VISION-LANGUAGE MODELS DO NOT KEEP LOOKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive Large Vision-Language Models solve multimodal reasoning tasks by retaining hundreds or thousands of visual tokens, allowing every generated token to attend to the visual tokens at any time during the generation process. However, whether continuous access to these visual tokens is necessary remains poorly understood. In this work, we investigate how visual token influence evolves along two dimensions: generation time and transformer depth. Using long-form visual reasoning benchmarks and multiple model scales, we first analyze the attention dynamics across generation time and transformer depth. We identify a temporal and layer-wise structure in the use of visual information. Visual attention peaks during the initial generation steps and then rapidly declines as attention shifts toward previously generated reasoning tokens. Across depth, visual influence is concentrated in intermediate layers and becomes very weak in the final layers. Motivated by these observations, we propose a simple intervention that removes all visual tokens across generation time and model depth. These interventions confirm a transition from early multimodal integration, during which visual information is incorporated into the generated token representations, to later generation, where the model relies primarily on these accumulated representations rather than direct access to the visual tokens. Thus, retaining the complete visual context at every layer and decoding step may be unnecessary. Our experiments demonstrate that we can remove up to 59% of visual tokens across generation time and 11% across model depth, while incurring only a negligible performance drop.

1 INTRODUCTION

Large Vision-Language Models (LVLMs)(Bai et al., 2025b;a) have rapidly evolved into systems capable of solving multimodal mathematics, visual logic, diagram understanding, and other reasoning tasks. In these settings, a model must not only recognize visual content but also sustain a long chain of reasoning grounded in that content. Existing LVLMs typically support this process by inserting hundreds or thousands of visual tokens into the context of the Large Language Model (LLM) they contain, and retaining them throughout autoregressive generation. As a result, every generated token can, in principle, attend back to the original image tokens at every transformer layer.

However, it remains unclear whether LVLMs actually use visual information in this way. Does a model repeatedly revisit the image throughout a long autoregressive reasoning trajectory? Are visual tokens equally important across the depth of the network during reasoning, or is their contribution concentrated in only a subset of layers? These questions are important for both understanding and efficiency. Mechanistically, they concern how perception and reasoning function inside LVLMs. Computationally, retaining the full visual context increases the key–value cache memory and attention computation cost at every decoding step, even when the model may no longer rely on the image. A fundamental question therefore arises: *how does the influence of visual information evolve throughout autoregressive generation and model depth, and must visual tokens remain accessible during the entire reasoning process?* A follow-up question is then: *what trade-off exists between removing all visual tokens and preserving model performance?*

Existing studies have shown that LVLM predictions can be strongly influenced by the LLM they contain, owing to the strong linguistic priors acquired during LLM pretraining (Pezeshki et al., 2021; Huang et al., 2022; Parcalabescu & Frank, 2025; Deng et al., 2025; Park et al., 2025; Lin et al., 2024; Tong et al., 2024). Nevertheless, these works do not explain how visual influence changes as a long autoregressive reasoning response unfolds, how the visual influence on reasoning is distributed

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.

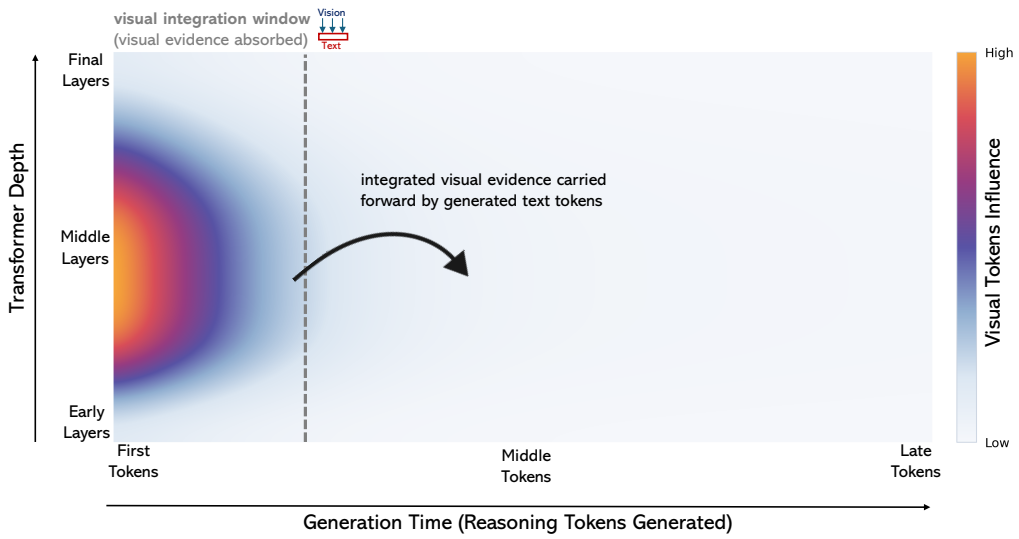


Figure 1: **Temporal and layer-wise dynamics of visual tokens influence in LVLM reasoning**. Visual tokens have their strongest impact during early generation steps and in intermediate transformer layers, forming a visual integration window where image information is incorporated into generated textual representations. Later tokens and final layers rely increasingly on the the visual information accumulated in the textual context, rather than direct access to the original visual tokens.

across transformer layers, or whether visual information is repeatedly retrieved from the original image tokens during generation. As a result, the internal division of labor between vision-language reasoning remains poorly understood.

In this work, we study the role of visual tokens along two axes: the generation time axis, which traces their influence over decoding steps, and the depth-wise axis, which traces the visual contribution through the transformer layers. We focus on reasoning benchmarks that elicit long-form outputs, where the interaction between visual grounding and language generation can be observed over an extended trajectory. Our **first finding** is that visual tokens influence is strongest in the early generation steps. Across tasks and model scales, we discover a *visual integration window*. During this window, information from the image is incorporated into the hidden representations of the first few generated tokens. Subsequent tokens can then access this information indirectly by attending to the growing textual context rather than repeatedly consulting the original visual tokens. Our **second finding** is that visual influence is also localized in depth during reasoning. We find visual token influence to be concentrated primarily in intermediate layers during reasoning, while its contribution becomes much less pronounced in the final layers across the entire decoding trajectory, regardless of the generation step. The final layers therefore operate predominantly on text representations, processing and selecting the next token after multimodal integration has already occurred in earlier layers.

