs and benchmarks demonstrate that SwiftVLM consistently outperforms existing pruning strategies, achieving superior accuracy–efficiency trade-offs and more faithful visual token selection behavior. The code is released at: https://github.com/qianchen214/SwiftVLM.

1 Introduction

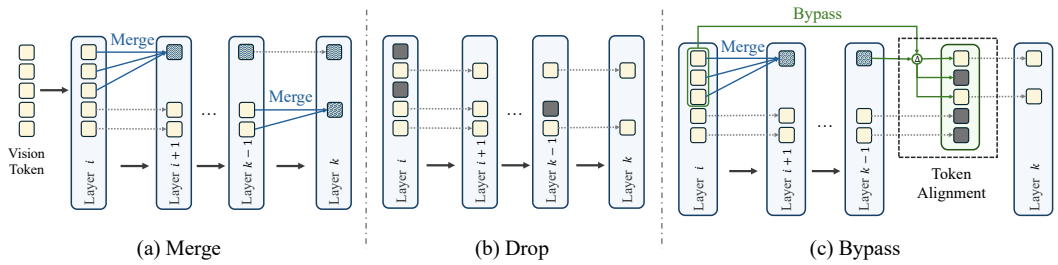
Vision–Language Models (VLMs) [1, 2, 3] have rapidly advanced in recent years and emerged as a central paradigm in multimodal learning. These models integrate a visual encoder with a large language model (LLM) [4, 5] through a cross-modal fusion module, enabling strong performance across a wide range of vision–language tasks [6, 7, 8, 9]. In practice, visual inputs are processed by generating a large number of visual tokens. However, only a small subset of these tokens is critical for text-conditioned reasoning, with the remainder largely increasing latency and computational overhead.

To reduce the number of visual tokens, prior studies adopt token merging strategies, such as ToMe [10], Qwen-VL [11], and VisionZip [12]. These methods aggregate visual features based on feature similarity or spatial proximity. While these approaches improve inference efficiency, such compression degrades fine-grained visual details, especially for precise localization tasks.

Another line of work leverages text-to-vision (T–V) attention in VLMs to rank visual tokens and dynamically drop low-ranked ones, as illustrated in Fig. 1(b). FastV [13] observes that T–V attention becomes highly concentrated on a small subset of visual tokens from the third layer onward, and

[*]Co-first authors, [†] Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Merge (b) Drop (c) Bypass

Figure 1: **Comparison of visual token pruning strategies in VLMs.** (a)–(b) Existing approaches suffer from irreversible loss of critical visual information once tokens are merged or dropped in shallow layers. (c) We propose Bypass, a pruning strategy that restores previously merged tokens via token alignment. Bypass provides critical visual tokens with an opportunity to be reconsidered at deeper layers with stronger token selection capability.

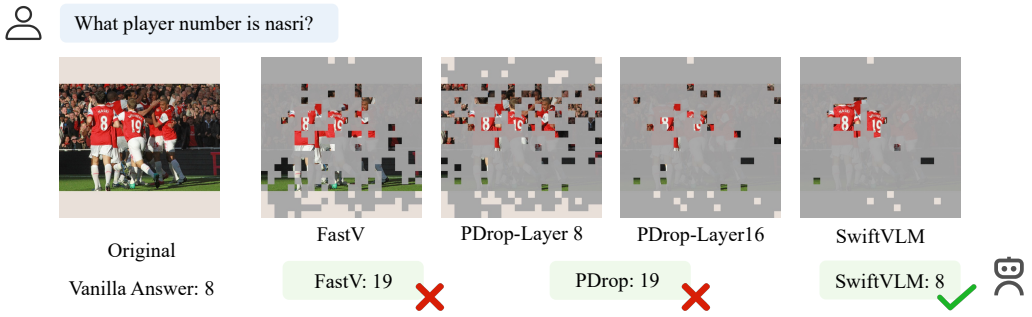


Figure 2: **Comparison of results from different pruning methods.** FastV applies aggressive early-layer pruning, whereas PDrop adopts progressive pruning. Both drop the visual token containing "NASRI", leading to incorrect answers. SwiftVLM preserves the query-relevant token at the final stage and answers correctly.

thus aggressively drops low-ranked ones in a shallow layer. PDrop [14] further shows that aggressive pruning in early layers leads to significant performance degradation, whereas the impact becomes less severe in deeper layers, motivating a progressive dropping strategy. This principle is subsequently adopted by works such as SparseVLM [15] and FEATHER [16]. However, we find that **the importance ranking of visual tokens varies across layers**.

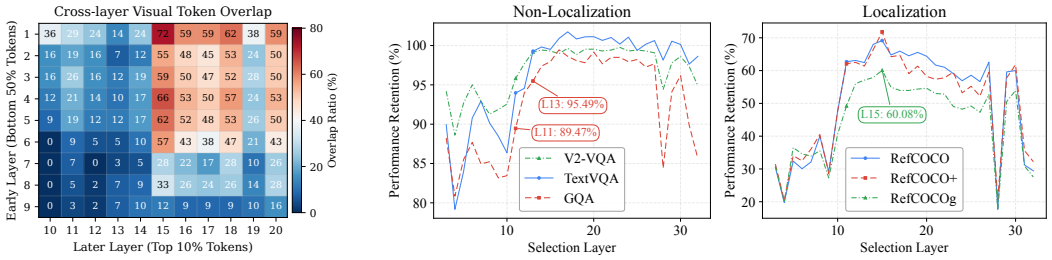
As illustrated in Fig. 3(a), we report the overlap ratio on a TextVQA [17] sample between the bottom 50% visual tokens selected by early layers (layers 1–9) and the top 10% visual tokens selected by later layers (layers 10–20) of LLaVA-1.5-7B [18]. We observe that visual tokens deemed unimportant and dropped in early layers can become highly important in deeper layers.

While existing methods perform early-layer pruning to improve efficiency, prematurely dropping task-relevant visual tokens can hinder subsequent reasoning. As shown in Fig. 2, methods such as FastV and PDrop force deeper layers to reason over incomplete visual evidence, often resulting in incorrect answers.

Based on these observations, we propose a third pruning paradigm, termed **bypass**. As illustrated in Fig. 1(c), at the first pruning layer, bottom-ranked visual tokens are not immediately discarded. Instead, they are fully preserved and forwarded directly to the next pruning layer for re-ranking of their importance. Meanwhile, these bottom visual tokens are merged according to feature similarity. The merged visual tokens then participate in subsequent inference.

At the following pruning layer, we derive a hidden-state offset from the merged visual tokens and use it to adjust the bypassed bottom-ranked tokens, aligning them with text tokens in the current representation space. These corrected tokens are then reintroduced for joint re-evaluation.

This design preserves the complete visual information while allowing each pruning layer to independently assess token importance, thereby avoiding irreversible critical information loss caused by premature pruning in early layers.



(a) (b)

Figure 3: **Two key observations on layer-wise visual token selection.** (a) **Layer-wise variation in visual token ranking.** For a representative TextVQA example, we report the overlap ratio between the bottom-ranked 50% of visual tokens selected at layers 1–9 and the top-ranked 10% selected at layers 10–20 of LLaVA. (b) **Non-monotonic layer-wise capability for visual token selection.** Across tasks and datasets, we record the layer-wise top 20% visual tokens of the vanilla model and re-evaluate it by retaining all tokens in layers 1–2 and only the layer-specific top 20% from layer 3 onward. Performance is reported relative to the vanilla baseline.

Furthermore, to determine the pruning layers used for token selection, we conduct a comprehensive layer-wise analysis across two task categories and six benchmark datasets. We first run the vanilla model and record, at each layer, the indices of the top 20% visual tokens selected based on T–V attention. Using the same set of evaluation samples, we then re-run the model while retaining all visual tokens in the first two layers and keeping only the layer-specific top 20% visual tokens from the third layer onward. The layer-wise results are reported in Fig. 3(b).

The results indicate that the ability to identify important visual tokens varies across layers and is **not monotonically increasing with depth**. Moreover, intermediate layers generally exhibiting stronger selection capability. Accordingly, we formulate the pruning-layer selection problem as a dynamic programming task, enforcing a monotonic increase in selection capability across the chosen pruning layers.

