ATTEND, TRANSFORM, OR SILENCE: OPERATOR-LEVEL VISUAL SKIPPING FOR EFFICIENT MULTIMODAL LLM INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal large language models (MLLMs) increasingly process long visual-token sequences, making decoder prefill computationally expensive. Existing acceleration methods typically remove visual tokens or skip their updates at the layer level, but these coarse strategies may discard fine-grained evidence or suppress useful operators together with redundant ones. We study visual computation from an answer-observable perspective and find that late visual-token updates can remain large while having little effect on answer representations and generated outputs. By decomposing each Transformer layer into attention and FFN operators, we further show that useful visual computation is operator-dominant and varies across layers. We therefore propose OpSkip, an operator-level visual-token skipping framework that preserves the full visual sequence while selectively bypassing visual attention, FFN, or both. OpSkip across four MLLMs and 10 VQA benchmarks reduce computation to **55%–67%** of the vanilla TFLOPs while retaining **95.0%–99.5%** average performance. OpSkip achieves the highest average retention on three of the four architectures and also provides consistent prefill and time-to-first-token speedups.

Code: `https://anonymous.4open.science/r/OpSkip-FB49`

1 INTRODUCTION

Multimodal large language models (MLLMs) encode images into visual tokens and concatenate them with text for autoregressive generation (Liu et al., 2024a; An et al., 2025; Liu et al., 2024b; Bai et al., 2025b;a). High-resolution inputs and image tiling improve perceptual coverage but can expand visual sequences to thousands of tokens, making LLM prefill costly (Li et al., 2023a; Liu et al., 2024b; Bai et al., 2025b;a; Chen et al., 2024). Existing acceleration methods reduce this computation either by pruning, selecting, or merging visual tokens (Huang et al., 2026; Xing et al., 2024; Zhang et al., 2026; Yang et al., 2025), or by skipping visual-token updates in selected Transformer layers (Yuan et al., 2025; Lin et al., 2025; Ma et al., 2026a). However, these approaches impose coarse redundancy assumptions: token reduction may irreversibly discard fine-grained evidence, layer-wise skipping removes attention and FFN computation together, and V-Skip (Ma et al., 2026a) assumes a fixed preference for retaining FFNs while skipping visual attention. They therefore cannot adaptively determine whether attention, FFN, or both are redundant at each layer, potentially suppressing useful computation.

We revisit visual redundancy by asking whether visual-token updates remain relevant to answer generation. Across multiple MLLMs and VQA benchmarks, late visual states continue to change substantially while exerting diminishing effects on answer-token representations. We term these large but weakly consequential updates *answer-silent*, showing that update magnitude alone does not measure useful computation. This redundancy may also arise within individual Transformer operators rather than align with entire layers. Recent analyses support this finer view, showing that attention primarily reconfigures information whereas FFNs drive semantic transformation (Xi et al., 2026). Their importance for visual-token computation should therefore be assessed separately at each layer.

Motivated by this observation, we decompose each Transformer layer into its attention and FFN operators and ask a finer question: *which operator is actually necessary for visual-token computa-*

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.

tion in each layer? By separately measuring the answer-observable contribution of attention and FFN, we find strong layer-wise heterogeneity, some layers are attention-dominant, some are FFN-dominant, and others show little answer-observable contribution. Based on this observation, we propose an operator-level visual-token skipping framework that preserves the full visual-token sequence while bypassing redundant visual computation. This design avoids the irreversible evidence loss of token pruning and releases visual skipping from the static assumption that either attention or FFN is always the important operator. Across four MLLMs and 10 VQA benchmarks, OpSkip reduces computation to **55%–67%** of the vanilla TFLOPs while retaining **95.0%–99.5%** average performance. These results show that useful visual computation is operator-dependent rather than block-aligned, motivating efficiency methods based on answer-observable contributions instead of update magnitude or fixed architectural boundaries.

Our contributions are summarized as follows:

- We identify answer-silent visual-token updates in late MLLM layers and introduce answer-observable diagnostics to measure their effects on final prompt representations, answer-token distributions, and generated sequences.
- We reveal that useful visual computation is operator-dominant and layer-dependent by decomposing each Transformer layer into attention and FFN operators, exposing visual redundancy beyond token and layer-level views.
- We propose OpSkip, an operator-level visual-token skipping framework that preserves the full visual-token sequence while bypassing redundant visual computation. Across four MLLMs and 10 VQA benchmarks, OpSkip reduces computation to **55%–67%** of the vanilla TFLOPs while retaining **95.0%–99.5%** average performance.

2 Answer-Silent Visual Updates

In decoder-only MLLMs, visual tokens are repeatedly updated throughout the decoder. However, because causal masking prevents them from attending to subsequent text tokens, continued changes in their hidden states do not necessarily imply a sustained influence on the final generated answer. This motivates our first question:

Question 1. *As visual token states evolve through the decoder, to what extent do their updates continue to influence answer generation?*

Measuring visual-update influence. We first trace each visual update to the final hidden state at the last prompt position, which we refer to as the *final prompt representation*. The language-modeling head directly maps this representation to the logits of the first answer token.

Let H_l and H'_l denote the input and output hidden states of decoder layer l, and let V and $\bar{V}$ index the visual and non-visual token positions, respectively. We define the visual update and its *Visual Update Magnitude* (VUM) as

$$\Delta V_l = (H'_l - H_l)_V, \qquad \mathrm{VUM} = |\Delta V_l|_F. \tag{1}$$

The layer-wise measurements show that VUM remains substantial throughout the decoder and generally becomes larger toward later layers. We next ask whether these increasingly large updates exert a correspondingly large influence on the final prompt representation.

To isolate the contribution of the visual update at layer l, we restore the visual-token outputs to their layer inputs while retaining the normally computed non-visual outputs:

$$(\widetilde{H}'_l)_V = (H_l)_V, \qquad (\widetilde{H}'_l)_{\bar{V}} = (H'_l)_{\bar{V}}. \tag{2}$$

We apply this intervention to one layer at a time and recompute all subsequent layers. It removes ΔV_l from the visual states propagated through subsequent decoder layers while preserving both the incoming visual representations and any visual information already incorporated into the non-visual outputs at layer l.

Let F_l denote the unmodified model computation following layer l. The vanilla and intervened passes yield

$$r_0 = F_l(H'_l), \qquad r' = F_l(\widetilde{H}'_l), \tag{3}$$

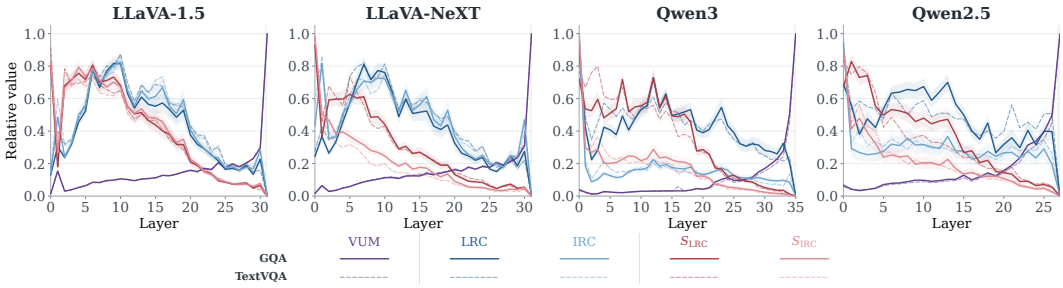

Figure 1: Layer-wise visual update magnitude, answer-observable influence, and answer-observable efficiency. All curves are normalized within each model and dataset for visualization. Across both GQA and TextVQA, late layers exhibit increasing visual-token update magnitude but decreasing answer-observable influence and efficiency.

where r_0 and r' are the final prompt representations obtained from the vanilla and intervened forward passes.

We first characterize this effect locally using the *Linearized Representation Change* (LRC):

$$\mathrm{LRC} = \left| DF_l(H'_l)\left[H'_l - \widetilde{H}'_l\right] \right|_2, \tag{4}$$

where $DF_l(H'_l)[\cdot]$ denotes the Jacobian–vector product evaluated at the vanilla layer output. The perturbation $H'_l - \widetilde{H}'_l$ equals ΔV_l at the visual positions and zero elsewhere.

Because LRC reflects both update magnitude and the sensitivity of the final prompt representation to the update after propagation through subsequent decoder layers, we further define

$$S_{\mathrm{LRC}} = \frac{\mathrm{LRC}}{\max(\mathrm{VUM}, \epsilon)}, \tag{5}$$

where $\epsilon = 10^{-8}$ is used for numerical stability. S_{LRC} measures the sensitivity of the final prompt representation to the visual-update direction after propagation through subsequent decoder layers.

Observation 1. Figure 1 reveals two opposing depth-wise trends: VUM increases in later layers, whereas LRC and S_{LRC} decrease. Thus, late visual states change more but exert less influence on the final prompt representation.