In summary, our contributions are as follows: (i) We analyze the visual influence in LVLMs across both time and depth in long-form reasoning benchmarks. (ii) Based on our observations, we propose causal interventions where we drop all image tokens during different times of the multi-modal reasoning trajectory, and across different layer depths. We also provide trade-offs between visual token reduction rates and performance, across both time and depth, and across different model scales. (iii) Overall, we challenge the prevailing assumption that long-form multimodal reasoning requires continuous access to the complete visual context, both in time and in depth, and find that retaining all visual tokens in all layers and at every generation step is unnecessary. Our work identifies a different temporal and layer-wise structure in the use of visual information in LVLMs.

2 Related Work

Visual reliance in Vision-Language Models: Recent studies have investigated the extent to which LVLMs genuinely rely on visual information. Since the language component is typically pretrained

extensively before multimodal alignment, LVLMs can exhibit strong language-driven behavior even for multimodal tasks (Pezeshki et al., 2021; Huang et al., 2022). Several works have analyzed this phenomenon through attribution-based approaches, showing that the contribution of visual inputs to model predictions can vary substantially across tasks and examples (Parcalabescu & Frank, 2025). Other studies have introduced benchmarks and evaluation protocols to measure visual grounding, multimodal dependence, and susceptibility to language-only reasoning shortcuts (Deng et al., 2025; Park et al., 2025; Lin et al., 2024; Tong et al., 2024). However, long-form multimodal reasoning introduces a temporal dimension that is particularly important yet remains largely unexplored. Existing analyses do not examine how visual influence evolves throughout an autoregressive reasoning trajectory, whether visual information is repeatedly retrieved from the original image tokens during generation, or how visual contribution is distributed across transformer depth during reasoning. Our study aims to answer these questions while also transferring this analysis to a practical setup of interventions.

Visual token pruning: A complementary line of work focuses on reducing the computational cost of LVLM inference by compressing or pruning visual tokens. Methods such as visual token pruning and selection aim to identify redundant image tokens and retain only a subset of them (Chen et al., 2024a; Zhang et al., 2025; Wu et al., 2026) without substantially affecting performance, particularly for short-form visual question answering (VQA) tasks. While existing pruning methods focus on identifying and removing subsets of non-informative visual tokens, our work investigates a fundamentally different question: whether the visual context as a whole remains necessary throughout long-form reasoning, rather than which individual visual tokens can be deemed redundant. Rather than identifying a subset of unimportant visual tokens and pruning them, we remove all visual tokens at different generation steps and transformer layers during multimodal reasoning, to determine when and where direct visual access becomes unnecessary. Furthermore, we move beyond short-form VQA settings where typically a single token or brief phrase is generated and the relevant visual context is in close contextual proximity, facilitating direct access to visual evidence during generation. Instead, we investigate long-form multimodal reasoning, where generation spans many decoding steps and later reasoning stages may still depend on visual evidence.

3 Attention Dynamics of Vision-Language Models

Notations: We consider an autoregressive vision-language model M comprising a visual encoder M_v, a projection module M_{pr}, and a language decoder M_{LLM} of L layers, where M_{LLM} is a large language model (LLM). We use l to denote a layer and a_l^h to denote the attention map of layer l and head h. Given an input image I, the visual encoder divides it into N patches and encodes them into visual features. These features are then projected by M_{pr} into the language embedding space of dimension d, yielding the visual representation $f_v = M_{pr}(M_v(I))$, where $f_v \in \mathbb{R}^{N\times d}$. We also denote f_v^l as the visual representations at a layer l. Similarly, a textual prompt p comprising the problem statement and a reasoning instruction is tokenized into T tokens and embedded into a textual representation $f_t \in \mathbb{R}^{T\times d}$. The visual and textual representations are then concatenated along the sequence dimension to form the input to the language decoder M_{LLM}, i.e., $f = [f_v; f_t]$, where $f \in \mathbb{R}^{(N+T)\times d}$. The language decoder M_{LLM} autoregressively produces a reasoning response g composed of Y tokens, and g_y denotes the y-th generated reasoning token.

Benchmarks and Models: We run experiments on four multimodal reasoning benchmarks: Wemath (Qiao et al., 2024), HallusionBench (Guan et al., 2024), MMStar (Chen et al., 2024b) and LogicVista (Xiao et al., 2024). WeMath assesses human-like mathematical reasoning through hierarchical visual-mathematics problems. HallusionBench evaluates reasoning robustness against entangled visual illusions and language hallucinations. MMStar measures multimodal capabilities across a balanced set of perception and reasoning-oriented tasks. LogicVista focuses on logical reasoning grounded in diverse visual contexts. Experiments are conducted using the 2B, 4B, 8B, and 32B variants of Qwen-3-VL (Bai et al., 2025a). We detail the prompts we used in Appendix F.