Based on these two observations, we propose **SwiftVLM**, a training-free method that performs pruning at layers with strong selection capability while ensuring independent pruning decisions at each stage.

We first identify model-specific optimal pruning layers (e.g., i and k in Fig. 1(c)) and fix them for evaluation at test time. After visual token pruning at layer i, the unselected visual tokens are preserved and re-evaluated at layer k with high selection capability.

The key contributions are summarized as follows:

- We reveal pronounced layer-wise disparities in visual token importance and propose **bypass**, a novel pruning strategy that forwards unselected visual tokens to subsequent pruning layers, enabling independent selection decisions.
- We reveal that the discriminative capability of layers for identifying critical visual tokens varies significantly across depth, exhibiting non-monotonic behavior.
- We present **SwiftVLM**, a simple yet effective training-free method that identifies high-discriminability pruning layers via dynamic programming and employs bypass to preserve fine-grained visual details while accelerating inference.
- Extensive experiments across two VLMs on nine benchmarks show SwiftVLM substantially outperforms existing training-free methods.

2 Related Work

To reduce the number of visual tokens and improve inference efficiency, existing studies [19, 20, 21] can be broadly classified into two categories.

Text-agnostic. Qwen2.5-VL [11] merges every four neighboring visual tokens into one, while ToMe [10] performs similarity-based merging. VisionZip [12] retains tokens with high [CLS]-attention scores and merges the rest by feature similarity, similar to VisPruner [22] and Prumerge [23].

However, these methods are driven solely by visual cues, making it difficult to preserve query-relevant details when the target regions are not visually salient.

Some training-based studies [24, 25] reduce the computational cost of VLMs by skipping selected modules. The key distinction between skipping and bypass lies in the explicit representation alignment mechanism introduced for low-ranked visual tokens, which facilitates their effective reintegration into subsequent layers. The empirical results indicate that direct skipping in a training-free manner leads to a noticeable performance degradation, as reported in the Appendix. A.3.

Text-aware. Q-Former [26] trains cross-modal modules to compress visual tokens into learnable tokens, while ATP-LLaVA [27] introduces trainable VLM modules to prune tokens using importance scores from text-vision and vision-vision attention. Although both leverage text queries for visual compression, they require additional trainable components, introducing extra optimization overhead.

Several training-free methods exploit the native cross-modal attention of VLMs. FastV [13] uses T-V attention to assess visual token importance and performs aggressive pruning at a shallow layer. PDrop [14] progressively reduces visual tokens across layers, based on the observation that pruning becomes less harmful at deeper layers. FEATHER [16] further refines this strategy by mitigating the influence of Rotary Position Embedding (RoPE) [28] on T-V attention, while SparseVLM [15] performs adaptive layer-wise pruning by estimating redundancy from the rank of the T-V attention matrix. Despite being training-free, these methods assume that tokens pruned early remain unimportant in deeper layers, which often fails in fine-grained visual reasoning, leading to performance degradation.

3 Method

3.1 Preliminary: Attention in VLMs

Let L denote the total number of tokens participating in computation. Let $h \in \mathbb{R}^{L\times d}$ denote the hidden states of all tokens. The query and key matrices are obtained via linear projections,

$$\mathbf{Q} = \mathbf{h}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{h}\mathbf{W}_K. \tag{1}$$

A single-head attention matrix $A \in \mathbb{R}^{L\times L}$ in a VLM is then defined as

$$A = \mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right). \tag{2}$$

VLMs adopt causal attention, under which each token is restricted to attending only to preceding tokens. As a result, the last text token attends to all input tokens. In practice, we extract its attention scores as the cross-modal component to evaluate the importance of visual tokens.

3.2 Pruning Layer Selection

In this section, we focus on how to accurately select pruning layers with high discriminative capability. Note that we exclude the first two layers from our analysis, as these layers exhibit distinct characteristics compared to other layers [29, 30].

For a model with L layers, we first record the top $V\%$ visual tokens selected by T–V attention at each layer using the vanilla model. Keeping the text and image inputs unchanged, we then re-evaluate the model by retaining all tokens in the first two layers and only the layer-specific top $V\%$ visual tokens from the third layer onward, producing a layer-wise performance profile. This performance sequence reflects the ability of each layer to identify task-relevant visual tokens. We formulate this as:

$$\{x_i\}_{i=1}^{L}, \quad x_i \in \mathbb{R}. \tag{3}$$

Intuitively, the progressively selected pruning layers should exhibit monotonically increasing performance in this sequence. Let the maximum performance before layer i be denoted as:

$$M_i = \max_{j<i} x_j. \tag{4}$$

Based on the condition $x_i > M_i$, we can identify multiple candidate sets S of pruning layers.

$$S = \{i_1, i_2, ..., i_k\}, \quad 3 \le i_1, ... \le i_k \le L. \tag{5}$$

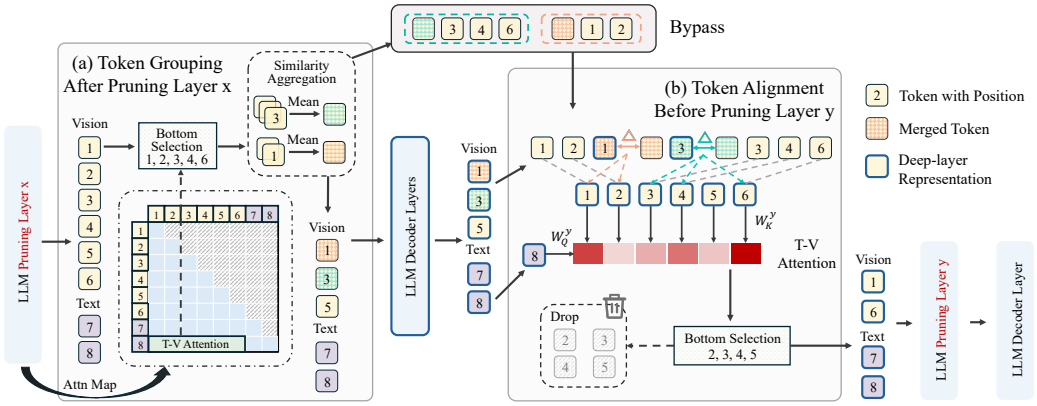


Figure 4: **SwiftVLM architecture overview.** (a) After layer x, unselected visual tokens are grouped for bypassing, with the resulting merged tokens participating in subsequent computation. (b) Before layer y, token alignment is applied to restore grouped tokens, enabling re-evaluation of visual tokens at layers with stronger token selection capability.

As the impact of visual token selection propagates through subsequent layers, we reformulate layer selection as an optimization problem that maximizes the overall layer contribution under a fixed budget of m pruning layers. Given a selection $s = \{i_1, \ldots, i_m\}$ with $i_1 < \cdots < i_m$, let $i_0 = 2$ and $i_{m+1} = L$. Then the model performance is formulated as:

$$P(s) = \frac{\sum_{k=0}^{m} x_{i_k} \, (i_{k+1} - i_k)}{L - 2}. \tag{6}$$

We denote the numerator of Eq. (6) by $U(s)$. The dynamic programming state $U_k(i)$ is defined as the maximum area accumulated from layer 2 to layer i over all selections of exactly k pruning layers, with the last pruning layer fixed at i. The state transition is

$$U_k(i) = \max_{2 \le h < i} \left[U_{k-1}(h) + x_h (i - h) \right], \quad k = 1, \ldots, m, \;\; 2 < i < L, \tag{7}$$

where h denotes the preceding pruning layer. Including the final interval ending at layer L, the optimal objective is

$$U^\star = \max_{2 < i < L} \left[U_m(i) + x_i (L - i) \right]. \tag{8}$$

The optimal pruning layers are recovered by backtracking the maximizing predecessors.

We evaluate the generality of the method on four models in Sec.4. For LLaVA-1.5-7B, we conduct layer selection experiments across six datasets, each evaluated on 1000 randomly sampled training instances, as illustrated in Fig. 3(b). Despite dataset-specific variations, the results show consistent trends: early layers exhibit noticeable fluctuations, while performance peaks around layer 15, suggesting shared characteristics in layer-wise token discriminability.

