WHEN PRUNING RANKINGS REVERSE: STATE-CONDITIONED VISUAL TOKEN COMPRESSION ACROSS ARCHITECTURES

Anonymous authors
Paper under double-blind review

ABSTRACT

The high cost of visual tokens makes token compression key to accelerating VLM inference. Existing methods mainly select tokens by token-wise task relevance or inter-token diversity, and typically apply a fixed rule to all inputs. However, we find that the relative effectiveness of these two representative approaches reverses across model architectures and shifts with task and compression ratio; even under the same architecture and task, preferences differ across samples. This indicates that token selection should adapt to each sample rather than follow a fixed rule. We therefore propose SATOR (Sample-Adaptive Token Reduction), a training-free, sample-adaptive token selection method that requires no architecture- or task-specific parameters. SATOR estimates the task contribution of each token from attention and Value norms. Before selection, it uses the visual redundancy and task support of the current input to balance two marginal gains: a distinctiveness gain that suppresses redundancy with retained tokens, and a representativeness gain that encourages coverage of unretained task-relevant information. On LLaVA-NeXT-7B, SATOR retains only 11.1% of visual tokens while preserving 97.5% of the original average performance, achieving a 3.24× prefill speedup, a 2.19× end-to-end speedup, and a 6.02× reduction in KV-cache size.

1 INTRODUCTION

Vision-language models (VLMs) encode an image into a sequence of visual tokens that are fed into the language model alongside the text. Visual tokens usually far outnumber text tokens: LLaVA-1.5 (Liu et al., 2023) uses 576 per image, LLaVA-NeXT (Liu et al., 2024a) up to 2,880, and dynamic-resolution models such as Qwen3-VL (Bai et al., 2025) and InternVL3 (Zhu et al., 2025) even more at higher resolutions. As attention cost grows quadratically with sequence length, visual tokens dominate inference cost, yet many of them are redundant or irrelevant to the question. Visual token compression has thus become an important means of accelerating VLM inference.

Existing methods select visual tokens mainly by one of two signals: *relevance*, which retains the tokens most attended by the text (e.g., FastV (Chen et al., 2024a) and SparseVLM (Zhang et al., 2024)), or *diversity*, which retains mutually dissimilar tokens (e.g., DART (Wen et al., 2025) and DivPrune (Alvar et al., 2025)). Some works combine both signals or adopt adaptive budgets, yet still apply a single fixed rule to all inputs, implicitly assuming that one signal dominates on every input. However, we find that this assumption does not hold. Taking FastV and DART as representative relevance- and diversity-oriented methods, respectively, we compare them across architectures (Figure 1(a)): DART leads by 9.3–18.8 points on LLaVA-1.5 but only by 0.8–1.9 points on InternVL3, whereas on Qwen3-VL, FastV leads by 5.8 and 3.0 points at retention ratios of 33.3% and 22.2%, respectively. The ranking thus reverses across architectures.

Since the best rule varies with architecture, a natural idea is to specify a separate rule for each configuration. Whether this works depends on the granularity at which preferences vary: if they vary only across configurations (architecture, task, or retention ratio), one rule per configuration is enough; if they also differ across samples within a configuration, the rule must be chosen per sample. Figure 1(b) shows that preferences shift with task and retention ratio even within one architecture: on InternVL3, each rule consistently wins a different subset of tasks; on Qwen3-VL, the leading rule

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

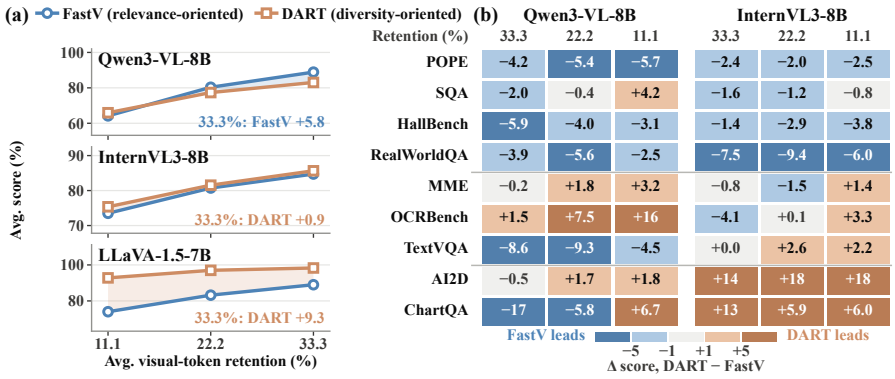


Figure 1: Selection-rule preferences. (a) Mean scores of relevance-oriented FastV and diversity-oriented DART vs. retention rate. (b) Per-benchmark DART − FastV gap (blue: FastV better; orange: DART better). Preferences vary with model, task, and retention rate (see Appendix A).

flips as retention drops from 33.3% to 11.1%. Yet this is still configuration-level variation. To probe the sample level, we fix architecture and task and compare the two rules sample by sample, finding two non-negligible groups in every configuration: one answered correctly only by the relevance rule, the other only by the diversity rule (Table 1). The "better" rule merely wins more samples without subsuming those won by the other, and choosing the right rule per sample would improve over it by a further 3.3–15.3 points. This gap far exceeds random variation, indicating that the preference difference is systematic and present within every configuration. Token selection should therefore adapt to each sample rather than follow a fixed rule.

Based on this analysis, we propose **SATOR** (Sample-Adaptive Token Reduction), a training-free, sample-adaptive token selection method with no architecture- or task-specific parameters. SATOR estimates each visual token's task contribution from relative attention and Value norms, and augments contribution-based ranking with two marginal gains: a distinctiveness gain that suppresses duplication with retained tokens when visual content is redundant, and a representativeness gain that encourages coverage of unretained task-relevant information when task evidence exceeds the budget. Before selection, SATOR estimates the input's visual redundancy and task support to derive a parameter-free routing weight that favors distinctiveness under higher redundancy and representativeness under greater support, letting selection adapt continuously to each sample.

In summary, our contributions are threefold:

- To our knowledge, we first systematically analyze token selection preferences across architectures, tasks, and samples: method rankings reverse across architectures, and samples favor different rules within one configuration, motivating sample-adaptive selection.
- We propose SATOR, a training-free, sample-adaptive token selection method that augments task contribution with distinctiveness and representativeness gains weighted by each input's redundancy and task support, requiring no per-configuration tuning.
- Across VLMs, benchmarks, and retention ratios, SATOR achieves the best average performance while substantially reducing latency and KV-cache cost.

2 Related Work

2.1 Multimodal Large Language Models

Multimodal large language models (MLLMs) integrate vision through visual encoders and cross-modal connectors. Flamingo (Alayrac et al., 2022) and BLIP-2 Li et al. (2023a) connected pre-trained vision and language models, while InstructBLIP Dai et al. (2023), MiniGPT-4 Zhu et al. (2024), and LLaVA Liu et al. (2023; 2024a) advanced multimodal understanding through alignment and instruction tuning. Qwen-VL Bai et al. (2023) and InternVL Chen et al. (2024b) improved visual and cross-modal representations; Qwen2-VL Wang et al. (2024), InternVL3 Zhu et al. (2025),

and Qwen3-VL Bai et al. (2025) further enhanced high-resolution understanding through dynamic resolution, multimodal pre-training, and visual fusion. The resulting longer visual sequences make token compression important for reducing inference cost.

2.2 Visual Token Compression in Vision-Language Models

Visual token compression reduces computation via compact representation, pruning, and merging. TokenPacker Li et al. (2025), LLaVA-Mini Zhang et al. (2025b), M^3 Cai et al. (2025), and DeCo Yao et al. (2024) compress tokens at the connector or representation level; LLaVA-PruMerge Shang et al. (2025), VisionZip Yang et al. (2025), and VisPruner Zhang et al. (2025a) combine saliency with aggregation or deduplication; FastV Chen et al. (2024a), SparseVLM Zhang et al. (2024), PyramidDrop Xing et al. (2024), and FitPrune Ye et al. (2025a) use language-model signals for one-shot or layer-wise pruning. TRIM Song et al. (2025) introduces image–text relevance, while DART Wen et al. (2025), DivPrune Alvar et al. (2025), CDPruner Zhang et al. (2026b), and SCOPE Deng et al. (2026) emphasize diversity, deduplication, and coverage. More recently, SIEVE Ou et al. (2026) and TRIO Zhang et al. (2026a) derive token importance from the inference process, while ATP-LLaVA Ye et al. (2025b), VisionTrim Yu et al. (2026), and AgilePruner Wang et al. (2026) explore adaptive budgets, text-guided merging, and image-aware policies. However, most methods fix a core criterion across images, limiting adaptation to input-dependent task information and redundancy.