3.1 Attention Dynamics across Generation Time

We analyze the attention dynamics of the generated reasoning chain g across generation time. For each reasoning token generated g_y, we examine its interaction with (i) the image tokens f_v and

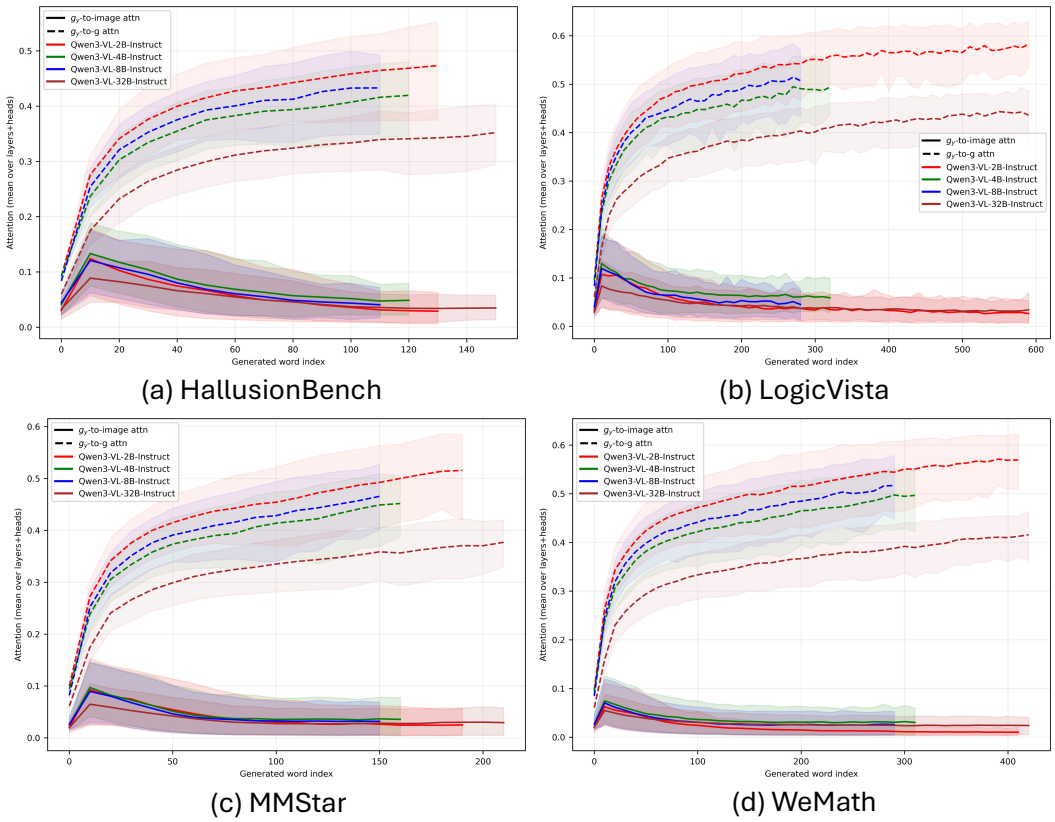


Figure 2: Attention Dynamics across Generation time using layer-head average analysis, for all models across (a) Hallusionbench, (b) LogicVista, (c) MMStar, and (d) WeMeth. For each model, we also include the variance across samples in the benchmark as a shaded region around the corresponding curve.

(ii) the previously generated reasoning tokens $g_{0\ldots y-1}$, which we denote as $g_{<y}$. Concretely, at a layer l and an attention head h, the attention map a_l^h for token g_y distributes attention across the N image tokens (f_v), the T text prompt tokens (p), and the preceding generated tokens $g_{<y}$. We denote $a_l^h(S) := \{a_l^h[i]\}_{i\in S}$ as the attention placed on a token subset S. We decompose a_l^h for g_y into three components (three subsets): (1) attention allocation to the image tokens, $a^{img} = a_l^h(f_v)$, (2) attention to the text prompt, $a^{pmt} = a_l^h(p)$, and (3) attention to previously generated reasoning tokens, $a^{gen} = a_l^h(g_{<y})$, where y is the index of the current generated token. Figure 2 plots a^{img} and a^{gen} attention scores as functions of the generated reasoning tokens for all four benchmarks across different models, averaging over all layers and heads and then averaging across all samples in the corresponding dataset. Because the attention weights sum to one, the attention from g_y to the T text tokens in the prompt p can be recovered by $a^{pmt} = 1 - \left(\sum_{f_v} a^{img} + \sum_{g_{<y}} a^{gen}\right)$; because our focus is the contribution of image attention as tokens are generated, we omit this term from the plots. Plotting details are provided in Section E of the Supp. material.

Across all four benchmarks and models, attention to the image tokens shows a consistent early peak, reaching its maximum within roughly the first few tens of generated tokens before steadily decreasing as generation proceeds, whereas attention to previously generated tokens rises sharply during the same initial period and continues increasing throughout the response, becoming dominant during the later reasoning process. The curves suggest a relatively short *visual integration window*, concentrated within the first few tens of generated tokens. During this early stage, the generated tokens attend strongly to the visual tokens, allowing visual information to be directly integrated into their emerging textual representations. After this window, the model appears to rely increasingly on previously generated reasoning tokens: the visual information encoded in these earlier representations

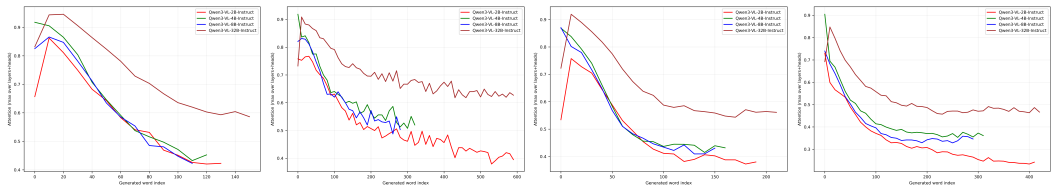


Figure 3: Attention Dynamics across Generation time using upper bound analysis, for all models across (a) Hallusionbench, (b) LogicVista, (c) MMStar, and (d) WeMeth.

is propagated forward to subsequent tokens, while direct attention to the original visual tokens diminishes. This behavior indicates that the visual tokens become progressively redundant as decoding proceeds, suggesting that maintaining full access to all visual tokens throughout the entire generation process may be unnecessary after the visual integration window.

In order to obtain an *upper bound* of the visual token influence on the generated tokens, for each generated token g_y, we take the maximum value across all layers and heads and average across all samples in the dataset. Results are shown in Figure 3. Even with the upper bound, the visual token influence gradually drops across all models and benchmarks, further supporting our finding. Interestingly, we also observe that larger models tend to exhibit higher maximum visual token influence than smaller models, suggesting that larger models rely less heavily on language priors.