After normalizing and averaging performance metrics across datasets, we obtain $\{x_i\}_{i=1}^{L}$. Following the layer selection protocol, layers 3, 11, and 15 are selected as pruning layers for LLaVA-1.5-7B and are fixed for all test samples.

3.3 Architecture

For each model, we first select a set of pruning layers, denoted as layers x and y in Fig. 4.

The first pruning operation is performed after layer x. Based on the attention map produced by layer x, we extract the T–V attention scores between the last text token and all visual tokens. The top-ranked visual tokens are retained and directly propagated to layer $x + 1$ for further inference. The remaining low-ranked visual tokens are grouped according to the similarity between their hidden states, measured by

$$s_{i,j} = \frac{(\mathbf{h}_i^x)^\top \mathbf{h}_j^x}{|\mathbf{h}_i^x||\mathbf{h}_j^x|}, \tag{9}$$

where $\mathbf{h}_i$ and $\mathbf{h}_j$ denote the hidden states of visual tokens i and j, respectively. Visual tokens within the same group are then merged by averaging their hidden states across feature dimensions, yielding a single merged token

$$\tilde{\mathbf{h}}_{gm}^{x} = \tilde{\mathbf{h}}_{g}^{x} = \frac{1}{|\mathcal{G}_g|} \sum_{i \in \mathcal{G}_g} \mathbf{h}_i^x, \tag{10}$$

which participates in the computation of layer $x+1$.

Here, we propose a new pruning strategy termed **bypass**. Instead of permanently discarding unselected visual tokens, bypass preserves these tokens and forwards them through a side pathway to the next pruning layer, where they re-participate in the pruning selection process.

Before the pruning layer y, we re-evaluate the importance of all visual tokens. For each group formed by merged tokens, we estimate the average offset of the group as

$$\Delta \mathbf{h}_{gm} = \tilde{\mathbf{h}}_{gm}^{y-1} - \tilde{\mathbf{h}}_{gm}^{x}. \tag{11}$$

To align the visual tokens transmitted through the bypass pathway with the deeper representations of other tokens, we correct each visual token in group g as follows:

$$\hat{\mathbf{h}}_i^{y-1} = \mathbf{h}_i^x + \Delta \mathbf{h}_{gm}, \qquad i \in \mathcal{G}_g. \tag{12}$$

Before pruning layer y, we project the aligned visual tokens with W_K^y and the last text token from layer $y-1$ with W_Q^y to recompute T-V attention. Only the selected important visual tokens are retained for subsequent prefill computation.

3.4 Representation Alignment Analysis

Transformer [31] layers adopt a residual formulation, where the hidden states are updated as

$$\mathbf{h}^{\ell} = \mathbf{h}^{\ell-1} + \mathcal{F}^{\ell}(\mathbf{h}^{\ell-1}), \tag{13}$$

with $\mathcal{F}^{\ell}(\cdot)$ denoting the combined attention and feed-forward transformation at layer ℓ.

For a visual token i belonging to group $\mathcal{G}_g$, its hidden state in the vanilla model evolves from layer $x+1$ to layer $y-1$ as

$$\mathbf{h}_i^{y-1} = \mathbf{h}_i^x + \sum_{\ell=x+1}^{y-1} \mathcal{F}^{\ell}(\mathbf{h}_i^{\ell-1}). \tag{14}$$

Taking the average over all tokens in group $\mathcal{G}_g$, we obtain

$$\tilde{\mathbf{h}}_g^{y-1} = \tilde{\mathbf{h}}_g^{x} + \sum_{\ell=x+1}^{y-1} \frac{1}{|\mathcal{G}_g|} \sum_{i \in \mathcal{G}_g} \mathcal{F}^{\ell}(\mathbf{h}_i^{\ell-1}), \tag{15}$$

We denote by $\Delta \mathbf{h}_g$ the accumulated group-level residual update.

In Sec. 4.3, we obtain $\Delta \mathbf{h}_g$ from the vanilla model and compare it with $\Delta \mathbf{h}_{gm}$. Under fine-grained grouping, their low-dimensional projections show near-complete overlap, providing empirical support for the proposed offset-based approximation. We further discuss the boundary cases under extreme settings in the Appendix.A.4.

3.5 FLOPs Computation

We consider pruning after the K-th VLM layer, where a fraction $D\%$ of visual tokens is removed. Let n_v and n_t denote the numbers of visual and non-visual tokens, respectively, with $n = n_v + n_t$. After pruning, the token count becomes $\hat{n} = (1 - D\%)n_v + n_t$. Given T layers, hidden dimension d, and FFN intermediate dimension m, the resulting FLOPs F are:

$$C_n = \big(4nd^2 + 2n^2d + 3ndm\big), \tag{16}$$

$$F = K \times C_n + (T - K) \times C_{\hat{n}}. \tag{17}$$

Table 1: **Performance comparison under different visual token budgets.** (+) and (g) denote RefCOCO+ and RefCOCOg, respectively.

Method	Localization				Non-localization							FLOPs (T)
	RefCOCO	(+)	(g)	Avg.	VQA^{Text}	GQA	SQA	MME	MMB	POPE	Avg.	
Upper Bound, 576 Tokens (100%)												
Vanilla	75.9 100%	67.0 100%	70.7 100%	100%	46.9 100%	61.4 100%	69.6 100%	1509 100%	64.6 100%	86.8 100%	100%	4.29
Retain 192 Tokens (↓ 66.7%)												
FastV *(ECCV'24)*	30.6 40.3%	25.8 38.5%	29.7 42.0%	40.3%	43.6 93.0%	57.2 93.2%	69.4 99.7%	1471 97.5%	63.2 97.8%	82.0 94.5%	95.9%	1.71
VisionZip *(CVPR'25)*	7.0 9.2%	5.7 8.5%	6.3 8.9%	8.9%	45.2 96.4%	58.9 95.9%	68.8 98.9%	1460 96.8%	62.9 97.4%	86.6 99.8%	97.5%	1.71
PDrop *(CVPR'25)*	22.2 29.2%	18.2 27.2%	18.7 26.4%	27.6%	42.9 91.5%	55.5 90.4%	69.2 99.4%	1365 90.5%	63.2 97.8%	81.1 93.4%	93.8%	1.72
SparseVLM *(ICML'25)*	8.7 11.5%	7.5 11.2%	7.1 10.0%	10.9%	**45.8** **97.7%**	58.9 95.9%	69.1 99.3%	1447 95.9%	64.2 99.4%	86.7 99.9	98.0%	1.72
FEATHER *(ICCV'25)*	52.0 68.5%	45.5 67.9%	45.4 64.2%	66.9%	42.9 91.5%	58.6 95.4%	**70.5** **101.3%**	1431 94.8%	63.9 98.9%	84.4 97.2%	96.5%	1.82
SwiftVLM	**66.6** **87.7%**	**58.5** **87.3%**	**60.6** **85.7%**	**86.9%**	45.3 96.6%	**60.7** **98.9%**	69.0 99.1%	**1503** **99.6%**	**64.5** **99.8%**	**87.1** **100.3%**	**99.0%**	1.75
Retain 128 Tokens (↓ 77.8%)												
FastV *(ECCV'24)*	12.8 16.9%	11.1 16.6%	13.8 19.5%	17.7%	39.7 84.6%	53.6 87.3%	68.5 98.4%	1377 91.3%	62.3 96.4%	77.7 89.5%	91.3%	1.29
VisionZip *(CVPR'25)*	4.6 6.0%	3.6 5.4%	4.3 6.1%	5.8%	**44.4** **94.7%**	57.5 93.6%	68.9 99.0%	1441 95.5%	62.0 96.0%	85.1 98.0%	96.1%	1.29
PDrop *(CVPR'25)*	3.0 4.0%	2.3 3.4%	2.3 3.3%	3.6%	39.9 85.1%	54.3 88.4%	**70.2** **100.9%**	1322 87.6%	61.9 95.8%	80.9 93.2%	91.8%	1.28
SparseVLM *(ICML'25)*	4.8 6.3%	3.9 5.8%	4.1 5.8%	6.0%	42.0 89.6%	57.4 93.5%	69.8 100.3%	1418 94.0%	63.6 98.5%	86.0 99.1%	95.8%	1.30
FEATHER *(ICCV'25)*	39.0 51.4%	34.3 51.2%	35.2 49.8%	50.8%	41.2 87.8%	56.5 92.0%	69.6 100%	1453 96.3%	63.2 97.8%	83.3 96.0%	95.0%	1.44
SwiftVLM	**55.2** **72.7%**	**46.6** **69.6%**	**47.4** **67.0%**	**69.8%**	41.8 89.1%	**59.2** **96.4%**	68.5 98.4%	**1477** **97.9%**	**63.9** **98.9%**	**86.1** **99.2%**	**96.7%**	1.31