However, the JVP characterizes only the local response around the unmodified layer output H'_l. Because the remaining decoder contains nonlinear transformations, a weak linearized response does not necessarily imply that complete removal of the update will have a similarly weak effect.

Question 2. *Does the attenuation persist when a late-layer visual update is removed in full and the remaining decoder is recomputed?*

Evaluating complete update removal. We therefore apply the full intervention in Eq. 2 and recompute all subsequent decoder layers to obtain r'. We define the *Interventional Representation Change* (IRC) as

$$\mathrm{IRC} = \|r_0 - r'\|_2, \tag{6}$$

and its update-normalized counterpart as

$$S_{\mathrm{IRC}} = \frac{\mathrm{IRC}}{\max(\mathrm{VUM}, \epsilon)}. \tag{7}$$

Observation 2. Figure 1 shows that IRC and S_{IRC} also decrease in later layers, confirming that the reduced influence persists under full update removal and nonlinear recomputation.

This leads to another question:

Question 3. *Does the separation between visual-state evolution and representation change also extend to answer prediction and generation?*

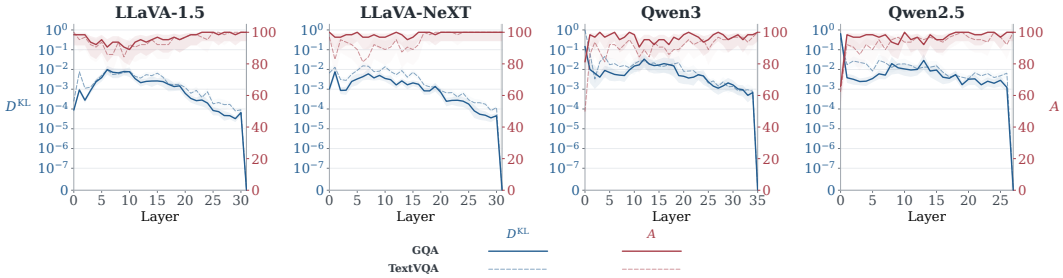

Figure 2: Layer-wise effects of visual-update removal on answer prediction and generation. D_l^{KL} measures the divergence in answer-token distributions, while A_l denotes exact sequence agreement between vanilla and intervened generations.

Measuring effects on answer prediction and generation. We evaluate the same intervention using both answer-token distributions and independently generated sequences. For an input x, let $y_{1:T}$ denote the greedy generation of the unmodified model, where T is the length of the generated answer. At answer position t, we condition both the vanilla and intervened runs on the same reference prefix $y_{<t}$, yielding next-token distributions $p(\cdot \mid x, y_{<t})$ and $p'_l(\cdot \mid x, y_{<t})$, respectively. We define their average full-vocabulary divergence as

$$D_l^{\mathrm{KL}}(x) = \frac{1}{T} \sum_{t=1}^{T} D_{\mathrm{KL}} \left[p(\cdot \mid x, y_{<t}) \,\|\, p'_l(\cdot \mid x, y_{<t}) \right]. \tag{8}$$

Conditioning both runs on the same prefix isolates the predictive change caused by the visual-update intervention from differences introduced by earlier decoding decisions. To determine whether these distributional changes ultimately alter the generated answer, we additionally decode the intervened model independently using its own generated prefix. For an evaluation set $\mathcal{X}$ containing N examples, let $y(x)$ and $y'_l(x)$ denote the complete vanilla and intervened greedy generations. We define *Exact Sequence Agreement* as

$$A_l = \frac{1}{N} \sum_{x \in \mathcal{X}} \mathbf{1} \left[y'_l(x) = y(x) \right]. \tag{9}$$

Observation 3. Figure 2 shows that D_l^{KL} decreases in later layers, while A remains high and generally increases. Together with the representation-level attenuation above, this reveals a progressive decoupling of late visual updates from answer generation, with diminishing influence on both token predictions and complete outputs.

We term this separation *answer-silent visual updating*. To validate it at the task level, we hold visual-token states fixed in late decoder layers while allowing text tokens to continue attending to these representations.

Table 1: Effect of freezing late visual-token states after layer 24.

Model	Method	GQA	POPE	TextVQA	SQA	MMB	MME	MMMU
LLaVA-1.5-7B	Full	61.9	85.9	58.2	69.5	64.2	1866.2	36.1
	Frozen visual	61.1	86.4	57.2	69.6	64.2	1878.1	35.7
Qwen3-VL-8B	Full	61.6	89.1	80.1	94.4	84.8	2390.4	51.3
	Frozen visual	61.0	88.8	79.9	94.7	84.6	2352.8	52.4

Table 1 shows that holding late visual-token states fixed causes only minor performance changes across seven benchmarks and both architectures. This intervention supports the answer-silent trend: late visual updates are often unnecessary, while text tokens can access the frozen visual representations.

Finding 1: Answer-Silent Visual Updating. Late-layer visual states continue to undergo substantial updates, yet the influence of these updates progressively weakens at the representation, prediction, and generation levels. Consistently, freezing late visual states avoids repeated visual-state updates with limited performance loss.

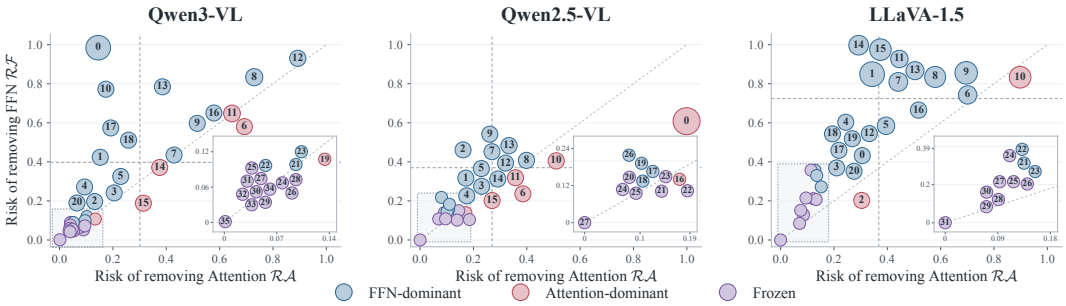

Figure 3: Layer-wise operator risks across 3 models. Each point represents a layer, with attention-skipping risk on the horizontal axis and FFN-skipping risk on the vertical axis. Layers above the diagonal belong to the FFN domain, while those below belong to the attention domain. Low-risk late layers are highlighted by the insets.

3 METHODOLOGY

The above finding motivates a finer-grained analysis of where the remaining answer-relevant visual computation resides within each Transformer block. We therefore move beyond the block-level view and decompose each layer into its attention and FFN operators, asking which operator contributes most strongly to answer-relevant visual computation at different depths.

3.1 OPERATOR RISK ANALYSIS

We first describe how to identify the layer-wise operator structure of the visual branch without retraining the model or using human-annotated answers. Given a decoder-only MLLM, let the hidden states at layer l be

$$H^l = [H^l_{\mathcal{V}}; H^l_{\mathcal{T}}], \tag{10}$$

where $H^l_{\mathcal{V}} \in \mathrm{R}^{N_v \times d}$ denotes the visual-token hidden states and $H^l_{\mathcal{T}}$ denotes the textual hidden states, including the system prompt, question tokens, and possible answer-prefix tokens. We decompose the visual branch of a Transformer block into two residual updates:

$$\bar{H}^l_{\mathcal{V}} = H^l_{\mathcal{V}} + \left[A^l(H^l)\right]_{\mathcal{V}}, \qquad H^{l+1}_{\mathcal{V}} = \bar{H}^l_{\mathcal{V}} + \left[F^l(\bar{H}^l)\right]_{\mathcal{V}}, \tag{11}$$

where A^l is the self-attention update and F^l is the FFN update at layer l. The subscript $\mathcal{V}$ denotes the visual-token rows. This decomposition allows us to ask a more fine-grained question than whether an entire layer can be skipped:

Question 4. *Which operator dominates the visual-branch computation at each layer?*

For an input sample x, the full model produces a reference next-token distribution from the final prompt-token hidden state after prefill:

$$p_0(\cdot|x) = \mathrm{softmax}(z_0(x)). \tag{12}$$

This reference distribution is obtained from the model itself and does not require ground-truth labels. We estimate layer-wise operator risks on a small calibration set $\mathcal{D}$ sampled from multiple benchmarks, using these samples only to probe how the output distribution changes under counterfactual visual-operator interventions.

$$\mathcal{R}s(l) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} D_{\mathrm{KL}}\Big(p_0(\cdot|x) \| p^{(l)}_s(\cdot|x)\Big), \quad s \in \{\mathcal{A}, \mathcal{F}, \mathcal{AF}\}. \tag{13}$$

Here, $\mathcal{RA}(l)$, $\mathcal{RF}(l)$, and $\mathcal{R}_{\mathcal{AF}}(l)$ measure the output risk of removing visual attention, removing visual FFN, and freezing the visual branch at layer l, respectively.

$$\Gamma_l = \log \frac{\mathcal{RA}(l) + \epsilon}{\mathcal{RF}(l) + \epsilon}. \tag{14}$$

A positive Γ_l indicates an attention-dominant layer, while a negative Γ_l indicates an FFN-dominant layer. Layers with uniformly small risks, especially small $\mathcal{R}_{\mathcal{AF}}(l)$, are treated as frozen layers.