Observation 1: Visual information is integrated during the first few tens of generated tokens, after which reasoning becomes increasingly dominated by previously generated textual context.

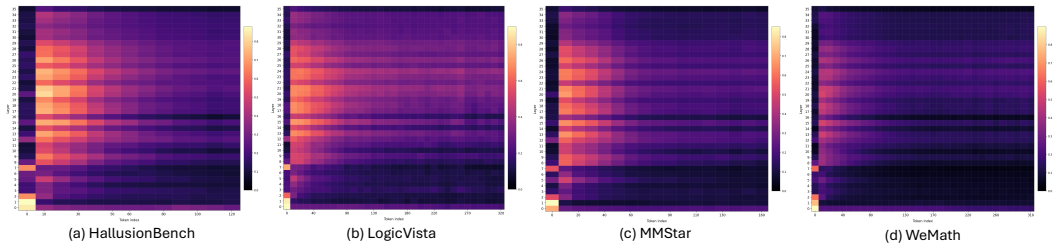


Figure 4: Attention Dynamics across Transformer Depth for Qwen3-VL-4B for (a) Hallusionbench, (b) LogicVista, (c) MMStar, and (d) WeMeth.

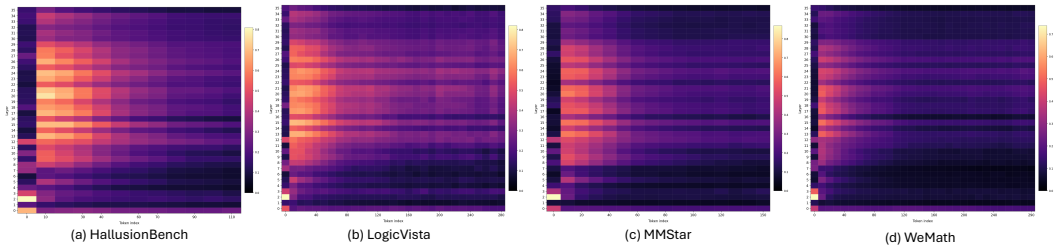


Figure 5: Attention Dynamics across Transformer Depth for Qwen3-VL-8B for (a) Hallusionbench, (b) LogicVista, (c) MMStar, and (d) WeMeth.

3.2 Attention Dynamics across Transformer Depth

Next, we analyze the attention dynamics of the generated reasoning chain g across the L transformer layers. For each generated reasoning token g_y and each layer l, we select the maximum attention from g_y to the visual tokens across all attention heads, and then average across all samples in the

corresponding dataset. The results are shown in Figures 4 and 5 for Qwen3-VL-4B and Qwen3-VL-8B, respectively, while results for the other models are provided in Section C of the Appendix. Both Qwen3-VL-4B and Qwen3-VL-8B consist of 36 transformer layers, shown on the y-axis. The x-axis represents generation time (i.e., the position of each generated reasoning token in the sequence). The heatmap encodes the strength of maximum attention to the visual tokens, with brighter regions indicating stronger visual attention.

Across all benchmarks and models, we observe a consistent pattern. Visual influence is concentrated primarily in the intermediate transformer layers, whereas both the early and late layers exhibit substantially weaker attention to the visual tokens. At the same time, we observe a clear temporal trend across the reasoning trajectory, mirroring the findings in Section 3.1. Visual attention is strongest for the first few generated reasoning tokens and gradually decreases as generation progresses. The last layers, have a relatively weaker visual influence across the entire decoding trajectory. These results suggest that the visual integration window is localized not only in generation time but also in transformer depth, with visual information being integrated primarily in intermediate layers during the early stages of decoding, while the final layers operate predominantly on the resulting textual representations.

Observation 2: Visual attention peaks in intermediate transformer layers during early reasoning, and progressively weakens across both generation time and later layers.

4 Causal Interventions

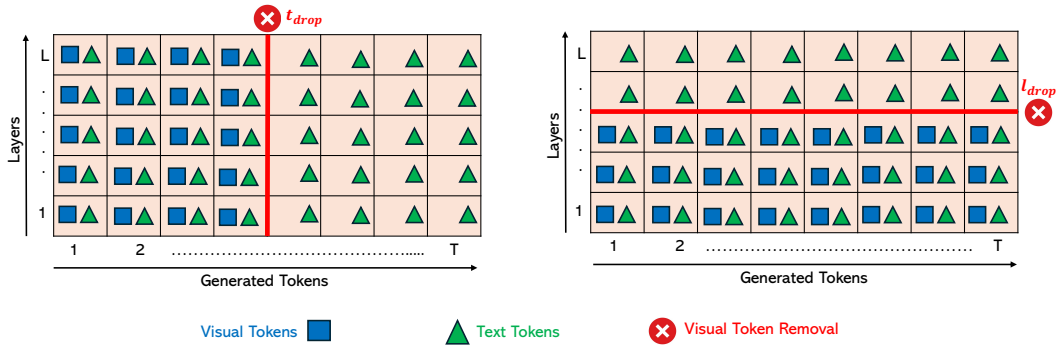


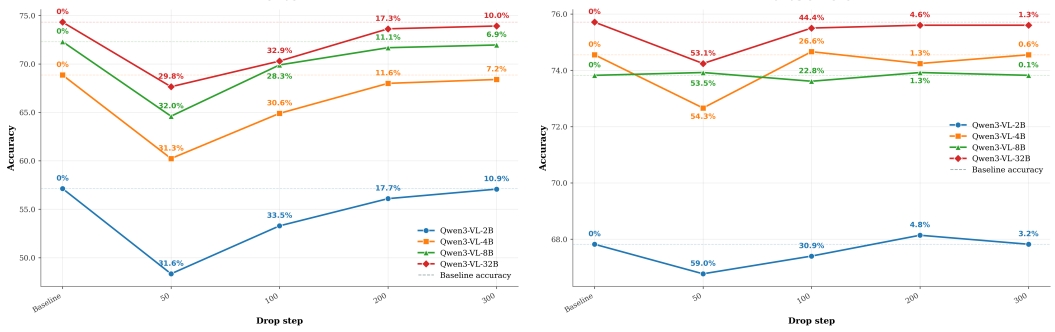
Figure 6: Causal Interventions through visual token removal across generation time (left) and transformer depth (right).