We further analyze the overhead introduced by our operations. Let R and Z denote the numbers of low-ranked and merged visual tokens, respectively. The merge and representation alignment steps incur costs of $2RZd$ and Rd. Query projection for the last text token costs $2d^2$, while key projection for visual tokens and the following dot products cost $2n_vd^2$ and $2n_vd$, respectively. Let r denote the ratio of visual tokens retained at layer y. The overall overhead F_o is given by:

$$\mathrm{F_o} = 2RZd + Rd + 2n_vd + 2d^2 + 2(1-r)n_vd^2. \tag{18}$$

4 Experiments

4.1 Overall Performance

Datasets. We categorize inference tasks into localization and non-localization types, which respectively emphasize fine-grained visual details and holistic information integration. The proposed method is evaluated on nine widely used benchmarks: RefCOCO, RefCOCO+, RefCOCOg [32, 33], TextVQA, GQA [34], SQA [35], MME [10], MMB [36], and POPE [37]. For TextVQA, we follow prior work [16] and exclude OCR prompts to isolate the effect of pruning on visual understanding.

Main Results. Given that RefCOCO bounding boxes cover about 102 visual tokens on average, Tab. 1 compares different methods on LLaVA-1.5-7B under 192-token and 128-token budgets. On non-localization tasks, all methods perform competitively.

In contrast, localization tasks exhibit much larger performance gaps. PDrop and SparseVLM reorder retained visual tokens after pruning, disrupting positional information and causing substantial

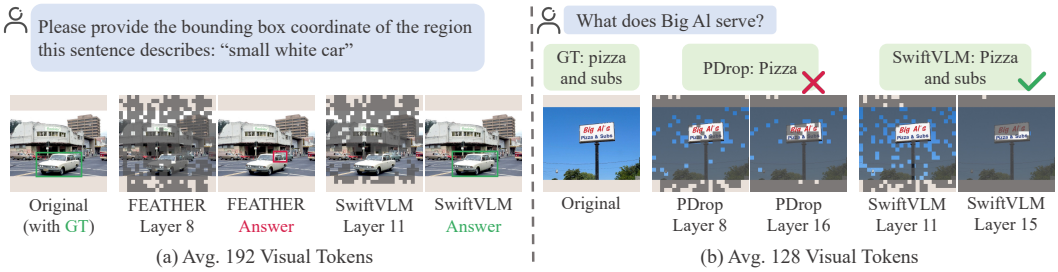


Figure 5: **Visualization of method performance under varying tasks and computation budgets.**

Table 2: **Ablation study.** X_S denotes layer selection. X_M denotes token merging, and X_A denotes the alignment mechanism.

Tokens	Method	RefCOCO	VQA^{Text}
192	Baseline	42.6	43.2
	+ X_S	64.5	45.3
	+ X_S + X_M	63.7	44.8
	+ X_S + X_M + X_A	**66.6**	**45.3**
128	Baseline	23.2	41.2
	+ X_S	42.8	40.1
	+ X_S + X_M	51.9	40.7
	+ X_S + X_M + X_A	**55.2**	**41.8**

Table 3: **Performance comparison on LLaVA-NeXT-7B.**

Method	RefCOCO	VQA^{Text}	GQA	MMB	Rel. Acc
Upper Bound, Retain 100% Tokens					
Vanilla	85.3	65.5	63.9	67.9	100%
Retain 33.3% Tokens					
FastV	40.5	58.7	59.0	48.3	75.1%
FEATHER	68.8	62.6	62.5	67.5	92.8%
SwiftVLM	**80.7**	**64.1**	**63.6**	**68.0**	**98.0%**
Retain 22.2% Tokens					
FastV	26.1	52.6	56.9	46.0	66.9%
FEATHER	53.1	60.9	61.9	66.5	87.5%
SwiftVLM	**79.6**	**62.4**	**63.5**	**67.7**	**97.1%**

degradation [38]. FEATHER reduces RoPE-induced bias by recomputing attention, but at the cost of higher FLOPs. More importantly, token discriminability across layers remains non-monotonic even without RoPE effects (Appendix D), and low-ranked visual tokens are still irreversibly discarded after initial pruning. As a result, FEATHER lags behind SwiftVLM by about 20%.

Visualization. We visualize examples from RefCOCO and TextVQA, showing the retained visual tokens as image patches along with the final answers. As illustrated in Fig. 5, FEATHER and PDrop adopt drop-based pruning and discard task-relevant visual tokens (e.g., the car in localization and the signboard in VQA), leading to incomplete or incorrect answers. More comparison in Appendix. E.

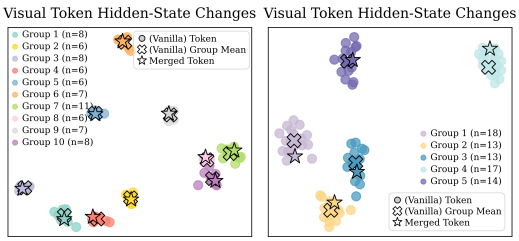
4.2 Ablation Study

The baseline adopts the same uniform layer selection strategy and pruning ratios as PDrop, while preserving the original positional information of retained visual tokens. In Tab. 2, we progressively introduce DP-based layer selection, token merging, and alignment, with all variants within each row evaluated under the same computational budget.

Under the 192-token setting, layers with monotonically increasing selection capability yield the largest gains, while token merging introduces unnecessary compression under a sufficient token budget. Under the more constrained 128-token setting, token merging becomes beneficial by alleviating the information loss caused by aggressive dropping. Overall, bypass consistently improves pruning performance across different budget settings.

4.3 Why Bypass Works?

To explain why bypassed visual tokens remain effective after representation alignment, we visualize the low-dimensional projections of token offsets defined in Sec. 3.4. Under the 128-token setting, Fig. 6(a) shows a TextVQA example, where Merged Token denotes Δh_{gm}. For each bypassed group, we run the vanilla model to obtain the actual hidden-state changes of individual tokens after layer 10, denoted as Vanilla Token, and their average change, denoted as Vanilla Group Mean. We observe that the vanilla group mean closely overlaps with the merged token offset and remains highly consistent with the changes of individual tokens within the group. We then substantially reduce the number of merged tokens and report the results for the same example in Fig. 6(b).



(a) Fine-grained Token Merging (b) Coarse-grained Token Merging

Figure 6: **t-SNE visualization of visual token hidden-state changes.** Colors denote similarity-based token groups. In the vanilla model, • shows per-token changes and × shows the group-wise mean. In our method, each group is merged into a single token, its change from layer 3 to layer 10 is shown as a ⋆. At $n = 18$, merged tokens account for less than 5%.

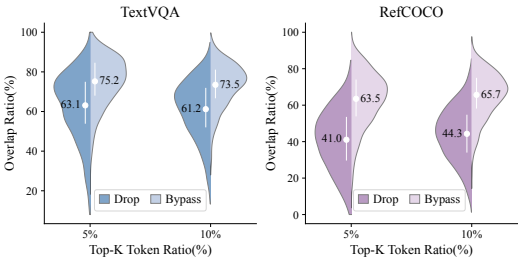


Figure 7: **Token selection overlap with vanilla for drop and bypass.** Under an equal computational budget, the overlap distribution and mean are reported over 4000 cases by comparing the tokens selected at layer 15 under different pruning schemes with those selected by the vanilla model, in order to assess their impact on intrinsic selection behavior.