3 Analysis: Why Fixed Selection Rules Are Insufficient

Existing visual token compression methods typically apply a fixed selection rule. To assess whether sample-adaptive selection is needed, we examine rule preferences across architectures, tasks, and inputs. A fixed rule suffices if preferences depend only on architecture or task, whereas sample-level adaptation is needed if they also vary across inputs.

We use two representative fixed rules as diagnostic probes: **FastV** (Chen et al., 2024a) for task relevance and **DART** (Wen et al., 2025) for token diversity. Let $u_r(x)$ denote the task utility of applying compression rule r to sample x. We define the sample-wise preference difference as $\Delta(x) = u_{\mathrm{DART}}(x) - u_{\mathrm{FastV}}(x)$. A positive $\Delta(x)$ indicates a preference for DART, and a negative value a preference for FastV. Rather than determining which rule is globally superior, we examine whether the sign of $\Delta(x)$ remains stable across compression conditions.

Rule Preferences across Architectures and Tasks. We first compare two representative rules and find that their ranking reverses across architectures (Figure 1(a)): DART leads by 9.3–18.8 points on LLaVA-1.5-7B and 0.8–1.9 points on InternVL3-8B, whereas FastV leads on Qwen3-VL-8B by 5.8 and 3.0 points at retention ratios of 0.333 and 0.222. Within an architecture, preferences further vary with task and compression strength (Figure 1(b)). On InternVL3, FastV wins on POPE, ScienceQA, HallBench, and RealWorldQA at all ratios, while DART wins on AI2D and ChartQA; on Qwen3-VL, FastV is preferred on 8 of 9 tasks at 0.333, yet 5 switch to DART at 0.111. Hence, the preferred rule is neither universal nor determined by architecture alone. However, if variation existed only at the configuration level, a fixed rule per architecture–task–retention setting would suffice. We therefore next examine whether samples within the same configuration also prefer different rules.

Sample-Level Preference Variation. Finally, we fix the architecture and task and compare the public implementations of the two rules at a retention ratio of 0.111 on 54,889 paired samples from POPE, MME, TextVQA, and ScienceQA, defining samples with different task utilities under the two rules as decisive samples. Table 1 shows that each rule wins on some samples in all 12 architecture–task configurations. On Qwen3-VL / POPE, for example, DART and FastV win on 9.3% and 9.6% of samples, respectively, so a near-zero average gap can mask opposing sample-level preferences. More importantly, a sample-wise oracle that selects the better rule for each input outperforms the best fixed rule by 3.3–15.3 percentage points in all configurations, indicating substantial room for input-dependent selection.

Table 1: **Sample-level preference.** Entries are DART/FastV win rates at a retention ratio of 0.111; parentheses show oracle gains (pp).

	POPE	MME	TextVQA	SQA
LLaVA	6.7/4.1 (4.1)	5.7/4.6 (4.6)	5.2/8.1 (4.3)	3.7/4.1 (3.7)
Qwen	9.3/9.6 (9.3)	8.2/7.7 (7.7)	11.5/14.5 (9.8)	9.1/5.3 (5.3)
InternVL	3.9/5.6 (3.9)	4.3/3.7 (3.7)	20.0/17.0 (15.3)	3.3/5.9 (3.3)

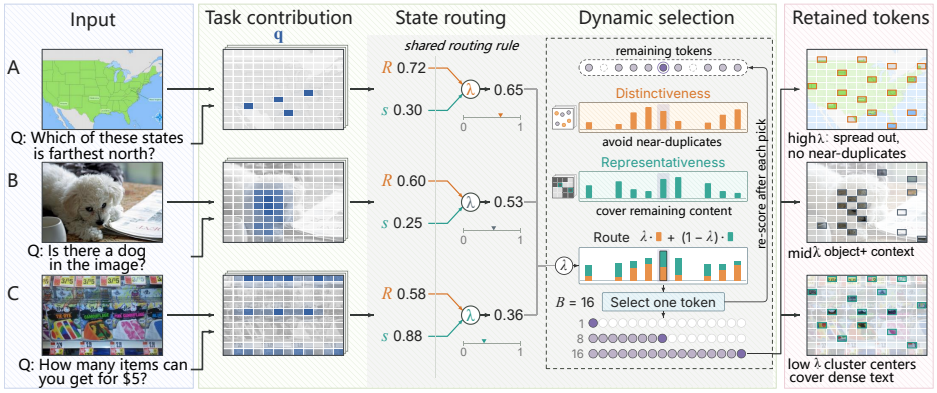


Figure 2: Framework overview. Different input states yield different λ: higher λ spreads retained tokens, while lower λ concentrates them on dense content.

Together, these analyses show that rule preference varies not only across architectures and tasks, but also across individual image–question pairs. Since task relevance and token diversity benefit different inputs, the selection mechanism should satisfy the following requirements:

- **R1: Continuous balancing.** Both token-wise task relevance and inter-token diversity are useful selection signals, so the selection objective should continuously balance the two rather than commit to either one.
- **R2: Input-conditioned adaptation.** Since the preferred behavior still varies after fixing the architecture and task, the balancing factor should be determined by the current input and be available before token selection.
- **R3: Calibration-free adaptation.** Since rule preference reverses across architectures and further changes with tasks and compression strength, the mechanism should avoid architecture- or task-specific calibrated constants that may not transfer across settings.

4 METHOD

We introduce a parameter-free, sample-adaptive token selection mechanism that satisfies R1–R3. If visual tokens are mutually dissimilar and all informative, retaining the top-K tokens by task contribution suffices. In practice, inputs deviate from this ideal in two ways.

- **Visual content can be redundant.** High-contribution tokens may be near-duplicates that waste the budget. We therefore introduce a *distinctiveness gain*, whose strength is governed by the redundancy state R.
- **Task evidence may exceed the budget.** The selected tokens should then also represent the relevant information that is not retained. We therefore introduce a *representativeness gain*, governed by the support state s.

Our method comprises task contribution estimation, two marginal gains, input state estimation, and dynamic routing with marginal selection, as shown in Figure. 2.

4.1 TASK-RELEVANT CONTRIBUTION ESTIMATION

To capture task relevance (R1), we define a contribution score q that weights both marginal gains and determines s. It measures contributions to textual representations rather than attention magnitude alone, and is comparable across images and compression stages.

Relative Attention. Let N be the number of visual tokens at the current stage, $\mathcal{L}$ the set of text tokens, and $\alpha_{r,j}^{(h)}$ the attention weight from text position r to visual token j in head h. Since total

visual attention may be affected by task-irrelevant factors such as attention sinks (Xiao et al., 2024), we keep only relative preferences among visual tokens:

$$p_{r,j}^{(h)} = \alpha_{r,j}^{(h)} \big/ \textstyle\sum_{k=1}^{N} \alpha_{r,k}^{(h)}, \qquad \bar{p}_j^{(h)} = \tfrac{1}{|\mathcal{L}|} \textstyle\sum_{r\in\mathcal{L}} p_{r,j}^{(h)}, \tag{1}$$

where $p_{r,j}^{(h)}$ denotes the relative attention weight on visual token j from text position r in head h, and $\bar{p}_j^{(h)}$ is its mean over all text positions. Accordingly, $p_r^{(h)} = [p_{r,1}^{(h)}, \ldots, p_{r,N}^{(h)}]$ and $\bar{p}^{(h)} = [\bar{p}_1^{(h)}, \ldots, \bar{p}_N^{(h)}]$ denote the visual attention distribution and its average, respectively.