Our findings in Section 3 suggest that access to the original visual tokens becomes progressively less important along two dimensions: as autoregressive generation proceeds and in the later layers of the transformer. We next test these observations through *causal interventions* in which we propose to remove all visual tokens from the model's context at different points in generation time and transformer depth. These interventions directly verify our findings and test empirically whether continued access to the visual tokens is necessary for reasoning. If visual information has already been integrated into the generated textual representations during the visual integration window, removing the original visual tokens afterwards should have only a limited effect on task performance. Similarly, as weak visual influence is observed in the final transformer layers, removing all visual tokens in those layers should lead to a limited effect on task performance. Beyond validating our findings, these interventions expose the practical trade-off between task accuracy and visual token reduction. Visual token removal reduces the visual context that must be maintained and attended to, thereby decreasing both key-value cache memory and attention computation during autoregressive decoding. Our causal intervention experiment is illustrated in Figure 6 for visual tokens removal across generation time (left) and across transformer depth (right). During attention in transformer models, only the keys and values are needed for past tokens, while the query is computed from the current token. Transformer models store the processed keys and values in a key-value (KV) cache. Therefore, dropping visual tokens involves dropping their key-value cache.

Dropping across generation time: At a generation step t_{drop}, we remove the KV cache entries of all N visual tokens in f_v^l from every transformer layer $l \in \{1, \ldots, L\}$. For all subsequent reasoning tokens g_y where $y \geq t_{\text{drop}}$, the model can no longer attend to f_v^l in any of the layers, and must continue generation using the layer-wise KV cache representations of the textual prompt and previously generated tokens.

We evaluate four dropping times: $t_{\text{drop}} \in \{50, 100, 200, 300\}$. Dropping performance across those generation times is reported in Table 1 for all benchmarks and models. For each benchmark-model combination, we report the baseline performance without any dropping. We also report in green the the visual token reduction (%), where samples not satisfying either of the following conditions are assigned a visual token reduction of 0%: (1) the number of generated tokens is less than the drop step, and (2) the number of generated tokens is greater than that of the baseline. It represents the percentage of visual tokens removed from the full input prompt, averaged across all samples, with ineligible samples contributing 0% reduction. We impose these criteria because visual token dropping is considered ineffective when the generation terminates before the drop step. Furthermore, if dropping visual tokens leads to a longer generation than the baseline, then dropping cannot reflect an advantage. We also report the number of generated tokens in Section D of the Appendix. These two conditions ensure a fair comparison with the baseline by assigning zero reduction whenever dropping is not actually applied or does not reduce the effective generation cost. We also summarize the results of Table 1 as a graph in Figure 7, where we show how performance varies as t_{drop} increases. For each point, we also provide the visual token reduction %.

The results show that dropping visual tokens early, while the visual integration window is still active, degrades performance because visual information has not yet been fully integrated into the textual representations. As t_{drop} increases (i.e., as visual tokens are removed later during generation), performance gradually recovers. Around timestep 200, the performance degradation becomes minimal, suggesting that the visual integration window has ended. We therefore identify this point as the *general best trade-off* point, providing a favorable balance between preserving performance and reducing visual tokens across most benchmarks. At this point, the average visual tokens reduction ratio is around 22%. For some datasets, however, the visual integration window appears to be shorter, with performance recovering as early as timestep 100. At this point, the average visual tokens reduction ratio is around 30%. Overall, these causal intervention results are consistent with the attention dynamics observed in Section 3.1, which confirms our findings. FLOPs and memory savings are reported in Appendix Section B.1.



(a) WeMath (b) HallusionBench

Figure 7: Visual token dropping across generation time for (a) Wemath, (b) HallusionBench. Different models are shown in different colors. The x-axis denotes different dropping times t_{drop} while the y-axis shows the accuracy. The visual token reduction % is shown above each point.

Dropping across transformer depth: At layer l_{drop}, we remove the KV cache entries of all N visual tokens in f_v^l from layers $l \geq l_{\text{drop}}$. We perform this during the prefill stage (before generation starts), and the dropping is maintained across the entire generation trajectory $y \in \{1, \ldots, Y\}$. Thus, layers $l < l_{\text{drop}}$ operate normally and can still process visual information, whereas the upper layers $l \geq l_{\text{drop}}$ operate without access to any visual tokens.

We evaluate three different layer dropping indices: $l_{\text{drop}} \in \{L-5, L-10, L-15\}$, where L is the total number of layers. For example, $L-5$ indicates that we apply dropping 5 layers before the final

Table 1: Visual token dropping at different generation steps across four benchmarks and four models. We also indicate the degradation from the baseline, where darker red indicates higher degradation. We also indicate the visual token reduction % next to each score.