Table 4: **Performance comparison on Qwen3-VL-2B-Instruct.**

Method	RefCOCO	VQA^{Text}	GQA
Upper Bound, Retain 100% Tokens			
Vanilla	84.9	78.9	58.2
	100%	100%	100%
Retain 33% Tokens			
FastV	57.1	70.3	49.5
	67.3%	89.1%	85.1%
FEATHER	60.7	74.6	54.7
	71.5%	94.6%	94.0%
SwiftVLM	70.0	74.4	55.2
	82.4%	94.3%	94.8%

Table 5: **Results of SwiftVLM on Video-LLaVA across video question-answering benchmarks.**

Method	MSRVTT		MSVD	
	Acc.	Score	Acc.	Score
Upper Bound, Retain 100% Tokens				
Vanilla	55.0	2.93	66.5	3.41
	100%	100%	100%	100%
Retain 33% Tokens				
FastV	53.2	2.85	63.3	3.26
	96.7%	97.3%	95.2%	95.6%
FEATHER	54.0	2.85	63.8	3.28
	98.2%	97.3%	95.9%	96.2%
SwiftVLM	54.7	2.87	64.9	3.31
	99.5%	98.0%	97.6%	97.1%

Given that VLMs employ causal attention, the hidden-state evolution of a visual token can actually only be influenced by preceding visual tokens. Moreover, since attention fundamentally operates through similarity-based interactions, we hypothesize that visual tokens with similar semantics exhibit similar transformation directions in the representation space, and can thus be well approximated by the changes of the corresponding merged token.

4.4 Why Is Bypass Better Than Drop?

Under the 128-token setting, we compare the visual tokens retained at layer 15 by drop and bypass with the top 5% and top 10% tokens selected by the vanilla model, and report their overlap ratios on TextVQA and RefCOCO in Fig. 7. Bypass exhibits a higher overlap with the vanilla model, indicating its ability to preserve visual tokens that are critical for reasoning. This overlap gap is more pronounced on RefCOCO, consistent with the larger performance differences observed across datasets under the 128-token setting in the ablation study.

4.5 Generalization

Following prior work, we evaluate generalization on LLaVA-NeXT [39] across four datasets, as reported in Tab. 3. Since LLaVA-NeXT removes image padding, we compare methods by visual token retention ratio. SwiftVLM consistently outperforms existing methods, especially on localization datasets. Additional results on Qwen3-VL [40] are provided in Tab. 4.

SwiftVLM is further implemented on Video-LLaVA [41], with results reported in Tab. 5. Following FastV, evaluation is conducted on the first 1000 samples of MSRVTT-QA and MSVD-QA [42].

Qwen3.5-Plus [43] serves as the evaluation model. The results demonstrate that SwiftVLM effectively generalizes to video understanding tasks.

5 Conclusion

In this work, we revisit visual token pruning in VLMs and reveal that visual token importance varies substantially across layers. This observation explains why existing drop-based pruning methods, which rely on early selection decisions, often struggle on tasks requiring fine-grained visual reasoning. To better preserve visual information, we introduce a novel pruning strategy, termed bypass, and integrate it into our proposed pruning framework, SwiftVLM. This design allows each pruning layer to perform token selection in a relatively independent manner. Experimental results demonstrate that bypass consistently outperforms drop, suggesting its potential as a promising pruning paradigm.

Acknowledgments and Disclosure of Funding

We sincerely thank the anonymous reviewers for their valuable comments. This work is supported in part by the NSFC under grant No. 62372265, No. 62332016, No. U24B20152, and No. 62602356, and in part by the China Postdoctoral Science Foundation under Grant Number 2025M781491, and GZB20260387.

References

[1] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[2] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. 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, 2024.

[3] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

[4] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[5] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[6] Chongkai Gao, Zixuan Liu, Zhenghao Chi, Junshan Huang, Xin Fei, Yiwen Hou, Yuxuan Zhang, Yudi Lin, Zhirui Fang, Zeyu Jiang, et al. Vla-os: Structuring and dissecting planning representations and paradigms in vision-language-action models. *arXiv preprint arXiv:2506.17561*, 2025.

[7] Tianwei Lin, Wenqiao Zhang, Sijing Li, Yuqian Yuan, Binhe Yu, Haoyuan Li, Wanggui He, Hao Jiang, Mengze Li, Xiaohui Song, et al. Healthgpt: A medical large vision-language model for unifying comprehension and generation via heterogeneous knowledge adaptation. *arXiv preprint arXiv:2502.09838*, 2025.

[8] Qi Yang, Chenghao Zhang, Lubin Fan, Kun Ding, Jieping Ye, and Shiming Xiang. Re-ranking reasoning context with tree search makes large vision-language models stronger. *arXiv preprint arXiv:2506.07785*, 2025.

[9] Jinbo Wang, Mingze Wang, Zhanpeng Zhou, Junchi Yan, Lei Wu, et al. The sharpness disparity principle in transformers for accelerating language model pre-training. *arXiv preprint arXiv:2502.19002*, 2025.

[10] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022.

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

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

[13] Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pages 19–35. Springer, 2024.

[14] Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, et al. Pyramiddrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. *arXiv preprint arXiv:2410.17247*, 2024.

[15] Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. Sparsevlm: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*, 2024.

[16] Mark Endo, Xiaohan Wang, and Serena Yeung-Levy. Feather the throttle: Revisiting visual token pruning for vision-language model acceleration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22826–22835, 2025.

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

[18] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 26296–26306, 2024.

[19] Yiwu Zhong, Zhuoming Liu, Yin Li, and Liwei Wang. Aim: Adaptive inference of multi-modal llms via token merging and pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20180–20192, 2025.

[20] Qinsi Wang, Hancheng Ye, Ming-Yu Chung, Yudong Liu, Yueqian Lin, Martin Kuo, Mingyuan Ma, Jianyi Zhang, and Yiran Chen. Corematching: A co-adaptive sparse inference framework with token and neuron pruning for comprehensive acceleration of vision-language models. *arXiv preprint arXiv:2505.19235*, 2025.

[21] Yanwei Li, Chengyao Wang, and Jiaya Jia. Llama-vid: An image is worth 2 tokens in large language models. In *European Conference on Computer Vision*, pages 323–340. Springer, 2024.

[22] Qizhe Zhang, Aosong Cheng, Ming Lu, Renrui Zhang, Zhiyong Zhuo, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. Beyond text-visual attention: Exploiting visual cues for effective token pruning in vlms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20857–20867, 2025.

[23] Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22857–22867, 2025.

[24] Jun Zhang, Desen Meng, Zhengming Zhang, Zhenpeng Huang, Tao Wu, and Limin Wang. p-mod: Building mixture-of-depths mllms via progressive ratio decay. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3705–3715, 2025.

[25] Weili Zeng, Ziyuan Huang, Kaixiang Ji, and Yichao Yan. Skip-vision: Efficient and scalable acceleration of vision-language models via adaptive token skipping. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 21384–21397, 2025.

[26] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[27] Xubing Ye, Yukang Gan, Yixiao Ge, Xiao-Ping Zhang, and Yansong Tang. Atp-llava: Adaptive token pruning for large vision language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 24972–24982, 2025.

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

[29] Vedang Lad, Jin Hwa Lee, Wes Gurnee, and Max Tegmark. The remarkable robustness of llms: Stages of inference? *arXiv preprint arXiv:2406.19384*, 2024.

[30] Seil Kang, Jinyeong Kim, Junhyeok Kim, and Seong Jae Hwang. Your large vision-language model only needs a few attention heads for visual grounding. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 9339–9350, 2025.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[32] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 787–798, 2014.

[33] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In *European conference on computer vision*, pages 69–85. Springer, 2016.

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

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

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

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

[38] Tzu-Chun Chien, Chieh-Kai Lin, Shiang-Feng Tsai, Ruei-Chi Lai, Hung-Jen Chen, and Min Sun. Grounding-aware token pruning: Recovering from drastic performance drops in visual grounding caused by pruning. *arXiv preprint arXiv:2506.21873*, 2025.

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

[40] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[41] Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 5971–5984, 2024.

[42] Dejing Xu, Zhou Zhao, Jun Xiao, Fei Wu, Hanwang Zhang, Xiangnan He, and Yueting Zhuang. Video question answering via gradually refined attention over appearance and motion. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1645–1653, 2017.

[43] Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026.

[44] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

A Additional Experiment Results

A.1 Efficiency Study

Following SparseVLM, we implement SwiftVLM in a FlashAttention-compatible [44] manner and report the corresponding latency results in Tab. 6. Compared to the vanilla model, all pruning-based methods achieve noticeable speedups. FastV attains the largest acceleration since it performs pruning only once.