Head Informativeness Weight. Let $H(\cdot)$ denote entropy and $\bar{H}_h = \frac{1}{|\mathcal{L}|} \sum_{r\in\mathcal{L}} H(p_r^{(h)})$ the mean entropy of head h. We define

$$\omega_h = 1 - \bar{H}_h \big/ H(\bar{p}^{(h)}) \in [0, 1], \tag{2}$$

which compares the mean entropy with the entropy of the mean distribution: distinct visual focus across text positions concentrates each $p_r^{(h)}$ and disperses $\bar{p}^{(h)}$, increasing ω_h; nearly invariant attention yields $\omega_h \approx 0$.

Raw Contribution. With $V_j^{(h)}$ the Value vector of visual token j in head h, we define the raw contribution as

$$c_j = \Big(\textstyle\sum_h \omega_h \big(\bar{p}_j^{(h)} \, \|V_j^{(h)}\|_2\big)^2 \Big)^{1/2}, \tag{3}$$

where h ranges over all attention heads. Since attention outputs are weighted sums of Value vectors, scaling by the Value norm avoids overvaluing highly attended tokens with small magnitudes Kobayashi et al. (2020); weighting by ω_h further favors text-adaptive heads.

Scale Normalization. To reduce scale differences across images and stages, we normalize raw contributions by their root mean square (RMS) τ within the current image, yielding the final score:

$$\tau = \big(\tfrac{1}{N} \textstyle\sum_{k=1}^{N} c_k^2\big)^{1/2}, \qquad q_j = c_j/(c_j + \tau) \in [0, 1). \tag{4}$$

Thus $q_j = 1/2$ when $c_j = \tau$, and tokens above or below this level score higher or lower, respectively.

4.2 Two Marginal Gains

Importance ranking fails when duplicates consume the budget or dispersed evidence remains uncovered, reflecting redundancy within the selected set and insufficient representation of unselected tokens, respectively. We therefore introduce distinctiveness and representativeness marginal gains, combined according to the input state (Sec. 4.4). Let x_j denote the hidden state of visual token j, T the selected set, and $A = \{1, \ldots, N\} \setminus T$ the candidate set.

Similarity Kernels. Since shared directions in Transformer hidden states inflate pairwise cosine similarities Ethayarajh (2019), we center them as $\tilde{x}_j = x_j - \frac{1}{N} \sum_{k=1}^{N} x_k$ and define two kernels:

$$S_{ij}^0 = \tfrac{1}{2}\big(1 + \cos(\tilde{x}_i, \tilde{x}_j)\big), \qquad S_{ij} = (S_{ij}^0)^2. \tag{5}$$

Mapping to $[0, 1]$ before squaring preserves the ordering of similarities, avoids treating strong negative correlations as high similarity, and suppresses weak similarities, while keeping $1 - S_{ij} \in [0, 1]$. By Gram matrix properties and the Schur product theorem, S^0 and S are positive semidefinite, as required by the determinant-based gain. We use S^0, linear in cosine similarity, for redundancy estimation (Sec. 4.3) and S for selection.

Distinctiveness Gain. To reduce repetition in the retained set, we favor candidates that are *important in themselves and not explained by already-selected tokens*. With the contribution-weighted kernel $L = \mathrm{diag}(q)\, S\, \mathrm{diag}(q)$ (i.e., $L_{ij} = q_i S_{ij} q_j$), the distinctiveness gain of candidate i is

$$g_i^{\mathrm{dis}}(T) = \frac{\det L_{T\cup\{i\}}}{\det L_T}, \tag{6}$$

which measures the growth of the information volume of the selected set after adding i, and reduces to ranking by task contribution ($g_i^{\mathrm{dis}} = q_i^2$) when $T = \varnothing$. Geometrically, g_i^{dis} is the squared residual

distance from the candidate to the subspace spanned by the selected tokens, so it avoids duplicating any single token and also captures *joint redundancy* induced by multiple selected tokens.

Representativeness Gain. When task-relevant tokens exceed the budget, we prefer tokens that *cover more of the unretained information*. Let $d_{ij} = 1 - S_{ij} \in [0, 1]$ and let $u_j = \min_{k \in T} d_{jk}$, initialized to 1, be the residual of token j. The representativeness gain of candidate i is

$$g_i^{\text{rep}}(T) = \sum_{j=1}^{N} q_j \big(u_j^2 - \min(u_j, d_{ij})^2\big), \tag{7}$$

The gain is thus high when i is near many high-contribution tokens still far from T. After selecting p, we update $u_j \leftarrow \min(u_j, d_{jp})$, so later picks target uncovered regions. When $S_{ij} \approx 0$ for all $i \neq j$, $g_i^{\text{dis}} \approx q_i^2$ and $g_i^{\text{rep}} \approx q_i$, reducing both gains to importance ranking. SATOR therefore corrects, rather than replaces, importance ranking under repetition or dense evidence.

4.3 Compression State of the Current Input

The need for each correction varies by input: high redundancy makes importance-only selection prone to duplicates, while sufficient budgets reduce coverage needs. To satisfy R2, we quantify both needs using only current-stage hidden states, contribution scores, and the retention budget B available before selection:

$$\rho = \tfrac{1}{N} \textstyle\sum_{i=1}^{N} \max_{j \neq i} S_{ij}^0, \qquad R = \text{clip}(2\rho - 1,\, 0,\, 1), \qquad s = \text{clip}\left(\tfrac{1}{B} \textstyle\sum_{j=1}^{N} q_j,\, 0,\, 1\right). \tag{8}$$

Redundancy R (Need for Distinctiveness Correction). We measure duplicate-selection risk via token substitutability, averaging per-token maximum similarities since a single close neighbor suffices and unrelated pairs would dilute the score. The term $2\rho - 1$ maps ρ back to the cosine scale as the mean nearest-neighbor cosine similarity, clipped at zero to discard nonpositive values; larger R calls for stronger distinctiveness correction.

Support s (Need for Representativeness Correction). We measure how much omitted task information the retained tokens must represent: $\sum_j q_j$ acts as a soft count of task-relevant tokens, and dividing by B compares demand with retention capacity. Larger s requires retained tokens to represent more information indirectly, calling for stronger representativeness correction.

4.4 Routing Weight and Dynamic Marginal Selection

We weight the distinctiveness gain by $\lambda \in [0, 1]$ and the representativeness gain by $1 - \lambda$. A continuous weight satisfies R1 while avoiding calibration-dependent thresholds. Since R and s characterize the needs for distinctiveness and representativeness correction, respectively, λ should vary with these needs and satisfy:

- **C1: Nondecreasing in R.** Higher redundancy raises the risk of duplicate selection, so the weight of distinctiveness correction should increase.
- **C2: Nonincreasing in s.** A greater need for representativeness correction should increase its weight $1 - \lambda$.
- **C3: Boundary behavior.** $\lambda = 0$ at $R = 0$ and $\lambda = 1$ at $R = 1$, regardless of s: without redundancy, no duplicates arise and no distinctiveness correction is needed; under full redundancy, tokens are interchangeable, so avoiding duplicates dominates.
- **C4: Parameter-free.** λ depends only on R and s, without architecture- or task-specific calibration parameters, satisfying R3.

We therefore adopt $\lambda = R^{1+s}$. This form uses R as the base weight and increases with redundancy; as s rises from 0 to 1, λ decreases continuously from R to R^2, favoring representativeness correction. Its boundary values are independent of s and it has no tunable parameters, thus satisfying C1–C4.

Selection. The two gains differ in scale and evolve differently during selection, so mixing them directly would distort the balance specified by λ. At each step, we therefore normalize each gain over the current candidate set A, i.e., $\hat{g}_i^{\text{dis}} = g_i^{\text{dis}} / \sum_{k \in A} g_k^{\text{dis}}$ and likewise $\hat{g}_i^{\text{rep}}$; if a gain sums to zero over A, its normalized term is set to zero. We then select

$$p^* = \arg\max_{i \in A} \left[\lambda\, \hat{g}_i^{\text{dis}} + (1 - \lambda)\, \hat{g}_i^{\text{rep}}\right]. \tag{9}$$

Table 2: Performance comparison of different token reduction methods on LLaVA Series.