Benchmark	Drop	Qwen3-VL-2B	Qwen3-VL-4B	Qwen3-VL-8B	Qwen3-VL-32B
WeMath	Baseline	57.13 00.0%	68.85 00.0%	72.30 00.0%	74.31 00.0%
	50	48.33 ↓31.6%	60.23 ↓31.3%	64.60 ↓32.0%	67.64 ↓29.8%
	100	53.28 ↓33.5%	64.89 ↓30.6%	69.89 ↓28.3%	70.29 ↓32.9%
	200	56.09 ↓17.7%	67.99 ↓11.6%	71.67 ↓11.1%	73.62 ↓17.3%
	300	57.07 ↓10.9%	68.39 ↓7.2%	71.95 ↓6.9%	73.91 ↓10.0%
LogicVista	Baseline	33.71 00.0%	39.73 00.0%	50.89 00.0%	51.56 00.0%
	50	24.11 ↓44.4%	26.34 ↓41.2%	34.38 ↓39.7%	27.90 ↓41.9%
	100	30.13 ↓48.0%	29.69 ↓43.4%	42.86 ↓41.1%	35.49 ↓40.3%
	200	32.14 ↓38.8%	38.62 ↓23.2%	49.11 ↓18.2%	45.98 ↓37.5%
	300	33.48 ↓26.5%	39.06 ↓12.8%	50.67 ↓9.6%	47.54 ↓28.2%
MMStar	Baseline	56.13 00.0%	66.67 00.0%	68.00 00.0%	62.33 00.0%
	50	52.47 ↓38.7%	62.47 ↓38.6%	64.20 ↓39.0%	56.87 ↓35.9%
	100	55.20 ↓31.7%	65.20 ↓24.3%	66.33 ↓22.4%	60.80 ↓35.2%
	200	56.27 ↓9.8%	65.93 ↓6.3%	68.00 ↓5.5%	61.13 ↓8.3%
	300	56.13 ↓8.0%	66.47 ↓4.7%	67.60 ↓3.9%	61.80 ↓4.0%
HallusionBench	Baseline	67.82 00.0%	74.55 00.0%	73.82 00.0%	75.71 00.0%
	50	66.77 ↓59.0%	72.66 ↓54.3%	73.92 ↓53.5%	74.24 ↓53.1%
	100	67.40 ↓30.9%	74.66 ↓26.6%	73.61 ↓22.8%	75.50 ↓44.4%
	200	68.14 ↓4.8%	74.24 ↓1.3%	73.92 ↓1.3%	75.60 ↓4.6%
	300	67.82 ↓3.2%	74.55 ↓0.6%	73.82 ↓0.1%	75.60 ↓1.3%

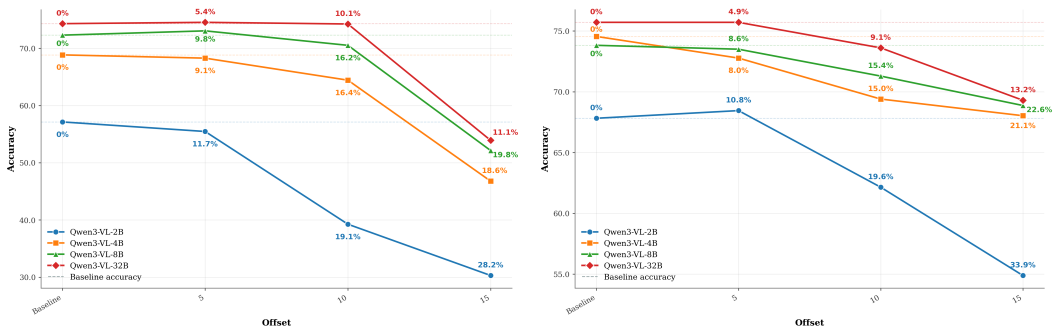
layer, where 5 is the offset. Dropping performance across those layer indices is reported in Table 2 for all benchmarks and models, along with the baseline performance without any dropping. We also report in green the visual token reduction (%), where samples for which the number of generated tokens exceeds that of the baseline are assigned a visual token reduction of 0%. For the same reason discussed in the previous experiment, this ensures a fair comparison with the baseline by assigning zero reduction whenever dropping leads to a longer generation than the baseline. The visual token reduction (%) represents the percentage of visual tokens dropped across all layers of the transformer model, averaged across all samples, with ineligible samples contributing 0% reduction. The number of generated tokens are reported in Section D of the Appendix. We also summarize the results of Table 2 as a graph in Figure 8, where we show how performance varies as the offset increases. For each point, we also provide the visual token reduction %.

Results in Table 2 and Figure 8 show that all visual tokens can be removed from the upper transformer layers without substantially affecting reasoning performance. Dropping visual tokens from the last five layers (Offset 5) produces negligible degradation across most benchmarks and model sizes, while achieving up to 12% visual token reduction, indicating that the final layers contribute relatively little direct visual processing and mainly operate on representations processed by lower layers. However, extending the removal to earlier layers causes a consistent performance decline, suggesting that intermediate layers are important for multimodal reasoning. Our results also suggest that larger models are more robust than smaller models under both intermediate and late layer dropping. The degradation caused by removing visual tokens from intermediate and late layers is substantially smaller than in compact variants, suggesting that increased capacity provides stronger redundancy in visual representation processing and that larger multimodal models encode visual information more efficiently in intermediate layers. Overall, the causal intervention results agree with the layer-wise attention analysis in Section 3.2, and we can remove visual tokens near the top of the transformer stack. FLOPs and memory savings are reported in Appendix Section B.2.

Dropping simultaneously across time and depth: We also explore a combined setting that applies token dropping along both transformer depth and generation time. Visual tokens are first dropped at layer l_{drop}, and as generation progresses, they are also dropped at timestep t_{drop}. As this configuration is an additional analysis, we refer readers to Section A of the Appendix.

Table 2: Visual token dropping at different layers across four benchmarks and four models. We drop at layer L - offset, where L is the final layer index. We also indicate the degradation from the baseline, where darker red indicates higher degradation. We also indicate the visual token reduction % next to each score.