Finding 2: Figure 3 shows that visual computation in MLLM Transformer layers exhibits layer-dependent operator dominance, suggesting that redundancy reduction should be guided by the dominant operator of each layer.

3.2 Operator-Level Skipping

We convert the layer-wise risk profile into an inference-time skipping policy through two steps, budgeted layer selection and policy assignment. Given a skipping budget B, we rank layers by $\mathcal{RA}(l) + \mathcal{RF}(l)$ and select the B lowest-risk layers for simplification. These are layers where removing either visual attention or visual FFN individually induces only a small output shift.

For each selected layer, we assign the least risky visual-branch policy. If $\mathcal{RA}(l) < \mathcal{RF}(l)$, we apply FFN-ONLY, skipping visual attention while preserving visual FFN. If $\mathcal{RF}(l) < \mathcal{RA}(l)$, we apply ATTENTION-ONLY, preserving visual attention while skipping visual FFN. If $\mathcal{RAF}(l)$ is sufficiently small, we apply the more aggressive FROZEN policy, skipping both visual operators and directly propagating $H_{\mathcal{V}}^{l}$ to the next layer.

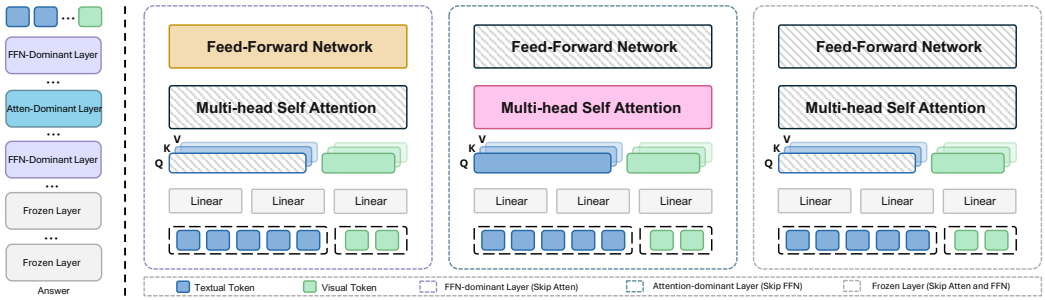

Figure 4: Overview of operator-level visual-token skipping. Each layer is assigned to an FFN-dominant, attention-dominant, or frozen domain based on its visual-token contribution. Accordingly, we skip only the redundant visual-token operator—attention, FFN, or both—while retaining visual key and value projections to preserve text access to the full visual context.

Unlike token pruning, our method preserves the full visual-token sequence and avoids irreversible evidence removal. Unlike layer-level skipping, it does not treat attention and FFN as an inseparable block, but preserves the dominant visual operator while bypassing redundant visual-branch computation.

3.3 Complexity Analysis

Let N_v be the number of visual tokens, d the hidden dimension, d_{ff} the FFN intermediate dimension, and M_V the number of valid query-key pairs associated with visual queries under the original attention mask. Since our method modifies only visual rows and keeps text-token computation unchanged, the computational reduction comes from skipping visual-row attention, visual-row FFN, or both.

$$\Delta\mathcal{C}_{\mathcal{A}} = \Theta\left(N_v d^2 + M_{\mathcal{V}} d\right), \qquad \Delta\mathcal{C}_{\mathcal{F}} = \Theta\left(\kappa N_v d d_{\text{ff}}\right), \tag{15}$$

where $\kappa = 3$ for gated FFNs used in LLaMA- and Qwen-style backbones, and $\kappa = 2$ for standard two-layer FFNs.

$$\Delta\mathcal{C}_{\text{skip}} = \sum_{l \in \mathcal{L}_{\text{FFN}}} \Delta\mathcal{C}_{\mathcal{A}}^{(l)} + \sum_{l \in \mathcal{L}_{\text{Attn}}} \Delta\mathcal{C}_{\mathcal{F}}^{(l)} + \sum_{l \in \mathcal{L}_{\text{Frozen}}} \left(\Delta\mathcal{C}_{\mathcal{A}}^{(l)} + \Delta\mathcal{C}_{\mathcal{F}}^{(l)}\right). \tag{16}$$

Table 2: Main comparison across four MLLM architectures and 10 multimodal benchmarks. Avg. Ret. denotes the mean score ratio to the corresponding vanilla baseline across benchmarks. **Bold** and underlined values indicate the best and second-best results among methods within each architecture.

Model	Method	Rel. TFLOPs	GQA	TextVQA	MME	MMB	MMMU	POPE	SQA	AI2D	OCRB	VizWiz	Avg. Ret.
LLaVA-1.5-7B	Vanilla	100%	61.94	58.21	1866.15	64.18	36.11	85.94	69.46	55.18	31.50	54.09	100.0
	VTW	66%	58.09	53.29	**1853.99**	64.35	35.22	87.03	**69.61**	55.34	18.80	51.79	94.0
	ShortV	66%	60.63	56.14	1827.94	64.60	35.44	86.39	68.77	54.27	28.90	50.46	97.4
	V-Skip	76%	60.77	**57.21**	1766.56	**64.69**	35.56	**87.33**	69.16	55.12	**30.80**	**53.95**	**98.9**
	V-Skip$^+$	76%	61.63	57.21	1840.62	64.60	36.44	87.26	69.26	55.12	30.50	54.16	99.6
	V2Drop	66%	60.51	55.92	1838.27	64.15	31.67	86.82	69.12	54.37	29.70	52.56	97.0
	APET	66%	60.45	56.36	1841.29	64.09	**36.11**	86.82	68.93	**55.54**	29.90	53.23	98.7
	Ours (B=20)	66%	**60.84**	56.40	1841.68	64.52	35.67	87.26	69.16	55.18	30.20	52.75	98.8
LLaVA-NeXT-7B	Vanilla	100%	64.30	61.36	1851.44	67.10	36.00	87.61	70.15	65.35	52.20	60.75	100.0
	VTW	67%	59.43	56.71	**1847.82**	66.43	36.44	87.67	**70.35**	**65.41**	29.70	56.88	93.6
	ShortV	67%	62.97	60.19	1838.47	66.92	36.33	87.62	69.75	64.83	50.00	57.09	98.4
	V-Skip	74%	**63.31**	59.74	1759.67	**67.01**	36.22	87.35	69.83	65.25	48.80	**59.35**	98.2
	V2Drop	67%	61.97	59.43	1841.72	66.94	36.37	87.98	68.52	64.31	50.30	55.34	97.7
	APET	67%	62.31	59.47	1831.51	66.59	**37.11**	88.03	69.35	64.27	49.80	57.36	98.3
	Ours (B=20)	67%	63.01	**60.65**	1797.57	66.75	36.78	**88.31**	69.96	65.03	**50.60**	57.99	**98.8**
Qwen2.5-VL-7B	Vanilla	100%	60.40	77.77	2515.82	83.25	50.00	87.62	87.51	82.42	84.10	70.81	100.0
	VTW	55%	56.68	68.67	1809.82	73.75	45.44	75.13	83.13	74.59	53.70	53.04	84.4
	ShortV	55%	57.52	67.66	2192.59	79.96	47.89	86.52	84.96	76.69	73.60	52.26	91.1
	V-Skip	67%	57.26	63.58	2039.69	80.15	47.00	84.38	85.72	78.04	69.40	63.85	91.0
	V2Drop	55%	58.87	**72.26**	2253.59	80.29	49.89	**87.34**	87.06	**81.54**	56.40	68.68	93.8
	APET	55%	**59.38**	70.90	**2263.83**	**81.70**	**50.11**	86.70	**87.51**	**81.54**	60.00	**69.02**	94.5
	Ours (B=20)	55%	59.29	72.02	2263.38	80.67	46.89	87.32	85.52	79.95	**76.90**	65.48	**95.0**
Qwen3-VL-8B	Vanilla	100%	61.60	80.07	2390.38	84.79	51.33	89.13	94.40	83.78	82.80	69.37	100.0
	VTW	66%	52.69	59.95	2287.31	76.88	48.31	83.01	92.25	74.29	46.20	56.19	85.7
	ShortV	66%	60.68	77.52	2287.31	82.97	52.37	89.09	93.40	82.03	74.60	71.22	98.0
	V-Skip	76%	60.67	75.98	2226.98	81.87	51.33	89.16	92.46	80.60	72.20	68.49	96.3
	V2Drop	66%	60.06	76.44	2290.27	82.56	52.44	88.34	91.72	81.64	63.80	69.78	96.0
	APET	66%	**60.96**	78.36	2332.50	82.47	52.53	88.54	92.56	82.16	75.80	68.82	98.0
	Ours (B=20)	66%	60.70	**78.45**	**2336.01**	**83.08**	**52.89**	**89.43**	**94.35**	**82.55**	**80.40**	**71.70**	**99.5**

Here, FFN-ONLY skips visual attention, ATTENTION-ONLY skips visual FFN, and FROZEN skips both operators. In practice, wall-clock speedup also depends on kernel implementation, memory movement, and hardware utilization. Therefore, we report both theoretical FLOPs and measured first-token latency.