Unlike FLOPs computation, FlashAttention does not provide direct access to attention maps, requiring attention scores to be recomputed in practice. Consequently, SwiftVLM incurs lower latency than SparseVLM, as it only computes attention between the final text token and visual tokens, whereas SparseVLM requires attention computation for all text tokens.

Table 6: **Efficiency study on LLaVA-1.5-7B.** Total Time denotes the wall-clock time required to process the entire POPE dataset. Prefilling Time refers to the average prefill latency per sample. Δ indicates the speedup factor relative to the vanilla model.

Tokens	Method	Total Time (s)	Δ	Prefilling Time (ms)	Δ
576	Vanilla	850.7	-	67.3	-
192	FastV	551.8	1.54×	34.7	1.92×
	SparseVLM	612.3	1.39×	40.7	1.65×
	SwiftVLM	573.8	1.48×	37.6	1.79×
128	FastV	539.4	1.58×	32.8	2.05×
	SparseVLM	583.9	1.46×	37.5	1.79×
	SwiftVLM	546.2	1.56×	33.0	2.04×

A.2 Peak GPU Memory Usage

We report the GPU resource usage of different methods in Tab. 7. PGM denotes the peak GPU memory increase during a full inference pass, measured relative to the post-loading memory footprint of the model. All experiments are conducted on an RTX 4090 GPU.

Table 7: **Peak GPU Memory Usage on LLaVA-1.5-7B.**

Tokens	Method	PGM (MB)
576	Vanilla	716.5
192	FastV	305.3
	SwiftVLM	365.1

A.3 Bypass vs. Skipping

Some studies [24, 25] reduce VLM computation by skipping selected modules, typically with additional training to preserve performance. Although bypass and direct skipping both aim to reduce computational cost, their mechanisms are fundamentally different. Bypass incorporates **explicit representation alignment** for low-ranked visual tokens, enabling their effective reintegration into subsequent layers.

The empirical results in Tab. 8 show that directly applying skipping without additional training leads to a noticeable performance degradation. In other words, skipping generally relies on additional training to maintain competitive performance. In contrast, bypass serves as a new paradigm for training-free manner, enabling computational savings during VLM inference while better preserving model performance.

Table 8: **Performance comparison between bypass and skipping under the Retain-192-Tokens setting.**

Method	RefCOCO	VQA^{Text}	POPE
Bypass	66.6	45.3	87.1
Skipping	47.3	44.7	83.3

A.4 Effect of Group Size on Performance

Visual inputs contain substantial semantic redundancy. Bypass leverages this redundancy to group visual tokens and estimates the representation offsets of group members using the offset of the corresponding merged token, thereby enabling representation alignment.

To further examine the limit of this estimation strategy, we report the results of SwiftVLM under the 128-token setting (retaining 35% visual tokens from layers 4 to 10) in Tab. 9, evaluated on 2000 samples from the TextVQA and RefCOCO training sets.

The results show a concave trend as the merge ratio decreases (i.e., as the group size increases). We consider that the best setting depends on the redundancy of the image, although the overall performance remains relatively stable within a reasonable range of merge ratios.

When the merge ratio is pushed to an extreme level, however, this setting is less aligned with the original intuition of bypass, which relies on clustering similar tokens for group-level estimation. In this case, the approximation becomes less faithful, and the resulting performance is therefore less favorable.

Nevertheless, even under the extreme 5% merge-ratio setting, Bypass still consistently outperforms the Drop baseline.

Table 9: **Effect of different group sizes on method performance under the 128-token setting on LLaVA-1.5-7B.**

Method	High-rank Token (%)	Merged Token (%)	VQA^{Text}	RefCOCO
Bypass	0	35	42.2	50.1
	5	30	43.1	53.2
	10	25	43.3	53.7
	15	20	43.9	52.5
	20	15	43.6	50.8
	25	10	43.0	48.7
	30	5	41.9	43.1
Drop	35	0	40.6	43.0

A.5 Computational overhead breakdown

Since LLaVA-NeXT supports variable-length inputs, we analyze SwiftVLM latency on LLaVA-NeXT-7B with different visual token counts under a 33.3% token retention ratio. LLaVA-NeXT-13B is used to evaluate latency across model scales. Our implementation is FlashAttention-compatible (batch size 1, mean over 50 runs after 10 warm-up runs).

(1) Theoretical complexity. Our method introduces four overhead components. As shown in Tab. 10, similarity grouping is the only superlinear term in V, but with a small constant (0.18, i.e., $\approx 9\%$ of a single layer's attention-score FLOPs). Its relative cost therefore has a constant upper bound independent of V.

(2) Measured breakdown. As shown in Tab. 11, even at the highest resolution, similarity grouping itself takes $< \mathbf{0.5\ ms}$ and the total overhead stays below 6.1 ms. As V grows $2.5\times$ ($1176 \rightarrow 2928$), the overhead's share of prefill decreases, because the compute saved by pruning grows faster than the overhead. Meanwhile, pruning reduces prefill latency by **48–61%** over the unpruned model

(7B: 153 $\rightarrow$ 80 ms at $V = 1176$ and 349 $\rightarrow$ 154 ms at $V = 2928$; 13B: 274 $\rightarrow$ 126 ms and 638 $\rightarrow$ 251 ms).

Table 10: **Computational complexity of each component.** V = number of visual tokens, D = hidden dimension.

Component	FLOPs complexity	Operation
Similarity grouping	$O(0.18 \cdot V^2 D)$	$0.6V$ grouped to $0.3V$
Token merging	$O(V \cdot D)$	scatter-based cluster reduction
Selection attention	$O(V \cdot D^2)$	key projection over the active sequence (across all pruning layers under FlashAttention)
Delta alignment	$O(V \cdot D)$	gather + residual addition

Table 11: **Runtime overhead breakdown.** All latencies are milliseconds per image. Percentages are relative to the pruned model's own prefill latency and its full inference latency, respectively.

Model	V	Grouping (ms)	Merging (ms)	Sel.-attn. (ms)	Alignment (ms)	Total (ms)	% of prefill	% of full inference
7B	1176	0.30	0.58	2.18	0.11	**3.17**	4.0%	0.6%
7B	2328	0.36	0.56	2.99	0.20	**4.12**	3.2%	0.6%
7B	2928	0.42	0.59	3.27	0.26	**4.54**	2.9%	0.6%
13B	1176	0.28	0.55	2.61	0.14	**3.58**	2.8%	0.4%
13B	2328	0.38	0.60	3.64	0.26	**4.88**	2.3%	0.5%
13B	2928	0.44	0.65	4.64	0.34	**6.07**	2.4%	0.6%

A.6 Experiments to evaluate trade-offs for grounding tasks

To investigate the trade-offs on grounding tasks, we added experiments providing a fine-grained analysis of token budget, accuracy, and computational cost.

Table 12: Accuracy, latency, and memory usage under different visual token budgets.

Avg. retained visual tokens	Acc (%)	Rel. Acc	TTFT (ms)	TTFT speedup	KV cache (MiB)	KV rel.	PGM (MB)
576 (vanilla)	76.1	100.0%	84.7	1.00×	319	100%	787
384	75.3	**99.0%**	71.2	1.19×	223	70%	620
288	71.5	94.0%	62.8	1.35×	175	55%	518
192	67.1	88.2%	54.4	1.56×	128	40%	429
128	54.9	72.1%	48.6	**1.74×**	95	**30%**	**355**

TTFT: End-to-end time-to-first-token.

PGM: Peak GPU memory increases over the post-loading footprint.

SwiftVLM is evaluated on a fixed 2000 examples sampled from the RefCOCO training split. A favorable trade-off is observed at the half-token budget, which retains 94% of the uncompressed-model accuracy while reducing the KV-cache size by 45%.

B Analysis of Task-specific Characteristics in VLMs

As mentioned in the Introduction, we evaluate model performance by retaining the top 20% visual tokens selected at different layers and using these tokens as input from the third layer onward. The resulting performance retention is shown in Fig. 3(b).

A closer analysis reveals distinct behaviors across tasks. For non-localization tasks, model performance varies within a margin of approximately 20% even when different visual tokens are selected. In contrast, localization tasks exhibit a much larger performance gap, reaching around 50%.