Benchmark	Drop	Qwen3-VL-2B	Qwen3-VL-4B	Qwen3-VL-8B	Qwen3-VL-32B
WeMath	Baseline	57.13 00.0%	68.85 00.0%	72.30 00.0%	74.31 00.0%
	Offset 5	55.46 ↓11.71%	68.28 ↓9.10%	73.05 ↓9.77%	74.54 ↓5.36%
	Offset 10	39.25 ↓19.11%	64.43 ↓16.43%	70.52 ↓16.22%	74.25 ↓10.06%
	Offset 15	30.29 ↓28.23%	46.78 ↓18.58%	52.13 ↓19.80%	53.91 ↓11.05%
LogicVista	Baseline	33.71 00.0%	39.73 00.0%	50.89 00.0%	51.56 00.0%
	Offset 5	31.25 ↓11.52%	40.40 ↓8.56%	47.54 ↓8.46%	51.12 ↓4.67%
	Offset 10	21.65 ↓19.05%	32.59 ↓15.56%	39.96 ↓15.07%	49.11 ↓9.07%
	Offset 15	25.45 ↓39.58%	20.98 ↓20.28%	26.12 ↓19.62%	25.67 ↓9.94%
MMStar	Baseline	56.13 00.0%	66.67 00.0%	68.00 00.0%	62.33 00.0%
	Offset 5	55.87 ↓10.82%	65.80 ↓8.29%	67.47 ↓8.96%	59.07 ↓5.04%
	Offset 10	45.93 ↓19.67%	62.53 ↓15.72%	64.47 ↓16.44%	60.13 ↓9.28%
	Offset 15	41.67 ↓31.75%	55.53 ↓22.58%	57.20 ↓23.81%	50.33 ↓12.70%
HallusionBench	Baseline	67.82 00.0%	74.55 00.0%	73.82 00.0%	75.71 00.0%
	Offset 5	68.45 ↓10.76%	72.77 ↓7.97%	73.50 ↓8.56%	75.71 ↓4.87%
	Offset 10	62.15 ↓19.60%	69.40 ↓15.01%	71.29 ↓15.36%	73.61 ↓9.05%
	Offset 15	54.89 ↓33.86%	68.03 ↓21.12%	68.87 ↓22.65%	69.30 ↓13.16%



(a) WeMath (b) HallusionBench

Figure 8: Visual token dropping across transformer depth for (a) WeMath, (b) HallusionBench. The x-axis denotes different layer offsets, where visual tokens are removed starting from the last $L-\text{offset}$ layers. The y-axis shows the accuracy. The visual token reduction % is shown next to each point.

5 Conclusion

We investigated how visual information is utilized during long-form LVLM reasoning. Through attention analysis and causal interventions, we show that visual influence is concentrated within an early generation window and intermediate transformer layers, while later reasoning increasingly relies on representations accumulated in the textual context. These findings reveal that continuous access to all visual tokens is not always necessary and provide a basis for reducing visual-token-related memory and computation during multimodal inference.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. 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. *arXiv preprint arXiv:2502.13923*, 2025b.

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.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, and Feng Zhao. Are we on the right way for evaluating large vision-language models? In *Advances in Neural Information Processing Systems*, 2024b.

Ailin Deng, Tri Cao, Zhirui Chen, and Bryan Hooi. Words or vision: Do vision-language models have blind faith in text? *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3867–3876, 2025. URL https://api.semanticscholar.org/CorpusID:276775193.

Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, Dinesh Manocha, and Tianyi Zhou. Hallusionbench: An advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14375–14385, 2024.

Yu Huang, Junyang Lin, Chang Zhou, Hongxia Yang, and Longbo Huang. Modality competition: What makes joint training of multi-modal network fail in deep learning? (Provably). In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9226–9259. PMLR, 17–23 Jul 2022.

Zhiqiu Lin, Xinyue Chen, Deepak Pathak, Pengchuan Zhang, and Deva Ramanan. Revisiting the role of language priors in vision-language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Letitia Parcalabescu and Anette Frank. Do vision & language decoders use images and text equally? how self-consistent are their explanations? In *The Thirteenth International Conference on Learning Representations*, 2025.

Simon Park, Abhishek Panigrahi, Yun Cheng, Dingli Yu, Anirudh Goyal, and Sanjeev Arora. Generalizing from SIMPLE to HARD visual reasoning: Can we mitigate modality imbalance in VLMs? In *Forty-second International Conference on Machine Learning*, 2025.

Mohammad Pezeshki, Sékou-Oumar Kaba, Yoshua Bengio, Aaron Courville, Doina Precup, and Guillaume Lajoie. Gradient starvation: A learning proclivity in neural networks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021.

Runqi Qiao, Qiuna Tan, Guanting Dong, Minhui Wu, Chong Sun, Xiaoshuai Song, Zhuoma GongQue, Shanglin Lei, Zhe Wei, Miaoxuan Zhang, et al. We-math: Does your large multimodal model achieve human-like mathematical reasoning? *arXiv preprint arXiv:2407.01284*, 2024.

Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes wide shut? exploring the visual shortcomings of multimodal llms. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9568–9578, 2024.

Hao Wu, Yingqi Fan, Dai Jinyang, Junlong Tong, Yunpu Ma, and Xiaoyu Shen. Hidrop: Hierarchical vision token reduction in MLLMs via late injection, concave pyramid pruning, and early exit. In *The Fourteenth International Conference on Learning Representations*, 2026.

Yijia Xiao, Edward Sun, Tianyu Liu, and Wei Wang. Logicvista: Multimodal llm logical reasoning benchmark in visual contexts, 2024.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis A Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. SparseVLM: Visual token sparsification for efficient vision-language model inference. In *Forty-second International Conference on Machine Learning*, 2025.

APPENDIX

A COMBINED DROPPING ACROSS TIME AND DEPTH

We explore dropping across both generation time and transformer depth simultaneously. In this setting, visual tokens are first dropped at layer l_{drop}, and as generation progresses, they are also dropped at timestep t_{drop}. We consider the combined criteria as detailed in Section 4 of the main manuscript. That is, the visual token reduction % is considered only for samples where 1) the number of generated tokens is lower than or equal to that of the baseline, 2) the number of generated tokens is bigger than that of the drop step t_{drop}. This is to establish a fair comparison and ensure that the dropping is effective.

We calculate how much visual-token computation is removed over the entire multimodal transformer inference process. Since visual tokens are not dropped everywhere at once (they are dropped only after a certain layer and after a certain generation step), the reduction is a weighted combination. We define the overall visual token reduction ratio by considering reductions across three dimensions: (i) the fraction of visual tokens in the input prompt, (ii) the fraction of transformer layers after the visual-token dropping layer, and (iii) the fraction of generation steps after the KV-cache dropping timestep.