4 EXPERIMENTS

In this section, we conduct systematic experiments across four open-source MLLMs and ten VQA benchmarks to examine the following research questions (RQs): **RQ1**: How well does OpSkip preserve performance under matched computational budgets? **RQ2**: Should skipping decisions consider which operators to skip rather than focusing only on which layers to skip? **RQ3**: How robust are OpSkip's operator assignments to the choice of calibration data? **RQ4**: Do the FLOPs reductions achieved by OpSkip translate into actual inference speedups? **RQ5**: Under what efficiency regimes is operator-level skipping particularly promising?

4.1 EXPERIMENTAL SETUP

Models and Baselines. We evaluate our method on four representative open-source MLLMs, including LLaVA-1.5-7B (Liu et al., 2024a), LLaVA-NeXT-7B (Liu et al., 2024b), Qwen2.5-VL-7B (Bai et al., 2025b), and Qwen3-VL-8B (Bai et al., 2025a). LLaVA-1.5-7B uses a fixed-resolution visual encoder and represents each image with 576 visual tokens, while LLaVA-NeXT and the Qwen-VL models support more flexible high-resolution image processing, allowing us to test generalization across architectures. We compare with training-free layer-wise visual skipping methods VTW (Lin et al., 2025), ShortV (Yuan et al., 2025), and V-Skip (Ma et al., 2026a), as well as token-reduction methods V2Drop (Chen et al., 2026) and APET (Ma et al., 2026b). V-Skip$^+$ uses dataset-specific optimal settings, whereas all other methods use one configuration per model across benchmarks. Unlike these coarse-grained baselines, our method preserves all visual tokens and selectively skips attention, FFN, or both according to layer-wise operator dominance. All evaluations use `lmms-eval` under the official benchmark protocols.

Benchmarks and Implementation Details. We evaluate on 10 multimodal benchmarks covering general VQA, text-rich understanding, hallucination sensitivity, multimodal reasoning, and scientific QA, including GQA (Hudson & Manning, 2019), TextVQA (Singh et al., 2019), MME (Fu et al., 2026), MMBench (Liu et al., 2024c), MMMU (Yue et al., 2024), POPE (Li et al., 2023b), ScienceQA (Lu et al., 2022), AI2D (Kembhavi et al., 2016), OCRBench (Liu et al., 2024d), and VizWiz (Gurari et al., 2018). For each model, we estimate layer-wise visual operator risks using a small calibration set with 8 examples from each of GQA, POPE, TextVQA, MME, MMMU,

and ScienceQA. These examples are used only to estimate $\mathcal{R}_{\mathcal{A}}(l)$, $\mathcal{R}_{\mathcal{F}}(l)$, and $\mathcal{R}_{\mathcal{AF}}(l)$. We set the threshold for assigning FROZEN to 0.7% of the corresponding model's $\mathcal{R}_{\mathcal{AF}}$ summed over all layers. Based on these risks, each selected layer is assigned ATTENTION-ONLY, FFN-ONLY, or FROZEN, and the resulting policy is fixed across all samples for the same model. We report relative FLOPs and first-token latency measured on NVIDIA A800 GPUs.

4.2 RESULTS AND ANALYSIS

RQ1: OpSkip better preserves performance under matched compute. As shown in Table 2, OpSkip achieves the highest average performance retention (Avg. Ret.) on three of the four MLLM architectures, while remaining the strongest method under the matched 66% compute budget on LLaVA-1.5-7B. With only 55%–67% of the vanilla TFLOPs, OpSkip retains 95.0%–99.5% of the original performance, demonstrating that substantial reductions in visual computation incur only limited overall degradation. Although OpSkip does not achieve the best score on every task, it consistently preserves performance across diverse capabilities, including visual question answering, OCR, knowledge reasoning, and visual perception. These results show that OpSkip can substantially reduce computation while largely preserving the capabilities of the original model.

RQ2: Layer selection alone is insufficient, operator choice also matters. The results in Table 2 suggest that layer importance alone may be insufficient to characterize redundancy in visual computation. To isolate the effect of operator selection, we conduct a controlled study on Qwen3-VL-8B by fixing the same 28 target layers and varying only which visual operators are skipped: FFN-ONLY skips visual attention, ATTENTION-ONLY skips the visual FFN, FROZEN skips both, and OpSkip assigns one of these actions according to each layer's operator contributions. As shown in Table 3, with the target layers held fixed, OpSkip achieves the best performance across all benchmarks, whereas fixed operator choices incur substantial and highly variable degradation. These results show that visual redundancy cannot be characterized solely by which layers are selected for skipping: even within exactly the same layers, skipping different operators leads to markedly different outcomes.

Table 3: Ablation on Qwen3-VL-8B. All variants simplify the same number of layers.

Method	GQA	TextVQA	MME	MMB	MMMU	AI2D	OCRB	VizWiz
Vanilla	61.60	80.07	2390.38	84.79	51.33	83.78	82.80	69.37
FFN-ONLY	59.56	72.38	2117.67	78.78	48.56	75.39	61.70	66.23
ATTENTION-ONLY	53.63	49.24	1506.10	70.10	47.00	70.79	11.20	58.80
FROZEN	26.78	25.63	975.10	20.02	36.56	34.49	20.00	52.30
Ours	**59.97**	**75.58**	**2158.87**	**80.93**	**51.00**	**79.31**	**76.40**	**69.06**

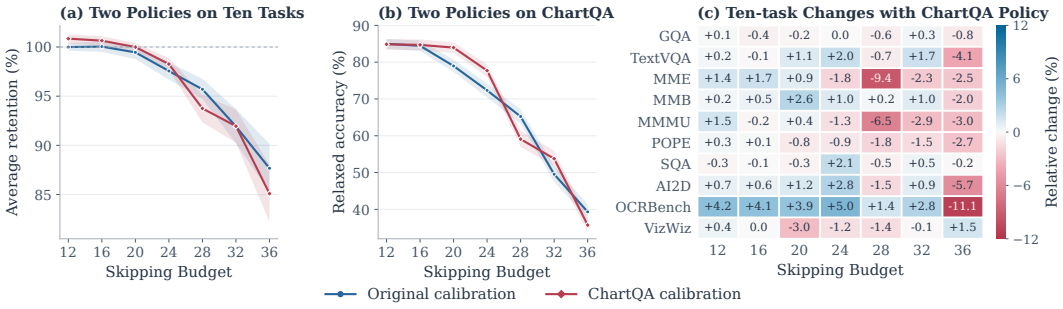

Figure 5: Sensitivity to calibration data on Qwen3-VL-8B. (a) Average retention over 10 benchmarks. (b) Relaxed accuracy on ChartQA. (c) Per-benchmark relative changes from the default six-task policy to the ChartQA policy.

RQ3: OpSkip's operator assignments generalize well across datasets. Table 2 shows that the original policy, calibrated on only six benchmarks, transfers well to four unseen benchmarks (MMB, AI2D, OCRB, and VizWiz). We further calibrate exclusively on ChartQA (Masry et al., 2022) with the same number of samples and disjoint evaluation data. Unlike our main suite, ChartQA emphasizes structured chart understanding and numerical reasoning, providing a distinct calibration domain. As shown in Figure 5, the ChartQA and original policies transfer well across evaluation domains under small and moderate skipping budgets, with no consistent advantage from in-domain calibration. Their differences become more pronounced only under aggressive skipping. These results indicate that OpSkip identifies largely dataset-transferable low-risk operators, while diverse calibration data provides more stable performance as the number of selected layers increases.

RQ4: OpSkip's FLOPs reductions translate into practical inference speedups. As shown in Figure 6, OpSkip consistently reduces prefill latency and time to first token (TTFT) across all four models. Fine-grained operator skipping introduces extra indexing and data-movement overhead, which can offset part of the theoretical compute savings. We therefore remove these redundant operations without changing the skipping policy, yielding OpSkip*, which further improves both prefill and TTFT speedups across models and skipping budgets. Although ShortV achieves higher raw speedups, OpSkip better preserves dominant-operator functionality and thus offers a stronger performance–efficiency trade-off under more aggressive skipping. At higher ShortV budgets, whole-layer skipping often produces corrupted outputs that reach the `lmms-eval` maximum generation length, making evaluation prohibitively slow. Given the available computational resources and evaluation time, we therefore report only the budgets for which average performance could be reliably measured. Figure 6 also shows architecture-dependent speedups, leaving further model-specific optimization for future work while demonstrating the practical acceleration potential of OpSkip.