Methods	Venue	GQA	MMB	MMB-cn	MME	POPE	SQA	VQAv2	TextVQA	Average
LLaVA-1.5-7B		**Upper Bound, 576 Tokens**								
Vanilla	–	61.9	64.7	58.1	1862	85.9	69.5	78.5	58.2	100%
LLaVA-1.5-7B		**Retain 192 Tokens (33.3%)**								
VisionZip	CVPR'25	59.3	63.0	57.3	1783	85.3	68.9	76.8	57.3	97.8%
HoloV	NeurIPS'25	59.0	65.4	58.0	1820	85.6	69.8	76.7	57.4	98.8%
ApET	CVPR'26	60.2	63.4	57.9	1808	86.3	68.5	76.2	54.4	97.7%
SpecFlow	ICML'26	58.3	65.8	56.1	1827	85.8	69.7	76.4	57.9	98.4%
Ours	–	61.1	64.6	58.7	1806	86.0	68.9	77.8	57.9	**99.3%**
LLaVA-1.5-7B		**Retain 128 Tokens (22.2%)**								
VisionZip	CVPR'25	57.6	62.0	56.7	1762	83.2	68.9	75.6	56.8	96.4%
HoloV	NeurIPS'25	57.7	63.9	56.5	1802	84.0	69.8	75.5	56.8	97.2%
ApET	CVPR'26	58.9	62.3	56.4	1801	86.1	68.7	75.1	53.9	96.6%
SpecFlow	ICML'26	57.6	64.3	55.9	1794	84.9	69.9	75.3	56.8	97.2%
Ours	–	60.5	63.0	58.1	1792	86.9	68.5	77.1	57.4	**98.5%**
LLaVA-1.5-7B		**Retain 64 Tokens (11.1%)**								
VisionZip	CVPR'25	55.1	60.1	55.4	1690	77.0	69.0	72.4	55.5	93.1%
HoloV	NeurIPS'25	55.3	63.3	55.1	1715	80.3	69.5	72.8	55.4	94.4%
ApET	CVPR'26	56.9	61.2	54.4	1714	84.4	68.9	72.5	53.0	94.1%
SpecFlow	ICML'26	55.3	63.7	54.3	1713	80.5	69.7	73.7	54.9	94.4%
Ours	–	58.6	62.3	56.7	1727	86.4	69.0	75.5	56.2	**96.7%**
LLaVA-NeXT-7B		**Upper Bound, All 2880 Tokens**								
Vanilla	–	64.2	67.4	60.6	1851	86.5	70.1	81.8	61.4	100%
LLaVA-NeXT-7B		**Retain 11.1% Tokens**								
VisionZip	CVPR'25	59.3	63.1	55.6	1702	82.1	67.3	76.2	58.9	93.7%
HoloV	NeurIPS'25	61.7	65.3	57.5	1738	83.9	68.9	79.5	58.7	96.2%
ApET	CVPR'26	61.0	63.5	56.6	1783	85.6	–	75.8	54.4	94.2%
SpecFlow	ICML'26	62.5	66.7	56.8	1707	85.0	68.6	80.1	59.5	96.7%
Ours	–	61.7	65.8	60.1	1832	85.2	68.1	79.1	58.7	**97.5%**

Table 3: Performance comparison of different token reduction methods on Qwen3-VL-8B.

Methods	Venue	MME	POPE	SQA	TextVQA	AI2D	ChartQA	HallBench	RealWorldQA	Average
Qwen3-VL-8B		**Upper Bound, All Tokens**								
Vanilla	–	2416	89.7	94.0	80.5	86.0	82.7	58.2	70.9	100%
Qwen3-VL-8B		**Retain 33.3% Tokens**								
HoloV	NeurIPS'25	2360	88.5	93.0	77.2	83.9	77.4	55.9	68.1	96.8%
TransPrune	CVPR'26	2390	89.8	93.5	78.1	84.1	71.6	56.3	68.5	96.7%
Ours	–	2398	90.0	93.2	79.3	84.4	77.0	56.8	69.8	**98.1%**
Qwen3-VL-8B		**Retain 22.2% Tokens**								
HoloV	NeurIPS'25	2328	88.6	92.0	75.8	82.1	71.6	53.7	67.4	94.6%
TransPrune	CVPR'26	2305	89.0	92.4	76.7	82.5	64.8	52.6	67.7	93.6%
Ours	–	2338	89.5	91.7	78.2	83.4	73.0	53.9	67.6	**95.6%**
Qwen3-VL-8B		**Retain 11.1% Tokens**								
HoloV	NeurIPS'25	2091	85.4	88.1	66.1	76.9	54.5	48.4	60.3	85.1%
TransPrune	CVPR'26	2093	86.4	88.3	71.5	76.3	35.8	48.6	60.7	83.4%
Ours	–	2261	88.4	88.2	73.3	79.6	52.5	48.2	62.9	**88.1%**

Both gains are updated as T grows, with λ balancing them according to the current input state, yielding dynamic marginal selection. Each compression stage independently recomputes R, s, and λ before selection.

5 Experiments

5.1 Experimental Setup

Models and Baselines. Using LLaVA-1.5 Liu et al. (2023), LLaVA-NeXT Liu et al. (2024a), Qwen3-VL Bai et al. (2025), and InternVL3 Zhu et al. (2025) as backbones, we compare our method with representative visual token compression baselines, e.g., HoloV Zou et al. (2026) and TransPrune Li et al. (2026).

Benchmarks. We conduct evaluations on GQA Hudson & Manning (2019), MME Fu et al. (2025), ScienceQA Lu et al. (2022), TextVQA Singh et al. (2019), VQAv2 Goyal et al. (2017), MM-Bench Liu et al. (2024b), POPE Li et al. (2023b), AI2D Kembhavi et al. (2016), ChartQA Masry et al. (2022), HallusionBench Guan et al. (2024), and RealWorldQAX.ai (2024).

Table 4: Performance comparison of different token reduction methods on InternVL3-8B.

Methods	Venue	MME	POPE	SQA	TextVQA	AI2D	ChartQA	HallBench	RealWorldQA	Average
InternVL3-8B		**Upper Bound, All Tokens**								
Vanilla	–	2335	91.2	97.8	78.9	85.3	81.7	48.2	71.2	100%
InternVL3-8B		**Retain 33.3% Tokens**								
HoloV	NeurIPS'25	2245	89.5	96.7	72.3	80.2	71.3	45.2	64.4	93.8%
TransPrune	CVPR'26	2355	91.1	97.0	77.1	83.1	71.3	45.6	70.8	97.0%
Ours	–	2348	91.1	97.0	77.2	84.1	75.1	47.2	71.1	**98.2%**
InternVL3-8B		**Retain 22.2% Tokens**								
HoloV	NeurIPS'25	2181	88.2	93.7	62.6	76.5	58.8	42.6	61.7	87.7%
TransPrune	CVPR'26	2336	90.9	96.6	75.2	82.0	65.0	45.4	69.7	95.2%
Ours	–	2322	91.2	96.8	75.6	83.5	69.9	45.5	70.2	**96.3%**
InternVL3-8B		**Retain 11.1% Tokens**								
HoloV	NeurIPS'25	1962	83.8	87.2	41.2	70.8	40.2	37.3	56.3	75.7%
TransPrune	CVPR'26	2255	89.5	95.4	67.6	79.2	53.9	42.6	68.4	90.1%
Ours	–	2266	90.1	96.3	69.4	80.2	58.8	42.0	69.5	**91.6%**

Table 5: **Input states and routing weights.** Mean [5th, 95th percentile] of R, s, and λ at the default budget; each row pools 1,600 image samples per benchmark (Table 10).

Model	Stage	R	s	λ
LLaVA-1.5	1	0.95 [0.90, 0.98]	1.00 [1.00, 1.00]	0.90 [0.81, 0.96]
	2	0.71 [0.61, 0.79]	0.62 [0.50, 0.71]	0.57 [0.45, 0.68]
	3	0.64 [0.55, 0.71]	0.93 [0.77, 1.00]	0.43 [0.31, 0.52]
Qwen3-VL	1	0.74 [0.70, 0.81]	0.90 [0.68, 1.00]	0.57 [0.50, 0.66]
	2	0.65 [0.58, 0.72]	0.37 [0.23, 0.51]	0.56 [0.48, 0.63]
	3	0.62 [0.54, 0.69]	0.68 [0.50, 0.88]	0.45 [0.35, 0.54]
InternVL3	1	0.70 [0.64, 0.80]	0.89 [0.66, 1.00]	0.52 [0.42, 0.64]
	2	0.68 [0.56, 0.75]	0.47 [0.36, 0.61]	0.57 [0.40, 0.66]
	3	0.65 [0.50, 0.72]	0.62 [0.43, 0.78]	0.50 [0.32, 0.60]

Table 6: **Ablation on the routing weight λ.** Average score (%) relative to the unpruned model on POPE, MME, TextVQA, and SQA, with 11.1% of visual tokens retained on average .