We further analyze the overlap ratio between the visual tokens selected at each layer and the ground-truth bounding boxes. Specifically, let V_s denote the set of visual tokens selected at a given layer, and

let V_g represent the set of visual tokens corresponding to image patches covered by the ground-truth bounding boxes. The overlap ratio between the two sets is defined as:

$$\text{Overlap}(V_s, V_g) = \frac{|V_s \cap V_g|}{|V_g|}, \tag{19}$$

and the results are shown in Fig. 8.

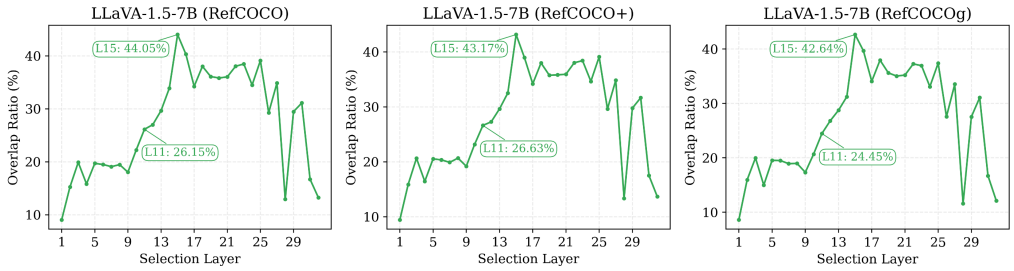


Figure 8: **Layer-wise analysis of visual localization capability.** We conduct experiments on three localization benchmarks using LLaVA. We first execute the vanilla model to record the layer-wise top 20% selected visual tokens, and then measure their overlap with the ground-truth bounding boxes by computing the overlap ratio.

We observe that model accuracy on localization tasks exhibits a highly similar trend to the overlap ratio between the selected visual tokens and the ground-truth bounding boxes. In other words, for localization tasks, selecting layers that enable accurate visual localization for token selection has a substantial impact on task performance and is therefore a critical factor that pruning methods must consider.

Prior work, such as FEATHER, has shown that for non-localization tasks, reasonably strong performance can still be achieved even when only the visual tokens corresponding to the latter half of the image are provided as input. Consequently, although different methods retain different sets of visual tokens, their performance on non-localization tasks shows relatively small differences.

Another notable distinction lies in the sensitivity of different tasks to the spatial positions of image patches associated with visual tokens. As shown in Tab. 1, methods such as VisionZip, PDrop, and SparseVLM do not preserve the original position indices of visual tokens after pruning. While they achieve impressive performance on non-localization tasks, they suffer from severe performance degradation on localization tasks.

Taking PDrop as an example, we introduce the preservation of position indices in an ablation study. As a result, its performance on the RefCOCO dataset under the 128-token setting improves dramatically, rising from 3% to 23.2%.

C Attention with RoPE in VLM

RoPE-induced Early-layer Selection Bias in VLMs. Consider a VLM that employs RoPE in its attention mechanism. Let $\mathbf{q}_t^{(l)}$ denote the query vector of a text token t at layer l, and $\mathbf{k}_i^{(l)}$ denote the key vector of a visual token with position index i. The query and key are given by

$$\tilde{\mathbf{q}}_t^{(l)} = R(\theta_t)\mathbf{q}_t^{(l)}, \quad \tilde{\mathbf{k}}_i^{(l)} = R(\theta_i)\mathbf{k}_i^{(l)}, \tag{20}$$

where θ_i is determined by the token position.

Then, the attention score between t and i at layer l satisfies

$$\tilde{\mathbf{q}}_t^{(l)\top}\tilde{\mathbf{k}}_i^{(l)} = \mathbf{q}_t^{(l)\top}\mathbf{k}_i^{(l)}\cos(\theta_t - \theta_i) + \mathbf{q}_t^{(l)\top}J\mathbf{k}_i^{(l)}\sin(\theta_t - \theta_i), \tag{21}$$

where J denotes a 90° rotation matrix.

At shallow layers, the attention score is dominated by the relative positional phase $(\theta_t - \theta_i)$. Consequently, visual tokens with smaller positional distance to the text token are preferentially attended.

In VLMs, tokens are typically ordered as system tokens, followed by visual tokens and text tokens. Visual tokens are usually arranged according to a patch-wise scan of the image from top to bottom. As a result, in the early layers, the model tends to assign higher weights to visual tokens located in the lower regions of the image, rather than selecting tokens based on their semantic importance.

D Attention without RoPE in VLM

To improve model efficiency, most existing methods perform token pruning in the shallow layers. However, due to the influence of RoPE, visual tokens cannot be accurately selected at early stages. To address this issue, FEATHER proposes to recompute the attention between visual and text tokens at the pruning layer, where RoPE is explicitly removed from the attention computation. Compared to RoPE-based attention, this design enables more reliable visual token selection in shallow layers.

Following a similar strategy, we also conduct layer selection experiments using attention without RoPE. As described in the Introduction, we first run the vanilla model once. Unlike the original approach, we recompute RoPE-free attention at each layer and rank visual tokens accordingly, selecting the top 20%.

In the second run, all visual tokens are retained in the first two layers. From the third layer onward, only the layer-specific top 20% visual tokens are preserved. Here, we report the results on a subset of the GQA training set, as shown in Fig. 9.

Despite removing the effect of RoPE, the relationship between layer depth and layer selection performance on GQA remains non-monotonic. For example, the layer selected by FEATHER (i.e., layer 8) does not exhibit sufficient capability in selecting informative visual tokens.

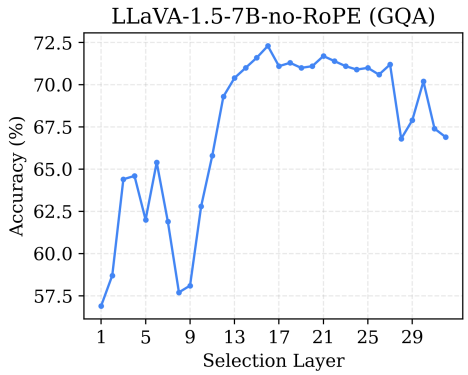


Figure 9: **Model-specific pruning layer analysis without RoPE.** We first run the vanilla model and recompute RoPE-free attention at each layer, recording the top 20% ranked visual tokens per layer. In the second run, all visual tokens are retained in the first two layers, while from the third layer onward only the layer-wise selected top 20% visual tokens are preserved. We report the results on the GQA training split. Despite removing RoPE, model performance still exhibits a non-monotonic relationship with layer depth.

E More Qualitative Comparisons

To provide a more intuitive comparison of SwiftVLM across different tasks, we present additional visualized results under varying computational budgets in Fig. 10 and Fig. 11.

F Limitation

Fine-grained grounding tasks are generally more sensitive to token budgets than non-localization tasks, as they require preserving more detailed visual evidence. While our current experiments evaluate representative budget settings, adaptively selecting task-specific token budgets remains an