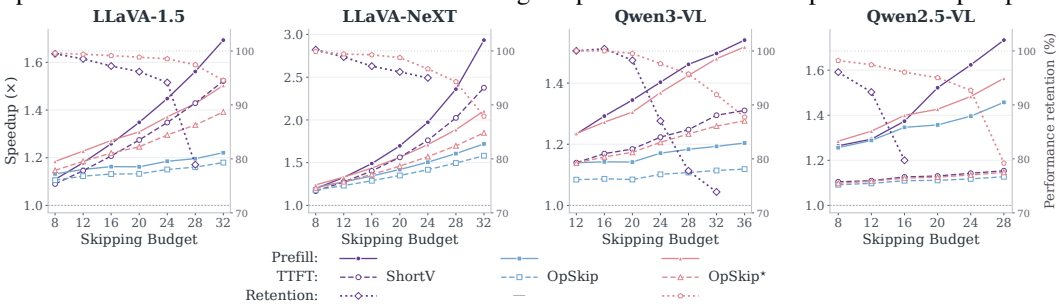

Figure 6: Prefill and TTFT speedups averaged across 10 benchmarks.

RQ5: Operator-level skipping is particularly effective under aggressive compression. Visual-token pruning is a common approach to accelerating prefill by shortening the visual sequence, but aggressive pruning can discard critical visual evidence. OpSkip instead reduces redundant computation on the retained tokens, making the two approaches complementary. As shown in Table 4, combining OpSkip with VisionZip or V2Drop achieves comparable or lower compute than further pruning, while preserving substantially more performance. This suggests that once visual-token retention becomes low, preserving the remaining evidence and skipping redundant operators is a more effective path to further compression than removing additional tokens.

Table 4: Combining operator-level skipping with token pruning on Qwen3-VL-8B.

VisionZip							V2Drop						
Token Ret.	TFLOPs	GQA	TextVQA	MME	MMB	POPE	Token Ret.	TFLOPs	GQA	TextVQA	MME	MMB	POPE
10%	1.67	49.8	61.0	1717.1	69.0	79.5	10%	2.59	54.7	60.4	2012.6	77.7	86.5
3%	$1.33_{\downarrow 0.34}$	$42.0_{\downarrow 7.8}$	$46.6_{\downarrow 14.4}$	$1412.9_{\downarrow 304.2}$	$47.9_{\downarrow 21.1}$	$65.5_{\downarrow 14.0}$	2%	$1.99_{\downarrow 0.60}$	$41.5_{\downarrow 13.2}$	$44.9_{\downarrow 15.5}$	$1482.2_{\downarrow 530.4}$	$44.1_{\downarrow 33.6}$	$67.5_{\downarrow 19.0}$
+Ours (10%)	$1.31_{\downarrow 0.36}$	$49.9_{\uparrow 0.1}$	$60.7_{\downarrow 0.3}$	$1723.7_{\uparrow 6.6}$	$66.3_{\downarrow 2.7}$	$80.3_{\uparrow 0.8}$	+Ours (10%)	$1.31_{\downarrow 1.28}$	$55.6_{\uparrow 0.9}$	$60.9_{\uparrow 0.5}$	$1999.2_{\downarrow 13.4}$	$77.4_{\downarrow 0.3}$	$86.4_{\downarrow 0.1}$

More Results and Analysis. Additional analyses and implementation details are provided in the Appendix.

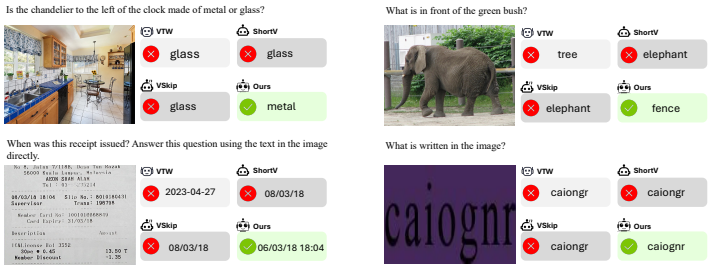

Figure 7: Qualitative visualization of model responses under operator-level visual-token skipping.

5 Conclusion

We demonstrated that visual redundancy in MLLMs is answer-observable and operator-dependent, with late visual-token updates often contributing little to answers. Our operator-level skipping preserves all visual tokens while bypassing redundant attention, FFN, or both, reducing visual computation across multiple backbones with little performance loss, without irreversible evidence loss from pruning.

AI USE STATEMENT

An LLM was employed during manuscript preparation for language editing and presentation refinement. Its use was limited to improving phrasing, correcting grammatical issues, enhancing readability, and organizing the exposition more clearly across different sections.

The LLM did not contribute to the formulation of the research questions, the development of the methodology, the design of the experiments, or the interpretation of the results. All ideas, analyses, and scientific decisions were produced and reviewed by the authors. Accordingly, the model's role was restricted to linguistic assistance rather than scientific reasoning or data analysis.

We remain fully accountable for the entire manuscript, including passages that were generated or revised with LLM assistance. We also reviewed the resulting text to ensure compliance with ethical standards and to avoid plagiarism, misrepresentation, or other forms of scientific misconduct.

ETHICS STATEMENT

This work studies the inference efficiency of publicly available multimodal language models and does not involve human participants, animal subjects, or the collection of new user data. We evaluate only on established public benchmarks—GQA, TextVQA, MME, MMBench, MMMU, POPE, ScienceQA, AI2D, OCRBench, VizWiz, and ChartQA—and use their data solely for calibration and evaluation under the applicable licenses and benchmark protocols. We do not attempt to identify individuals depicted in benchmark images or infer sensitive personal attributes. Because OpSkip preserves the underlying model and changes only which visual-token operators are evaluated, it does not address biases, unsafe outputs, or other limitations inherited from the base MLLMs and their training data. These models should therefore remain subject to the same safety review and deployment safeguards as their original versions.

REPRODUCIBILITY STATEMENT

OpSkip is training-free and uses publicly available checkpoints for LLaVA-1.5-7B, LLaVA-NeXT-7B, Qwen2.5-VL-7B, and Qwen3-VL-8B. Section 4.1 specifies the evaluated benchmarks, baselines, calibration data, computational budgets, and evaluation protocol. The appendix further reports the hardware configuration, model-specific software environments and package versions, layer-wise policies, and the row-wise pseudocode for implementing ATTENTION-ONLY, FFN-ONLY, and FROZEN execution. All benchmark results are produced with `lmms-eval` under its standard task protocols. These details, together with the complete budget-sensitivity and calibration-size results, are intended to support independent reproduction of both the policy construction and the reported evaluations.

REFERENCES

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.

Xiang An, Yin Xie, Kaicheng Yang, Wenkang Zhang, Xiuwei Zhao, Zheng Cheng, Yirui Wang, Songcen Xu, Changrui Chen, Didi Zhu, et al. Llava-onevision-1.5: Fully open framework for democratized multimodal training. *arXiv preprint arXiv:2509.23661*, 2025.

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*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025b. URL https://arxiv.org/abs/2502.13923.

Junjie Chen, Xuyang Liu, Zichen Wen, Yiyu Wang, Siteng Huang, and Honggang Chen. Variation-aware vision token dropping for faster large vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3489–3499, 2026.

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, 2024.

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. *Advances in Neural Information Processing Systems*, 38, 2026.

Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3608–3617. IEEE, 2018.

Yihong Huang, Fei Ma, Yihua Shao, Jingcai Guo, Zitong Yu, Laizhong Cui, and Qi Tian. N\" uwa: Mending the spatial integrity torn by vlm token pruning. *arXiv preprint arXiv:2602.02951*, 2026.

Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6693–6702. IEEE, 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.

Jiwan Kim, Kibum Kim, Wonjoong Kim, Byung-Kwan Lee, and Chanyoung Park. Why and when visual token pruning fails? a study on relevant visual information shift in mllms decoding. In *European Conference on Computer Vision*, pp. 244–263. Springer, 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.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 292–305, 2023b.

Zhihang Lin, Mingbao Lin, Luxi Lin, and Rongrong Ji. Boosting multimodal large language models with visual tokens withdrawal for rapid inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 5334–5342, 2025.

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.

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

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, 2024c.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12):220102, 2024d.

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.

Jie Ma, Zhike Qiu, Jiayi Ji, Xiaoshuai Sun, and Rongrong Ji. Look less, reason more: Block-wise attention skipping for efficient multimodal llms. *arXiv preprint arXiv:2606.08511*, 2026a.