Variant	LLaVA-1.5	Qwen3-VL	InternVL3	Mean
Importance ranking only	91.79	92.23	95.05	93.02
Representativeness only	94.94	93.35	92.92	93.74
Distinctiveness only	96.44	93.83	95.91	95.39
Best fixed λ (0.6)	96.47	94.05	96.19	95.57
Stage-wise constant	96.26	93.73	96.17	95.38
Without support state	96.43	93.72	95.89	95.35
Reversed R direction	95.55	93.67	95.29	94.84
Reversed s direction	96.55	93.84	96.11	95.50
SATOR	97.30	94.27	96.47	96.01

5.2 Main Results

Results on the LLaVA Series Table 2 compares token reduction methods on LLaVA-1.5-7B and LLaVA-NeXT-7B. On LLaVA-1.5-7B, our method achieves 99.3%, 98.5%, and 96.7% average relative performance at 192/128/64-token budgets, ranking first throughout. Reducing retention from 33.3% to 11.1% costs only 2.6 points, versus 4.7 for VisionZip, demonstrating robustness to aggressive compression. On LLaVA-NeXT-7B, with up to 2880 visual tokens, our method maintains 97.5% performance at just 11.1% retention, with an MME score of 1832 exceeding all competitors (1702–1783), highlighting its adaptability to high-resolution, long-sequence inputs.

Results on Qwen3-VL-8B

As shown in Table 3, our method ranks first at all three budgets, achieving 98.1%, 95.6%, and 88.1% average performance. At 33.3% retention, POPE (90.0) even exceeds the uncompressed model (89.7), and MME (2398) nearly matches it (2416). At the tightest 11.1% retention, our method still leads the runner-up by 3.0 points and scores 73.3 on TextVQA, above HoloV (66.1). These results highlight adaptability across architectures without tuning and strong performance under tight budgets.

Results on InternVL3-8B Table 4 shows that our method reaches 98.2%, 96.3%, and 91.6% across the three budgets, outperforming all baselines. At 11.1% retention, where HoloV falls to 75.7%, it still leads the runner-up TransPrune by 1.5 points, showing robustness under tight budgets. Its consistent lead on all three architectures highlights the adaptability of sample-wise dynamic selection without architecture- or task-specific tuning.

5.3 Ablation Study and Analysis

Dynamics of the Routing Weight The routing weight λ is recomputed per sample at each compression stage from the current R and s (Section 4.4; statistics in Table 5). As near-duplicates are removed, R falls from 0.70–0.95 at stage 1 to 0.62–0.65 at stage 3, while s rises again at stage 3 as

the remaining tokens are mostly task-relevant. Hence λ drops from 0.56–0.57 at stage 2 to 0.43–0.50 at stage 3, shifting selection toward representativeness in all 12 architecture–task configurations (Table 10). It also varies across architectures and inputs: at stage 1 it is 0.90 on LLaVA-1.5 versus 0.52–0.57 on the other two, and even with architecture, stage, and task fixed, its 5th–95th percentile range remains 0.09–0.34 wide, so λ adapts to each input rather than acting as a constant.

Ablation on the Routing Weight To examine the design of λ, we change only how it is set and keep all other components fixed (Table 6; details in Appendix C). Importance ranking only keeps the top-B tokens by q; representativeness only and distinctiveness only fix $\lambda = 0$ and 1; the best fixed λ is the test-set optimum over all swept values, and the stage-wise constant uses the mean λ of each architecture and stage (Table 5); removing the support state sets $\lambda = R$; and the reversed variants $\lambda = (1 - R)^{1+s}$ and $\lambda = R^{2-s}$ make λ decrease with R and increase with s, respectively.

The results support the design of SATOR in three respects. (1) The two corrections should be combined according to the input state (R1): neither suffices alone, and representativeness only even falls below importance ranking only on InternVL3 (92.92% vs. 95.05%). (2) The gain beyond single corrections comes mainly from sample-adaptive routing (R2, R3): SATOR exceeds the stronger endpoint, distinctiveness only, by 0.62 points on average, whereas the best fixed λ, though tuned on the test sets, recovers less than a third of this margin (95.57%). The stage-wise constant matches the mean λ of SATOR at each stage but removes per-sample variation, and it recovers none of the margin (95.38%), trailing SATOR by 0.63 points on average and by 1.04 points on LLaVA-1.5. (3) The support state and the monotonic directions are indispensable (C1, C2): removing the support state, reversing s, and reversing R lower the average to 95.35%, 95.50%, and 94.84%, respectively, with the last even below distinctiveness only; all three trail SATOR on every architecture.

Component Ablation We replace one component at a time (Table 7; full in Table 12). Spatially uniform attention keeps head weights but spreads each head's attention uniformly over visual tokens; equal head weights sets $\omega_h \equiv 1$; without Value norm drops $\|V_j^{(h)}\|_2$ from c_j; and mean-similarity redundancy uses the all-pair mean instead of nearest-neighbor similarity in ρ. Uniform attention costs 9.59 points , confirming relative attention as the main signal. Equal head weights and without Value norm trail SATOR on all models ; mean-similarity redundancy costs 0.44 points , as unrelated pairs dilute the mean.

Table 7: Avg. relative performance (%) on POPE, MME, TextVQA, and SQA at 11.1% avg. per-layer retention.

Variant	LLaVA-1.5	Qwen3-VL	InternVL3	Mean
Spatially uniform attention	94.84	86.67	77.76	86.42
Equal head weights	96.63	94.10	96.00	95.58
Without Value norm	97.07	94.15	96.13	95.78
Mean-similarity redundancy	96.46	94.29	95.96	95.57
SATOR	97.30	94.27	96.47	96.01

Efficiency Analysis. All methods keep 11.1% of visual tokens. Times (mm:ss) are wall-clock over all 8,910 POPE questions on one RTX 4090 (FP16, batch size 1), including token selection but not model loading. Relative to Full (Table 8), SATOR speeds up prefill by 3.24×/1.77× and end-to-end inference by 2.19×/1.42× on LLaVA-NeXT/LLaVA-1.5, and cuts the per-sample KV cache from 1156/319 to 192/63 MB. Versus near-zero-overhead Random, SATOR runs 36%/9% longer but gains 6.1/10.6 F1; SparseVLM is slower and less accurate. Each compression stage costs $O(TD^2 + N^2D + BN^2)$ (N/B: visual tokens before/after; T: sequence length; D: hidden size); the closed-form λ and incremental greedy updates keep overhead low (full derivation in Appendix D.1).

Table 8: **Efficiency comparison on POPE**

Model	Method	Total time	Prefill time	KV cache (MB)	POPE F1
LLaVA-NeXT	Full	46:11	37:21	1156	86.5
	Random	15:33	7:20	192	79.1
	SparseVLM	35:55	26:57	200	82.4
	SATOR	21:05	11:31	192	85.2
LLaVA-1.5	Full	16:39	9:55	319	85.9
	Random	10:49	4:10	63	75.8
	SparseVLM	12:58	6:18	66	77.5
	SATOR	11:46	5:37	63	86.4

6 Conclusion

In this paper, we show that the relative effectiveness of visual token selection rules varies across architectures, tasks, and even individual samples, so no fixed rule generalizes across inputs. We therefore propose SATOR, a training-free and parameter-free method that adaptively balances distinctiveness and representativeness according to the redundancy and task support of each input. SATOR achieves the best average performance on four VLMs across all tested retention ratios.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language editing, including improving grammar, clarity, and presentation. We did not use generative AI tools to generate research ideas, develop the methodology, analyze results, write code, create figures, or identify citations. All AI-assisted text was carefully reviewed by the authors. We take full responsibility for the final content of this work, including any text produced with the aid of generative AI.