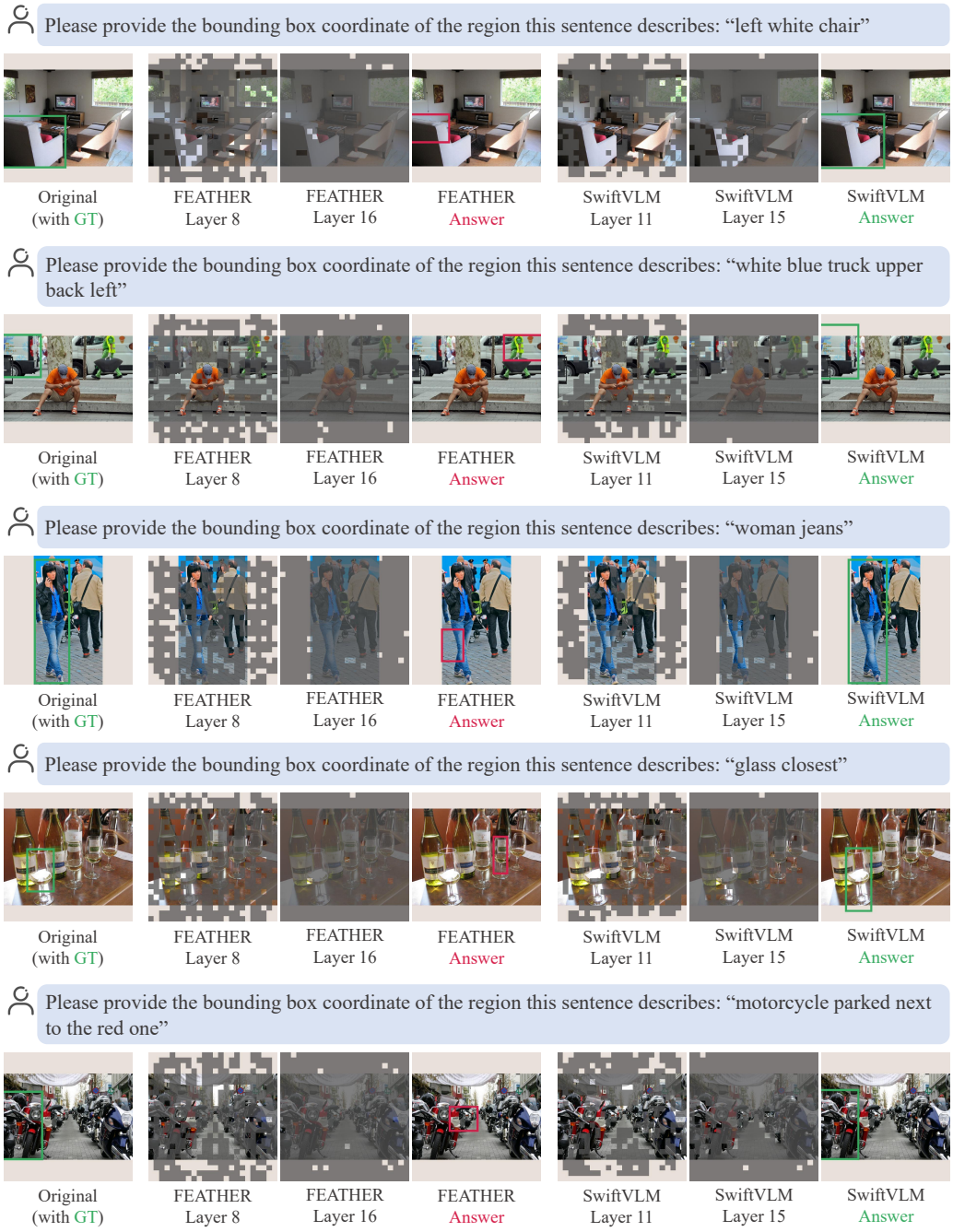
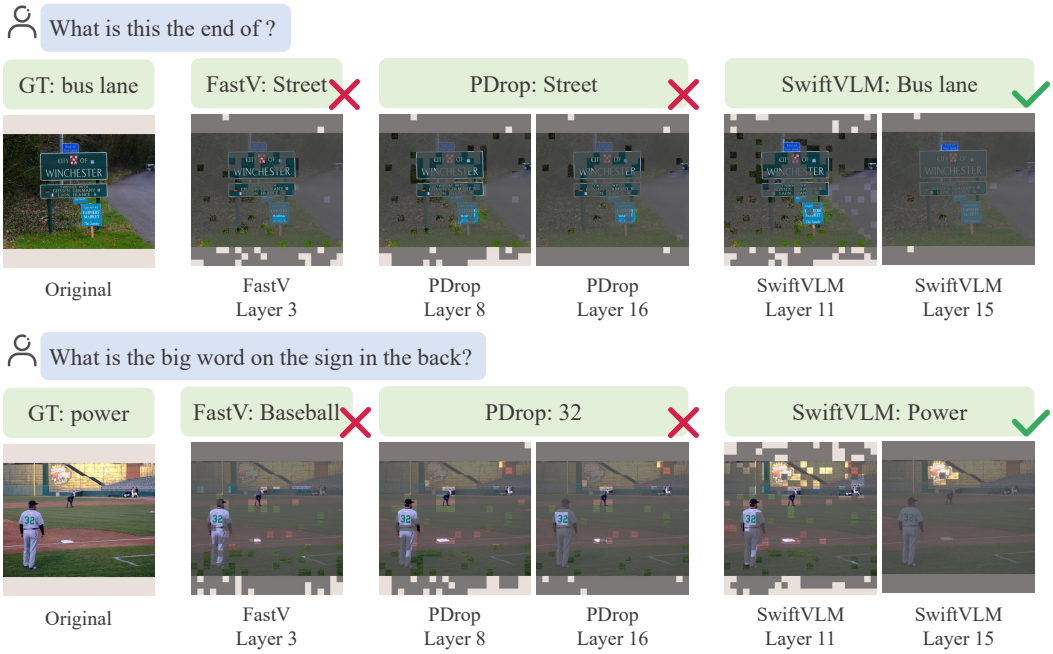
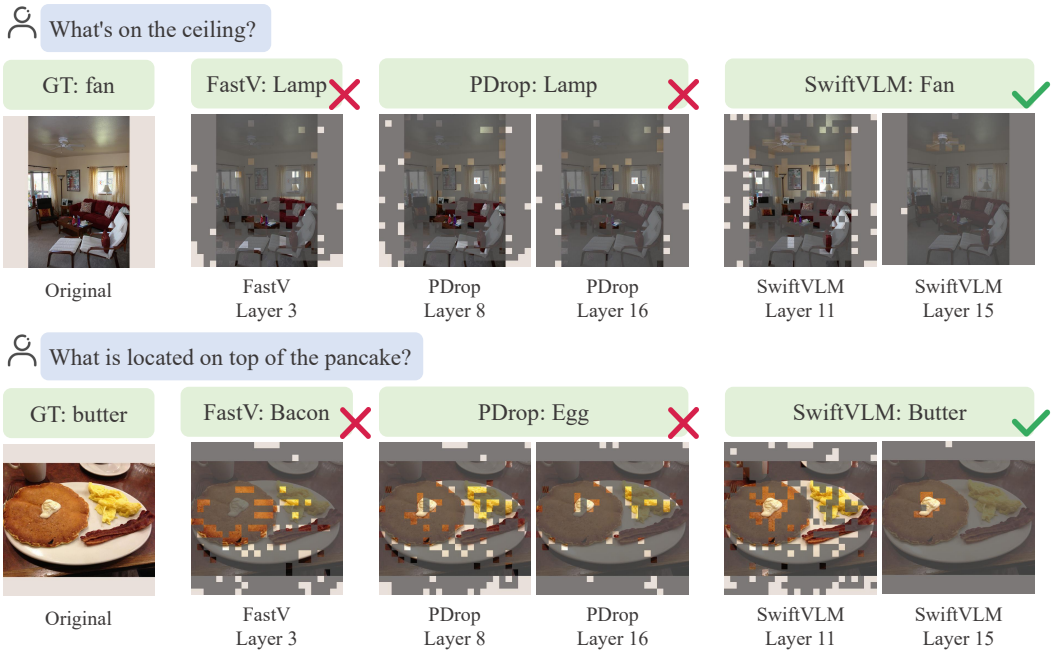


Figure 10: **Visualization comparison on localization tasks with an average of 192 visual tokens.**

interesting direction for future work, especially for balancing efficiency and fine-grained localization accuracy.



(a) Qualitative Results on TextVQA



(b) Qualitative Results on GQA

Figure 11: **Visualization comparison on non-localization tasks with an average of 128 visual tokens.**

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All claims are supported by the experimental results.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in Appendix.F.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: All assumptions and complete proofs are provided in the main paper.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The layer selection strategy is described in the main paper, while the pruning ratios are provided in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: Code will be publicly released before publication. All datasets and models used are publicly available.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The settings for each experiment are specified in their corresponding sections.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The runs for the experiments in the paper have low variance.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The computational resource usage is reported in the Appendix.A.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We confirm adherence to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Our work is focused on efficient VLM inference. We do not foresee any particular societal impacts from this work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not involve the release of new models or datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets and models used are open-source and publicly available.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not release any new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We did not conduct any crowdsourcing experiments or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: We did not conduct any research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLM is used only for writing, editing, or formatting purposes.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.