Qiankun Ma, Ziyao Zhang, Haofei Wang, Zhen Song, Jie Chen, and Hairong Zheng. Apet: Approximation-error guided token compression for efficient vlms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26306–26316, 2026b.

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.

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 *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8309–8318. IEEE, 2019.

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, Weijia Li, Conghui He, and Linfeng Zhang. Token pruning in multimodal large language models: Are we solving the right problem? In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 15537–15549, 2025.

Gongli Xi, Ye Tian, Mengyu Yang, Huahui Yi, Liang Lin, Xiaoshuai Hao, Kun Wang, and Wendong Wang. Large vision-language models get lost in attention. *arXiv preprint arXiv:2605.05668*, 2026.

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.

Qianhao Yuan, Qingyu Zhang, Yanjiang Liu, Jiawei Chen, Yaojie Lu, Hongyu Lin, Jia Zheng, Xianpei Han, and Le Sun. Shortv: Efficient multimodal large language models by freezing visual tokens in ineffective layers. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 329–339. IEEE, 2025.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9556–9567, 2024.

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, 2026.

A APPENDIX

A.1 RELATED WORK

Layer-wise visual-computation skipping. As high-resolution inputs, image tiling, and dense visual encoders increase the length of visual prefixes, the LLM prefill stage has become a major bottleneck for MLLM inference (Chen et al., 2024; Yuan et al., 2025; Lin et al., 2025; Ma et al., 2026a). One line of work reduces this cost by modifying how visual tokens are processed across Transformer layers. VTW withdraws visual tokens from later layers to avoid repeatedly propagating visual states through the full LLM backbone (Lin et al., 2025), while ShortV keeps visual-token states fixed in layers identified as visually ineffective (Yuan et al., 2025). V-Skip studies block-wise attention skipping and shows that visual attention computation can be redundant in selected layers (Ma et al., 2026a). Related structural acceleration methods also exploit layer-wise or block-wise redundancy to reduce visual computation without directly deleting all visual inputs (Chen et al., 2024; Ma et al., 2026a). Together, these studies show that visual computation can be redundant in a depth-dependent manner, but they differ in the granularity at which redundancy is modeled. VTW and ShortV primarily act on full visual-token states, whereas V-Skip focuses on attention computation. Our work is complementary, instead of treating a layer or a single operator family as the default unit of analysis, we compare the answer-observable risks of visual-row attention and visual-row FFN within each layer. This operator-level view allows the policy to preserve the operator that remains useful at a given depth while skipping the redundant one, or to use a frozen visual update when both operators are answer-silent.

Visual-token pruning and compression. Another major direction accelerates MLLM inference by reducing the number of visual tokens. Early and recent methods prune, merge, or select visual tokens according to attention scores, token similarity, spatial redundancy, diversity, or output sensitivity (Chen et al., 2024; Shang et al., 2025; Yang et al., 2025; Xing et al., 2024; Zhang et al., 2026; Alvar et al., 2025; Ma et al., 2026b; Chen et al., 2026). These approaches directly shorten the visual sequence and can reduce both attention and FFN costs in subsequent layers. However, token reduction is inherently destructive, once visual tokens are removed, merged, or prematurely withdrawn, the corresponding spatial evidence cannot be recovered by later textual reasoning. This limitation is especially pronounced in fine-grained recognition tasks, including OCR-centric VQA, chart or diagram understanding, small-object reasoning, and visually grounded question answering, where the decisive evidence may occupy only a small image region or become relevant only after interacting with the question. Recent studies further show that aggressive token pruning can damage spatial integrity, shift relevant visual information during decoding, and even underperform random selection in certain settings (Wen et al., 2025; Huang et al., 2026; Kim et al., 2026; Wang et al., 2026). In contrast, our method preserves the full visual-token sequence throughout inference and reduces computation at the operator level. This avoids irreversible loss of fine-grained visual evidence while still removing answer-silent visual updates, matching the motivation in the introduction: efficient MLLM inference should preserve visual information and reduce only computation that is not observable from the answer side.

A.2 LIMITATIONS

Our results show that a subset of attention and FFN computations in MLLMs can be selectively omitted with only marginal changes in model outputs, indicating that the computation allocated by current dense architectures is not always fully utilized. However, our method exploits such redundancy only after training and does not address the training dynamics or architectural factors that give rise to it. A more fundamental direction is to incorporate operator-level objectives, regularization, or architectural constraints during pretraining, so that attention and FFN modules learn complementary and functionally specialized roles. Such designs could reduce underutilized computation at its source and ensure that each executed operator provides meaningful marginal utility for multimodal reasoning. This may enable more compact architectures, more effective use of model capacity, and improved performance–efficiency trade-offs.

B Implementation Details

B.1 Hardware and System Setup

All experiments were conducted on an internal GPU cluster equipped with NVIDIA A800-SXM4 GPUs. Each compute node contains two Intel(R) Xeon(R) Platinum 8358 CPUs at 2.60GHz, providing 128 logical CPU cores in total, approximately 1 TiB of system memory, and 8 NVIDIA A800-SXM4 GPUs with 80GB VRAM each. The NVIDIA driver version is 535.161.08. The system CUDA toolkit is CUDA 12.2, while the PyTorch binaries used in our conda environments are built with CUDA 12.1.

B.2 Software Environments

We use the official `lmms-eval` framework for all reported multimodal benchmark results and follow its default evaluation protocols. Since different model families rely on different dependency constraints, we maintain separate conda environments for Qwen2.5-VL, Qwen3-VL, LLaVA-1.5, and LLaVA-NeXT. Model checkpoints are loaded from local copies of the corresponding public Hugging Face repositories. The evaluated benchmark suite includes MME, MMBench, OCRBench, POPE, GQA, TextVQA, ScienceQA, MMMU, AI2D, and VizWiz when applicable.

Table 5: Conda environments used for different model families.

Model family	Python	PyTorch	Torchvision	Transformers	Accelerate	lmms-eval
Qwen2.5-VL	3.12.12	2.4.1	0.19.1	4.51.3	1.13.0	0.6.1
Qwen3-VL	3.12.12	2.4.1	0.19.1	5.5.4	1.13.0	0.6.1
LLaVA-1.5	3.12.9	2.4.1	0.19.1	4.37.2	0.21.0	0.6.1
LLaVA-NeXT	3.10.20	2.1.2+cu121	0.16.2+cu121	4.40.0.dev0	0.29.3	0.6.1

Table 6: Additional package versions used in each environment.

Model family	`Qwen2.5-VL`	`Qwen3-VL`	`LLaVA-1.5`	`LLaVA-NeXT`
`datasets`	2.16.1	2.16.1	2.16.1	5.0.0
`tokenizers`	0.21.4	0.22.2	0.15.1	0.15.2
`sentencepiece`	0.2.1	0.2.1	0.2.1	0.2.1
`qwen-vl-utils`	0.0.14	0.0.14	–	–
`llava`	–	–	1.2.2.post1	1.7.0.dev0

B.3 Implementation of Operator-Level Skipping

Visual operator skipping. Algorithm 1 gives the row-wise implementation of the three policies defined in Section 3.2. Under FFN-ONLY, the visual rows bypass attention, while the FFN is applied to the full intermediate state $\bar{H}^l$. Under ATTENTION-ONLY, attention updates the full sequence, while the FFN is evaluated only on the non-visual rows $\mathcal{T}$. Under FROZEN, both operators are skipped on $\mathcal{V}$, giving $H^{l+1}_{\mathcal{V}} = H^l_{\mathcal{V}}$. The FULL policy recovers the original Transformer layer. Non-visual rows receive both operator updates under every policy.

What computation is omitted. The ATTEND routine forms queries only for the requested row set $\mathcal{R}$. Rows outside $\mathcal{R}$ therefore incur no query projection, attention-score computation, value aggregation, or output projection. Keys and values are still constructed for the full sequence, so active text queries can attend to every visual token under the original attention mask. Likewise, evaluating the FFN only on $\bar{H}^l_{\mathcal{T}}$ removes its projections, activation, and output computation for the visual rows. OpSkip therefore preserves the complete visual context while omitting only the visual-row updates specified by the layer policy.