REFERENCES

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.

Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. Divprune: Diversity-based visual token pruning for large multimodal models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9392–9401. IEEE, 2025.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization. *Text Reading, and Beyond*, 2(1):1, 2023.

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.

Mu Cai, Jianwei Yang, Jianfeng Gao, and Yong Jae Lee. Matryoshka multimodal models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Uhj5OxAz7I.

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*, pp. 19–35. Springer, 2024a.

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*, pp. 24185–24198, 2024b.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning. *Advances in neural information processing systems*, 36:49250–49267, 2023.

Jinhong Deng, Wen Li, Joey Tianyi Zhou, and Yang He. Scope: Saliency-coverage oriented token pruning for efficient multimodel llms. *Advances in Neural Information Processing Systems*, 38: 161527–161552, 2026.

Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 55–65, 2019.

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

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

Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, et al. Hallusionbench: an advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14375–14385. IEEE, 2024.

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*, pp. 6700–6709, 2019.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *European conference on computer vision*, pp. 235–251. Springer, 2016.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7057–7075, 2020.

Ao Li, Yuxiang Duan, Jinghui Zhang, Congbo Ma, Yutong Xie, Gustavo Carneiro, Mohammad Yaqub, and Hu Wang. Transprune: Token transition pruning for efficient large vision-language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 39529–39538, 2026.

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*, pp. 19730–19742. PmLR, 2023a.

Wentong Li, Yuqian Yuan, Jian Liu, Dongqi Tang, Song Wang, Jie Qin, Jianke Zhu, and Lei Zhang. Tokenpacker: Efficient visual projector for multimodal llm. *International Journal of Computer Vision*, 133(10):6794–6812, 2025.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023b.

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

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26286–26296. IEEE, 2024a.

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*, pp. 216–233. Springer, 2024b.

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.

Ahmed Masry, Jia Qing Tan, Shafiq Joty, Enamul Hoque, et al. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the association for computational linguistics: ACL 2022*, pp. 2263–2279, 2022.

Congyang Ou, Ruike Song, Yang Zhou, Libo Sun, Haokui Zhang, and Zhenbo Luo. When vision becomes text: Visual token pruning via cross-modal residual guidance in vlms. *arXiv preprint arXiv:2608.10489*, 2026.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22857–22867. IEEE, 2025.

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*, pp. 8317–8326, 2019.

Dingjie Song, Wenjun Wang, Shunian Chen, Xidong Wang, Michael X Guan, and Benyou Wang. Less is more: A simple yet effective token reduction method for efficient multi-modal llms. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 7614–7623, 2025.

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

Yahong Wang, Juncheng Wu, Zhangkai Ni, Longzhen Yang, Yihang Liu, Chengmei Yang, Ying Wen, Lianghua He, Xianfeng Tang, Hui Liu, et al. When token pruning is worse than random: Understanding visual token information in vllms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 31910–31919, 2026.

Zichen Wen, Yifeng Gao, Shaobo Wang, Junyuan Zhang, Qintong Zhang, Weijia Li, Conghui He, and Linfeng Zhang. Stop looking for "important tokens" in multimodal language models: Duplication matters more. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 9972–9991, 2025.

X.ai. Grok-1.5 vision preview. `https://x.ai/blog/grok-1.5v`, 2024.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024.

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.

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 *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19792–19802. IEEE, 2025.

Linli Yao, Lei Li, Shuhuai Ren, Lean Wang, Yuanxin Liu, Xu Sun, and Lu Hou. Deco: Decoupling token compression from semantic abstraction in multimodal large language models. *arXiv preprint arXiv:2405.20985*, 2024.

Weihao Ye, Qiong Wu, Wenhao Lin, and Yiyi Zhou. Fit and prune: Fast and training-free visual token pruning for multi-modal large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 22128–22136, 2025a.

Xubing Ye, Yukang Gan, Yixiao Ge, Xiao-Ping Zhang, and Yansong Tang. Atp-llava: Adaptive token pruning for large vision language models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24972–24982. IEEE, 2025b.

Hanxun Yu, Wentong Li, Xuan Qu, Song Wang, Junbo Chen, and Jianke Zhu. Visiontrim: Unified vision token compression for training-free mllm acceleration. *arXiv preprint arXiv:2601.22674*, 2026.

Haokui Zhang, Congyang Ou, Dawei Yan, Peng Wang, Qingsen Yan, Yu Zhang, Ying Li, and Rong Xiao. Trio: Token reduction via inference-objective guidance for efficient vision-language models. *arXiv preprint arXiv:2602.04657*, 2026a.

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 *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20857–20867. IEEE, 2025a.

Qizhe Zhang, Mengzhen Liu, Lichen Li, Ming Lu, Yuan Zhang, Junwen Pan, Qi She, and Shanghang Zhang. Beyond attention or similarity: Maximizing conditional diversity for token pruning in mllms. *Advances in Neural Information Processing Systems*, 38:25438–25468, 2026b.

Shaolei Zhang, Qingkai Fang, Yang Yang, and Yang Feng. Llava-mini: Efficient image and video large multimodal models with one vision token. In *International Conference on Learning Representations*, volume 2025, pp. 53285–53310, 2025b.

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.

Deyao Zhu, Xiaoqian Shen, Xiang Li, Mohamed Elhoseiny, et al. Minigpt-4: Enhancing vision-language understanding with advanced large language models. In *International Conference on Learning Representations*, volume 2024, pp. 18378–18394, 2024.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.

Xin Zou, Di Lu, Yizhou Wang, Yibo Yan, Yuanhuiyi Lyu, Xu Zheng, Linfeng Zhang, and Xuming Hu. Don't just chase "highlighted tokens" in mllms: Revisiting visual holistic context retention. *Advances in Neural Information Processing Systems*, 38:39800–39832, 2026.

APPENDIX

A COMPLETE RESULTS OF THE FASTV–DART COMPARISON

Table 9 provides the exact values for all results shown in Figure 1. The retention rate denotes the average fraction of visual tokens retained relative to the original number of visual tokens.

B PER-BENCHMARK INPUT STATES AND ROUTING WEIGHTS

Table 10 breaks down the statistics in Table 5 by benchmark.

Stage-wise trends are consistent across benchmarks. In all 12 architecture–task configurations, R is non-increasing across compression stages (only InternVL3/MME stays unchanged between the first two stages), s is lowest at stage 2, and λ at stage 3 is lower than at stage 2. The trends in the main text therefore do not arise from mixing benchmarks. On LLaVA-1.5, s saturates at 1 at stage 1 on every benchmark, in which case $\lambda = R^2$.

Task differences agree with task characteristics. At stages 2 and 3, POPE has the lowest s among the four benchmarks on every architecture, consistent with its task evidence concentrating on the few tokens of the queried object. On LLaVA-1.5 and Qwen3-VL, ScienceQA-IMG has the highest R and λ at all three stages, possibly because its diagram-like images contain large uniform regions.

Variation across samples exceeds variation across tasks. With the architecture, stage, and task fixed, the widths of the 5th–95th percentile ranges are 0.05–0.27 for R, 0.10–0.44 for s (excluding the saturated stage 1 of LLaVA-1.5), and 0.09–0.34 for λ. In contrast, the spread of mean λ across the four benchmarks within an architecture and stage is only 0.04–0.09, and in all nine architecture–stage combinations it is smaller than the percentile range width of any single benchmark. A fixed per-task λ therefore cannot capture this variation, and λ must be determined per sample.

C DETAILS OF THE ROUTING-WEIGHT ABLATION

All variants share the pruning layers, retention budget, and remaining pipeline of SATOR, and differ only in how λ is set. For each benchmark, we divide the score by that of the unpruned model (using the total score for MME) and average the ratios over the four benchmarks. The best fixed λ applies one constant to all samples and stages; it is selected on the test sets by the average over the three architectures, so it serves as an optimistic reference. The stage-wise constant uses the mean λ of each architecture and stage in Table 5: 0.90/0.57/0.43 for LLaVA-1.5, 0.57/0.56/0.45 for Qwen3-VL, and 0.52/0.57/0.50 for InternVL3. Unlike s, R cannot be removed alone: setting $R \equiv 1$ yields $\lambda \equiv 1$, which coincides with distinctiveness only. Table 11 summarizes all variants.

D DETAILS OF THE COMPONENT ABLATION

Setup. All variants share the pruning layers, retention budget, and evaluation protocol of SATOR (Appendix C), and each replaces a single component (Table 12). Besides the four variants described in the main text, mean head aggregation replaces the cross-head ℓ_2 aggregation with the mean; max normalization sets $q_j = c_j / \max_k c_k$ while keeping s unchanged; without saturation replaces $c_j/(c_j + \tau)$ with c_j/τ and recomputes s accordingly; the linear kernel uses S^0 instead of S in selection; and unnormalized mixing combines the two gains without normalization.

Results. All nine variants yield lower average performance than SATOR, with the linear kernel being the closest (95.96%) and spatially uniform attention causing the largest drop (9.59 points). The variants examine the following designs: spatially uniform attention tests whether the spatial distribution of attention is the main selection signal; equal head weights and mean head aggregation test the head informativeness weight and the cross-head ℓ_2 aggregation; without Value norm tests the role

Table 9: FASTV and DART on three architectures at three average retention rates (complete values for Figure 1).

Qwen3-VL-8B Method	Ret.	MME	POPE	SQA	TextVQA	AI2D	ChartQA	OCRBench	HallBench	RealWorldQA	Avg
FastV	0.333	2254.03	88.95	91.52	75.31	78.98	64.44	54.1	53.78	64.58	88.88
DART	0.333	2249.53	84.80	89.49	66.74	78.53	47.48	55.6	47.89	60.65	83.05
FastV	0.222	2089.93	87.70	87.31	70.63	73.45	46.48	39.1	48.71	61.70	80.34
DART	0.222	2139.66	82.28	86.91	61.36	75.13	40.72	46.6	44.71	56.08	77.34
FastV	0.111	1738.79	77.95	78.04	57.31	66.22	19.40	18.2	41.60	50.46	64.17
DART	0.111	1828.92	72.29	82.20	52.79	68.01	26.12	33.8	38.55	47.97	65.94
InternVL3-8B Method	Ret.	MME	POPE	SQA	TextVQA	AI2D	ChartQA	OCRBench	HallBench	RealWorldQA	Avg
FastV	0.333	2288.50	90.24	95.79	59.91	63.44	44.20	65.8	44.75	66.41	84.72
DART	0.333	2265.55	87.79	94.15	59.94	77.30	56.96	61.7	43.33	58.95	85.65
FastV	0.222	2250.05	89.39	94.50	54.35	58.48	42.20	55.1	42.17	67.19	80.72
DART	0.222	2209.31	87.37	93.26	56.95	76.62	48.12	55.2	39.23	57.78	81.54
FastV	0.111	2108.25	87.79	90.08	48.88	55.38	31.16	41.6	39.31	62.35	73.50
DART	0.111	2147.27	85.25	89.24	51.11	73.38	37.12	44.9	35.47	56.34	75.36
LLaVA-1.5-7B Method	Ret.	Tokens	GQA	MMB	MMB-cn	MME	POPE	SQA	VQAv2	TextVQA	Avg
FastV	0.333	192	52.7	61.2	57.0	1612	64.8	67.3	67.1	52.5	89.0
DART	0.333	192	60.0	63.6	57.0	1856	82.8	69.8	76.7	57.4	98.3
FastV	0.222	128	49.6	56.1	56.4	1490	59.6	60.2	61.8	50.6	83.2
DART	0.222	128	58.7	63.2	57.5	1840	80.1	69.1	75.9	56.4	97.0
FastV	0.111	64	46.1	48.0	52.7	1256	48.1	51.1	55.0	47.8	73.6
DART	0.111	64	55.9	60.6	53.2	1765	73.9	69.8	72.4	54.4	92.8

of write magnitude; max normalization and without saturation test the bounded, RMS-referenced mapping of the contribution score; the linear kernel tests the suppression of weak similarities by squaring; mean-similarity redundancy tests nearest-neighbor redundancy estimation; and unnormalized mixing tests the scale alignment of the two gains. Together, these results confirm that each component contributes to the average performance of SATOR.

D.1 COMPUTATIONAL COMPLEXITY

Notation. We analyze one sample in a single compression. Let N be the number of visual tokens before compression, $B \leq N$ the budget, T the current sequence length, and $L_q \leq T$ the number of text query positions after the image that are used to read the task contribution. The LLM has hidden size D, H attention heads of dimension d_h with $Hd_h = D$, and key/value dimension $D_{kv} \leq D$. Costs are counted as multiply–accumulate operations, and element-wise operations are counted as $O(1)$.

Cost of a single execution. *(i) Query, key, and value representations.* Reading the task contribution requires query, key, and value representations of the current hidden states, obtained by linear projections at a cost of $O\big(TD(D + 2D_{kv})\big) \subseteq O(TD^2)$. Strictly, only L_q queries are needed, so this is an upper bound.

(ii) Task contribution $\mathbf{q}$. Attention logits between L_q queries and T keys over H heads cost $O(HL_qTd_h) = O(L_qTD)$, and the softmax costs $O(HL_qT)$. Normalizing the conditional distributions over the visual span and computing the marginal distribution and both entropies, which yield the head weights ϕ_h, cost $O(HL_qN)$; value norms cost $O(ND)$; cross-head aggregation and RMS normalization cost $O(HN)$. The total is $O(L_qTD)$.

(iii) Input state. Centering and normalizing the visual hidden states costs $O(ND)$. The centered cosine matrix $\mathbf{U}\mathbf{U}^\top$ is a single dense product costing $O(N^2D)$. The kernel $\mathbf{K} = \mathbf{K}^0 \circ \mathbf{K}^0$, the dissimilarity $\mathbf{\Delta} = \mathbf{1} - \mathbf{K}$, and the redundancy R (a row-wise nearest neighbor followed by a mean) each cost $O(N^2)$. The total is $O(N^2D)$.

(iv) Routing. The support s is a single sum and $\lambda = R^{1+s}$ is a scalar operation, for a total of $O(N)$.

Table 10: **Per-benchmark input states and routing weights.** Means and 5th–95th percentiles of R, s, and λ under the default budget; each cell covers 1,600 samples.