Algorithm 1 OpSkip: operator-level visual-token skipping at layer l

Require: Layer input H^l; visual indices $\mathcal{V}$; policy s_l; position indices p; attention mask M
Ensure: Layer output H^{l+1}; keys K^l and values V^l

$\mathcal{I} \leftarrow \{0, \dots, S-1\}$; $\mathcal{T} \leftarrow \mathcal{I} \setminus \mathcal{V}$
$Z^l \leftarrow \mathrm{Norm}_l^{\mathrm{attn}}(H^l)$
$K^l \leftarrow \mathrm{Key}_l(Z^l, p)$; $V^l \leftarrow \mathrm{Value}_l(Z^l)$
if $s_l =$ FFN-ONLY **then** ▷ Skip visual attention
  $\bar{H}^l \leftarrow H^l$
  $\bar{H}^l_{\mathcal{T}} \leftarrow H^l_{\mathcal{T}} + \textsc{Attend}(\mathcal{T})$
  $H^{l+1} \leftarrow \bar{H}^l + \mathrm{FFN}_l(\mathrm{Norm}_l^{\mathrm{ffn}}(\bar{H}^l))$
else if $s_l =$ ATTENTION-ONLY **then** ▷ Skip visual FFN
  $\bar{H}^l \leftarrow H^l + \textsc{Attend}(\mathcal{I})$
  $H^{l+1} \leftarrow \bar{H}^l$
  $H^{l+1}_{\mathcal{T}} \leftarrow \bar{H}^l_{\mathcal{T}} + \mathrm{FFN}_l(\mathrm{Norm}_l^{\mathrm{ffn}}(\bar{H}^l_{\mathcal{T}}))$
else if $s_l =$ FROZEN **then** ▷ Skip both visual operators
  $\bar{H}^l \leftarrow H^l$
  $\bar{H}^l_{\mathcal{T}} \leftarrow H^l_{\mathcal{T}} + \textsc{Attend}(\mathcal{T})$
  $H^{l+1} \leftarrow \bar{H}^l$
  $H^{l+1}_{\mathcal{T}} \leftarrow \bar{H}^l_{\mathcal{T}} + \mathrm{FFN}_l(\mathrm{Norm}_l^{\mathrm{ffn}}(\bar{H}^l_{\mathcal{T}}))$
else if $s_l =$ FULL **then** ▷ Retain both operators
  $\bar{H}^l \leftarrow H^l + \textsc{Attend}(\mathcal{I})$
  $H^{l+1} \leftarrow \bar{H}^l + \mathrm{FFN}_l(\mathrm{Norm}_l^{\mathrm{ffn}}(\bar{H}^l))$
end if
return H^{l+1}, K^l, V^l

function $\textsc{Attend}(\mathcal{R})$ ▷ Compute attention only for selected rows
  $Q^l_{\mathcal{R}} \leftarrow \mathrm{Query}_l(Z^l_{\mathcal{R}}, p_{\mathcal{R}})$
  $A^l_{\mathcal{R}} \leftarrow \mathrm{Attention}_l(Q^l_{\mathcal{R}}, K^l, V^l; M_{\mathcal{R},:})$
  return $\mathrm{Out}_l(A^l_{\mathcal{R}})$
end function

B.4 Layer-Wise Policy Configuration

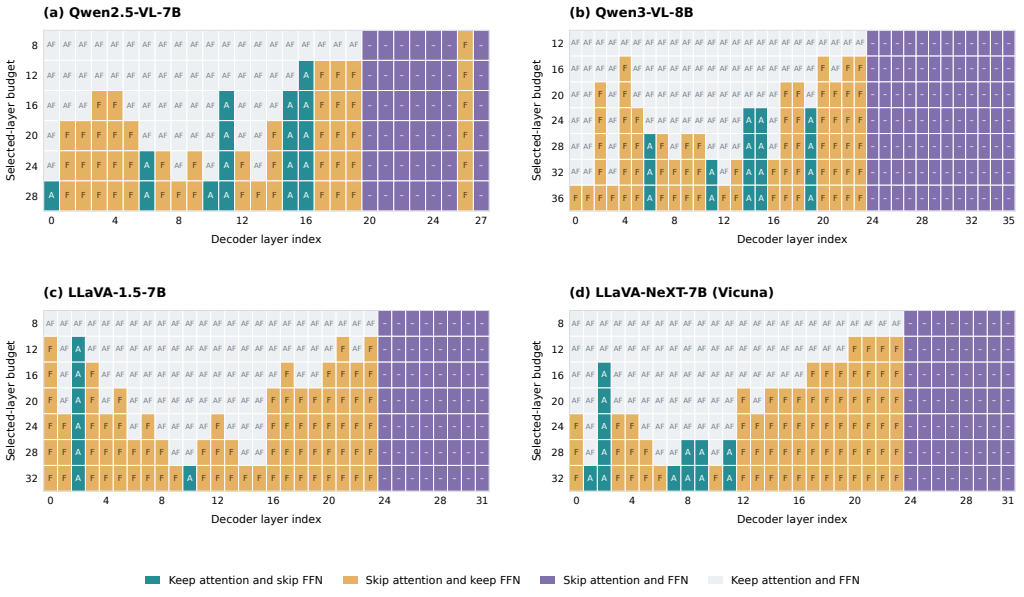

Figure 8: Layer-wise skipping policies under different budgets. Each cell indicates whether a layer keeps both visual operators (AF), keeps attention while skipping the FFN (A), keeps the FFN while skipping attention (F), or skips both operators (–).

Figure 8 visualizes how the operator-risk structure in Figure 3 is converted into model-specific skipping configurations. Layers in the attention-dominant region retain attention and skip the FFN, whereas FFN-dominant layers retain the FFN and skip attention. Layers with low risks for both operators are frozen by skipping both updates. As the budget increases, the policy expands from the lowest-risk layers to additional operator-dominant layers, while many late layers consistently remain frozen. The resulting patterns differ across architectures, confirming that the policy follows the measured layer-wise risks rather than imposing a fixed depth-based schedule.

C Additional Results and Analysis

C.1 Calibration Sample Size Study

We study how the number of calibration samples from each benchmark affects the analysis decisions made by our method. For each of the six calibration benchmarks used in the main experiments, we randomly sample 8, 16, or 20 examples and estimate the layer-wise operator risks for LLaVA-1.5-7B, Qwen2.5-VL-7B, and Qwen3-VL-8B. The resulting risk profile determines the selected layers and their operator-skipping policies, and the same policy is then evaluated on the full benchmark suite.

Table 7 reports the average retention under different layer budgets. The goal is to test whether using more calibration samples leads to a substantially more reliable policy. Since calibration is used only for risk estimation and does not require labels or model training, a small sample size is desirable if it already supports stable policy selection.

Table 7: Effect of calibration sample size under different layer budgets. Each setting samples the same number of examples from each of six calibration benchmarks.

Model	Samples Per Benchmark	Avg. Ret. under Layer Budget							
		8	12	16	20	24	28	32	36
LLaVA-1.5-7B	8	99.58	99.25	99.15	98.76	98.46	97.37	94.48	–
	16	99.62	99.37	99.28	98.81	98.52	97.49	94.32	–
	20	99.74	99.55	99.52	99.01	98.57	97.60	94.08	–
Qwen2.5-VL-7B	8	98.16	97.42	96.14	94.97	92.66	79.23	–	–
	16	98.18	97.51	96.17	95.10	92.68	79.26	–	–
	20	97.92	97.76	96.65	95.19	92.63	79.31	–	–
Qwen3-VL-8B	8	–	100.00	100.04	99.45	97.55	95.73	91.93	87.67
	16	–	99.99	100.01	99.32	97.56	94.91	91.49	86.75
	20	–	100.06	100.10	99.15	97.16	93.60	88.42	84.22

The results show that increasing the number of calibration samples can bring improvements at some layer budgets, but the gains are not consistently large or monotonic. For example, more samples slightly improve several LLaVA-1.5-7B settings, while the trends for Qwen2.5-VL-7B and Qwen3-VL-8B vary across budgets. This suggests that 8 samples per benchmark are already sufficient to support the risk analysis used by our method. Adding more samples can provide stronger evidence in some cases, but it is not always necessary for obtaining stable operator-skipping decisions.

C.2 Ablation on Risk-Based Layer Allocation

We compare our budgeted layer allocation with a simple last-layer heuristic. As described in Section 3.2, our policy first ranks layers by the summed single-operator risk $\mathcal{R}_{\mathcal{A}}(l) + \mathcal{R}_{\mathcal{F}}(l)$ and allocates the skipping budget to the lowest-risk layers. This design assumes that layers with smaller summed risk can safely discard their non-dominant visual operator earlier, whereas layers with larger summed risk should be postponed because even their non-dominant operator may still affect the output.

Table 8: Ablation against a heuristic that skips visual computation in the last 20 layers on Qwen3-VL-8B.

Method	GQA	TextVQA	MME	MMB	SQA	AI2D	OCRB
Last-20 heuristic	60.57	77.40	2282.61	82.47	93.01	81.44	78.80
Ours (B=20)	60.70	78.45	2336.01	83.08	94.35	82.55	80.40

This ablation isolates the effect of this layer-allocation rule. For both methods, we keep the per-layer operation type fixed; each selected layer uses the same skip-operator decision or the same frozen update determined by the operator-risk profile. The only difference is which layers are selected. Our method selects the 20 layers with the smallest $\mathcal{R}_{\mathcal{A}}(l) + \mathcal{R}_{\mathcal{F}}(l)$, while the Last-20 heuristic directly selects the final 20 Transformer layers regardless of their risk scores. Table 8 reports the Qwen3-VL-8B comparison, testing whether risk-based allocation is preferable to choosing late layers purely by position.