Model	Benchmark	Stage	R	s	λ
LLaVA-1.5-7B	POPE	1	0.93 [0.90, 0.98]	1.00 [1.00, 1.00]	0.87 [0.80, 0.96]
		2	0.68 [0.60, 0.75]	0.59 [0.49, 0.65]	0.55 [0.45, 0.64]
		3	0.62 [0.55, 0.69]	0.89 [0.75, 1.00]	0.41 [0.32, 0.50]
	MME	1	0.94 [0.90, 0.98]	1.00 [1.00, 1.00]	0.89 [0.80, 0.96]
		2	0.71 [0.62, 0.78]	0.61 [0.51, 0.70]	0.58 [0.47, 0.67]
		3	0.64 [0.57, 0.71]	0.91 [0.76, 1.00]	0.43 [0.34, 0.52]
	TextVQA	1	0.94 [0.89, 0.98]	1.00 [1.00, 1.00]	0.89 [0.80, 0.96]
		2	0.68 [0.58, 0.78]	0.61 [0.45, 0.71]	0.54 [0.40, 0.67]
		3	0.62 [0.52, 0.70]	0.95 [0.78, 1.00]	0.40 [0.27, 0.51]
	ScienceQA-IMG	1	0.96 [0.93, 0.98]	1.00 [1.00, 1.00]	0.93 [0.87, 0.96]
		2	0.76 [0.69, 0.81]	0.66 [0.59, 0.72]	0.63 [0.54, 0.70]
		3	0.67 [0.59, 0.72]	0.97 [0.87, 1.00]	0.46 [0.35, 0.53]
Qwen3-VL-8B	POPE	1	0.73 [0.70, 0.76]	0.80 [0.64, 1.00]	0.57 [0.52, 0.61]
		2	0.62 [0.57, 0.66]	0.27 [0.22, 0.32]	0.54 [0.48, 0.59]
		3	0.59 [0.54, 0.64]	0.61 [0.49, 0.69]	0.43 [0.37, 0.50]
	MME	1	0.74 [0.70, 0.79]	0.95 [0.77, 1.00]	0.56 [0.50, 0.63]
		2	0.64 [0.59, 0.71]	0.35 [0.23, 0.45]	0.55 [0.49, 0.62]
		3	0.63 [0.57, 0.69]	0.64 [0.47, 0.79]	0.47 [0.40, 0.55]
	TextVQA	1	0.74 [0.69, 0.78]	0.97 [0.83, 1.00]	0.55 [0.48, 0.61]
		2	0.65 [0.58, 0.72]	0.44 [0.33, 0.54]	0.54 [0.44, 0.63]
		3	0.59 [0.49, 0.67]	0.74 [0.51, 0.95]	0.41 [0.26, 0.51]
	ScienceQA-IMG	1	0.77 [0.71, 0.83]	0.89 [0.67, 1.00]	0.60 [0.53, 0.69]
		2	0.68 [0.64, 0.73]	0.40 [0.30, 0.50]	0.58 [0.52, 0.65]
		3	0.67 [0.62, 0.70]	0.73 [0.62, 0.89]	0.49 [0.43, 0.55]
InternVL3-8B	POPE	1	0.69 [0.65, 0.72]	0.88 [0.67, 1.00]	0.50 [0.43, 0.57]
		2	0.68 [0.61, 0.73]	0.44 [0.35, 0.52]	0.57 [0.48, 0.64]
		3	0.65 [0.58, 0.71]	0.56 [0.37, 0.73]	0.52 [0.41, 0.61]
	MME	1	0.69 [0.65, 0.77]	0.84 [0.61, 1.00]	0.51 [0.43, 0.62]
		2	0.69 [0.60, 0.75]	0.47 [0.37, 0.57]	0.58 [0.46, 0.66]
		3	0.66 [0.55, 0.72]	0.66 [0.50, 0.83]	0.50 [0.37, 0.60]
	TextVQA	1	0.69 [0.64, 0.74]	0.88 [0.67, 1.00]	0.50 [0.42, 0.58]
		2	0.66 [0.57, 0.72]	0.47 [0.38, 0.56]	0.54 [0.43, 0.63]
		3	0.63 [0.53, 0.70]	0.58 [0.43, 0.72]	0.48 [0.36, 0.57]
	ScienceQA-IMG	1	0.74 [0.63, 0.84]	0.97 [0.83, 1.00]	0.55 [0.40, 0.71]
		2	0.68 [0.52, 0.76]	0.50 [0.36, 0.67]	0.57 [0.35, 0.69]
		3	0.65 [0.46, 0.73]	0.66 [0.48, 0.80]	0.49 [0.27, 0.61]

(v) Greedy selection. Let S, with $|S| = t$, be the selected set at step $t \in \{0, \dots, B-1\}$. The distinctiveness gain is the conditional variance $d_j^2 = \det \mathbf{L}_{S\cup\{j\}} / \det \mathbf{L}_S$ under the DPP kernel $\mathbf{L} = \mathrm{diag}(\mathbf{q})\,\mathbf{K}\,\mathrm{diag}(\mathbf{q})$. Evaluating determinants for every candidate costs $O(Nt^3)$ per step and $O(NB^4)$ in total. Following Incremental Cholesky update , we instead maintain $\mathbf{c}_j \in \mathbb{R}^t$ and d_j^2 for every candidate; after selecting p,

$$e_j = \frac{L_{jp} - \langle \mathbf{c}_j, \mathbf{c}_p \rangle}{d_p}, \qquad \mathbf{c}_j \leftarrow [\mathbf{c}_j;\, e_j], \qquad d_j^2 \leftarrow d_j^2 - e_j^2. \tag{10}$$

The inner products for all candidates form one $N \times t$ matrix–vector product, so step t costs $O(Nt)$ and the total is $\sum_{t<B} O(Nt) = O(B^2 N)$. Since $\mathbf{K}$ is positive semidefinite, so is $\mathbf{L}$, and the residuals remain non-negative in exact arithmetic.

Table 11: **Variants in the routing-weight ablation.**

Variant	λ	Removed component	Question tested
Importance ranking only	top-B by q	Both corrections	Are the corrections necessary?
Representativeness only	0	Distinctiveness gain and input states	Is representativeness alone sufficient?
Distinctiveness only	1	Representativeness gain and input states	Is distinctiveness alone sufficient?
Best fixed λ	Tuned constant	State conditioning	Can a tuned constant replace state conditioning?
Stage-wise constant	Stage mean of λ	Within-stage variation across samples	Is per-sample adaptation needed (R2)?
Without support state	R	Support state s	Does s add benefit?
Reversed R direction	$(1-R)^{1+s}$	Direction of C1 (also violates C3)	Does the direction of C1 matter?
Reversed s direction	R^{2-s}	Direction of C2	Does the direction of C2 matter?
SATOR	R^{1+s}	–	–

Table 12: **Complete results of the component ablation.** Average performance (%) over POPE, MME, TextVQA, and ScienceQA relative to the unpruned model under the default budget. Each variant replaces one component of SATOR.

Variant	LLaVA-1.5	Qwen3-VL	InternVL3	Mean
Spatially uniform attention	94.84	86.67	77.76	86.42
Equal head weights	96.63	94.10	96.00	95.58
Mean head aggregation	96.88	94.45	96.22	95.85
Without Value norm	97.07	94.15	96.13	95.78
Max normalization	97.02	94.13	96.29	95.81
Without saturation	97.04	94.21	96.51	95.92
Linear kernel	96.89	94.90	96.10	95.96
Mean-similarity redundancy	96.46	94.29	95.96	95.57
Unnormalized mixing	96.80	94.47	95.64	95.64
SATOR	97.30	94.27	96.47	96.01

The representativeness gain relies on the residual distance $r_i = \min\big(1, \min_{u\in S} \Delta_{iu}\big)$ from each token to the selected set. Recomputing the coverage objective for every candidate costs $O(N(t+1))$, i.e., $O(N^2 t)$ per step and $O(N^2 B^2)$ in total. Maintaining $\mathbf{r}$ instead gives

$$g_j = \sum_{i=1}^{N} q_i \big[r_i^2 - \min(r_i, \Delta_{ij})^2\big], \tag{11}$$

which costs $O(N)$ per candidate and $O(N^2)$ per step; after selecting p, the update $r_i \leftarrow \min(r_i, \Delta_{ip})$ costs $O(N)$. The total is $O(BN^2)$.

Normalizing the two gains over the unselected set, mixing them with λ, and taking the argmax cost $O(N)$ per step and $O(BN)$ in total. Greedy selection therefore costs $O(BN^2 + B^2N) = O(BN^2)$ since $B \le N$, and is dominated by the representativeness gain.

(vi) Sequence reassembly. Sorting the selected indices costs $O(B \log B)$, and gathering and concatenating the tokens costs $O(TD)$.

Total. Summing (i)–(vi) and using $L_q \le D$,

$$C = O\big(\underbrace{TD^2}_{\text{query/key/value}} + \underbrace{L_q TD}_{\text{contribution}} + \underbrace{N^2 D}_{\text{input state}} + \underbrace{BN^2}_{\text{greedy}}\big) = O\big(TD^2 + N^2 D + BN^2\big). \tag{12}$$

The TD^2 term stems from the query, key, and value representations; state estimation and greedy selection themselves cost $O\big(N^2(D+B)\big)$, which reduces to $O(N^2 D)$ when $B \le D$.

Overhead of per-sample adaptation. Compared with the same selector under a fixed λ, SATOR additionally computes only R and s. R is the mean of row-wise maxima over $\mathbf{K}^0$, which is already formed to build $\mathbf{K}$, and costs $O(N^2)$; s is a single sum over $\mathbf{q}$ and costs $O(N)$; and $\lambda = R^{1+s}$ costs $O(1)$. None of these involves a matrix product, and all are dominated by the $O(N^2 D)$ term of (iii), so adaptation leaves the complexity unchanged. Because λ is obtained in closed form, each sample requires a single greedy pass; selecting λ by running the selector with $|\Lambda|$ candidate values would instead raise the greedy cost to $O(|\Lambda| B N^2)$. Finally, the greedy procedure runs exactly B steps with no convergence criterion or data-dependent branching, so its cost depends only on (N, B) and not on image content.