The results support our hypothesis. With the same number of selected layers and the same per-layer operator actions, risk-based allocation consistently outperforms the Last-20 heuristic across all reported benchmarks. This indicates that small-$\mathcal{R}_{\mathcal{A}} + \mathcal{R}_{\mathcal{F}}$ layers are indeed safer places to discard non-dominant visual operators, while simply choosing the final layers may remove useful visual computation whose operator risk remains high.

C.3 Different Budgets for OpSkip

Table 9 studies how the operator-skipping budget affects performance across four MLLM backbones. Overall, the results show that our operator-level policy is not tied to a particular architecture: LLaVA-1.5, LLaVA-NeXT, Qwen2.5-VL, and Qwen3-VL all maintain high average retention under moderate budgets, despite their different visual encoders, language backbones, and dependency stacks. In particular, small and medium budgets usually preserve nearly all vanilla performance, and several individual benchmarks even improve over the budget-0 setting. This suggests that the skipped operators are often answer-silent rather than uniformly useful, supporting the reliability of the proposed answer-observable selection criterion.

Table 9: Sensitivity to the operator-skipping budget across the four evaluated MLLMs. Avg. Ret. is the mean retention relative to the vanilla model. Bold denotes the best score for each benchmark across skipping budgets.

Budget	GQA	TextVQA	MME	MMB	MMMU	POPE	SQA	AI2D	OCRB	VizWiz	Avg. Ret.
LLaVA-1.5-7B (32 Layers)											
0	**61.94**	**58.21**	1866.15	64.18	**36.11**	85.94	69.46	55.18	**31.50**	**54.09**	100.0
8	61.14	57.18	**1878.10**	64.18	35.67	86.43	**69.61**	**55.51**	31.40	53.23	99.6
12	61.10	57.12	1874.85	64.26	35.44	86.82	69.36	55.34	31.00	52.73	99.3
16	61.02	56.30	1856.99	64.18	35.89	86.93	69.36	55.18	31.30	52.45	99.1
20	60.84	56.40	1841.68	64.52	35.67	**87.26**	69.16	55.18	30.20	52.75	98.8
24	60.71	56.22	1797.31	64.61	35.78	87.22	69.16	55.28	29.60	53.46	98.5
28	60.23	55.61	1784.78	**65.38**	34.67	86.94	69.16	54.53	28.30	53.06	97.4
32	57.89	53.80	1728.08	62.03	35.67	85.01	69.21	52.46	25.10	52.76	94.5
LLaVA-NeXT-7B (32 Layers)											
0	**64.30**	61.36	**1851.44**	**67.10**	36.00	87.61	**70.15**	65.35	**52.20**	60.75	100.0
8	63.75	**61.37**	1845.83	**67.10**	36.33	87.54	**70.15**	**65.48**	51.40	**61.05**	99.9
12	63.55	61.08	1837.72	67.01	35.67	87.57	70.10	65.32	51.30	60.06	99.4
16	63.36	60.77	1832.66	**67.10**	36.33	87.57	69.96	65.19	51.20	59.45	99.3
20	63.01	60.65	1797.57	66.75	**36.78**	**88.31**	69.96	65.03	50.60	57.99	98.8
24	62.30	58.36	1777.99	66.15	36.00	88.20	69.26	63.76	45.70	58.21	96.7
28	61.75	56.49	1744.20	64.18	35.00	86.87	68.67	63.02	39.50	59.73	94.3
32	59.64	50.67	1616.91	59.54	34.00	85.26	66.48	61.27	26.30	58.67	87.9
Qwen2.5-VL-7B (28 Layers)											
0	**60.40**	**77.77**	**2515.82**	**83.25**	**50.00**	**87.62**	**87.51**	82.42	**84.10**	**70.81**	100.0
8	59.85	76.62	2346.15	82.56	49.11	87.29	87.41	**82.58**	81.80	68.25	98.2
12	59.60	74.52	2343.01	82.65	49.33	87.08	87.06	82.22	80.50	66.77	97.4
16	59.38	73.41	2281.94	81.70	48.22	86.43	86.91	81.41	78.40	66.16	96.1
20	59.29	72.02	2263.38	80.67	46.89	87.32	85.52	79.95	76.90	65.48	95.0
24	58.75	70.28	2159.23	79.38	46.56	86.52	82.94	78.82	70.60	64.84	92.7
28	57.27	54.75	1884.70	69.07	39.33	86.29	77.94	62.08	42.30	54.81	79.2
Qwen3-VL-8B (36 Layers)											
0	**61.60**	**80.07**	**2390.38**	**84.79**	51.33	89.13	94.40	**83.78**	**82.80**	69.37	100.0
12	60.98	79.93	2352.81	84.62	52.44	88.83	**94.70**	83.65	81.90	70.78	100.0
16	61.04	79.64	2356.06	84.45	**53.67**	88.76	**94.70**	83.65	81.40	70.12	100.0
20	60.70	78.45	2336.01	83.08	52.89	**89.43**	94.35	82.55	80.40	**71.70**	99.5
24	60.24	77.01	2273.65	82.05	53.00	89.31	90.53	80.18	77.90	69.77	97.6
28	59.97	75.58	2158.87	80.93	51.00	89.19	88.60	79.31	76.40	69.06	95.7
32	59.59	72.67	1909.52	78.01	49.44	88.42	85.72	76.39	71.20	66.86	91.9
36	58.51	69.91	1707.86	75.34	48.22	87.42	81.85	72.38	63.80	63.77	87.7

The budget trend also reveals a consistent efficiency–accuracy pattern. As the budget increases, average retention decreases smoothly rather than collapsing abruptly, indicating that the policy removes redundant visual computation in a controlled manner. This behavior is especially important for practical deployment, users can choose conservative budgets when accuracy is the priority, or larger budgets when higher efficiency is needed. Across all models, moderate budgets retain strong performance, while aggressive settings expose the trade-off when too much visual computation is skipped. These results show that operator-level visual skipping is broadly applicable and robust under different layer-level budgets.

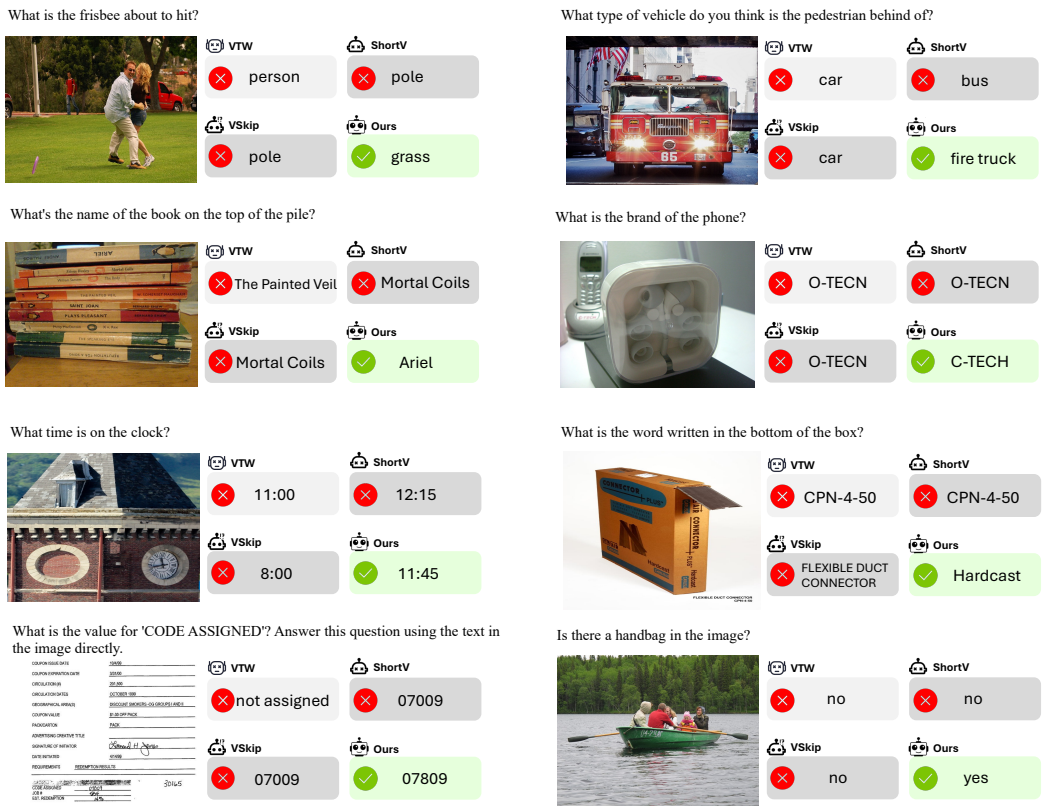

Figure 9: Additional qualitative examples of operator-level visual-token skipping.