Let N_{prompt} denote the total number of input prompt tokens, including both text and visual tokens:

$$N_{\text{prompt}} = N_{\text{text}} + N_{\text{image}},$$

where N_{text} and N_{image} are the numbers of text and visual tokens, respectively. The fraction of visual tokens in the prompt is therefore

$$r_{\text{image}} = \frac{N_{\text{image}}}{N_{\text{prompt}}}.$$

Given a transformer with L layers, visual tokens are removed at layer l_{drop}. We define the number of layers before and after visual token removal as

$$L_{\text{early}} = l_{\text{drop}}, \qquad L_{\text{late}} = L - l_{\text{drop}}.$$

The corresponding layer fractions are

$$r_{\text{early}} = \frac{L_{\text{early}}}{L}, \qquad r_{\text{late}} = \frac{L_{\text{late}}}{L}.$$

During autoregressive generation, visual tokens are removed from the KV cache after timestep t_{drop}. Let N_{gen} denote the total number of generated tokens and $N_{\text{gen,after KV drop}}$ denote the number of generated tokens after the KV-cache dropping timestep. The remaining generation fraction is

$$r_{\text{gen}} = \frac{N_{\text{gen,after KV drop}}}{N_{\text{gen}}}.$$

Combining the above factors, the overall visual token reduction percentage is computed as

$$\text{Visual Token Reduction}(\%) = r_{\text{image}}\,(r_{\text{late}} + r_{\text{early}}\, r_{\text{gen}}) \times 100$$

The reported reduction values are computed only for samples satisfying the following conditions:

$$N_{\text{gen}}^{\text{drop}} \leq N_{\text{gen}}^{\text{baseline}},$$

and

$$N_{\text{gen}}^{\text{drop}} > t_{\text{drop}},$$

ensuring a fair comparison and that the visual token dropping operation is effectively applied. The final values are averaged over all valid samples. The results are shown in Table 1.

Table 1: Combined visual token dropping across both generation time and transformer depth. We indicate the visual token reduction % next to each score.

Benchmark	Offset	Drop after	Qwen3-VL-2B	Qwen3-VL-4B	Qwen3-VL-8B	Qwen3-VL-32B
WeMath	Baseline		56.72 00.0%	69.25 00.0%	73.16 00.0%	74.37 00.0%
	5	200	55.34 ↓11.17%	67.24 ↓8.39%	72.01 ↓7.84%	72.93 ↓8.19%
	5	300	55.17 ↓9.97%	68.16 ↓7.22%	72.13 ↓7.01%	73.28 ↓6.11%
	10	200	40.29 ↓13.26%	63.51 ↓10.93%	69.48 ↓10.60%	71.84 ↓9.42%
	10	300	40.23 ↓12.55%	63.97 ↓10.21%	69.94 ↓9.85%	73.56 ↓7.88%
LogicVista	Baseline		34.82 00.0%	39.06 00.0%	50.45 00.0%	50.45 00.0%
	5	200	32.59 ↓20.96%	35.94 ↓10.31%	46.43 ↓10.26%	43.75 ↓18.17%
	5	300	33.71 ↓17.66%	39.06 ↓9.32%	47.99 ↓8.84%	45.76 ↓14.48%
	10	200	19.20 ↓22.85%	31.47 ↓14.20%	40.18 ↓12.52%	43.97 ↓20.01%
	10	300	18.97 ↓21.03%	33.71 ↓13.57%	40.62 ↓11.83%	45.98 ↓15.59%
MMStar	Baseline		57.27 00.0%	66.80 00.0%	68.40 00.0%	61.67 00.0%
	5	200	54.47 ↓9.27%	64.47 ↓7.27%	66.27 ↓7.22%	59.33 ↓5.23%
	5	300	54.60 ↓8.67%	64.93 ↓7.01%	66.60 ↓6.88%	59.47 ↓4.50%
	10	200	46.00 ↓12.92%	62.13 ↓10.70%	63.87 ↓10.68%	59.40 ↓6.98%
	10	300	46.00 ↓12.70%	62.13 ↓10.48%	64.20 ↓10.45%	60.00 ↓6.36%
HallusionBench	Baseline		69.09 00.0%	74.97 00.0%	74.03 00.0%	76.87 00.0%
	5	200	69.19 ↓9.77%	72.87 ↓6.70%	74.45 ↓6.69%	76.03 ↓4.32%
	5	300	69.40 ↓9.62%	72.98 ↓6.59%	74.55 ↓6.67%	76.03 ↓3.97%
	10	200	60.99 ↓17.30%	69.93 ↓12.29%	71.19 ↓12.62%	73.82 ↓7.46%
	10	300	60.88 ↓17.18%	70.14 ↓12.19%	71.29 ↓12.60%	74.13 ↓7.36%

B FLOPs and Memory

B.1 Dropping Across Generation Time

We quantify the computational and memory savings induced by image token dropping across generation time. Since KV-cache reduction only affects the attention operation, we focus these experiments on the attention QKV multiplications.

KV-cache computation savings: Let P denote the prompt length before dropping, N the number of visual KV tokens removed, T the generation length of the KV-drop run, and t_{drop} the generation step at which the visual KV entries are removed. We further denote by L the number of transformer layers, by H_q the number of query attention heads, by H_{kv} the number of key-value heads, by d_h the attention head dimension.

At decoding step y, before visual-token dropping, the attention sequence length is

$$S_y = P + y. \tag{1}$$

For a single transformer layer and a single generated token, we have only one query token that attends to S_y KV cached tokens. Computing the query-key product $QK^\top$ requires

$$2H_q d_h S_y \tag{2}$$

floating-point operations (FLOPs), where one multiply-add is counted as two FLOPs. Multiplying the resulting attention weights by the value matrix requires an additional

$$2H_q d_h S_y \tag{3}$$

FLOPs. Therefore, across all L transformer layers, the attention computation at generation step y is

$$F_{\text{attn}}(y) = 4LH_q d_h S_y. \tag{4}$$

After dropping N visual KV tokens, the attention sequence length for every subsequent generation step is reduced by N. The attention FLOPs saved per post-drop generation step are

$$\Delta F_{\text{step}} = 4LH_q d_h N. \tag{5}$$

Since dropping at t_{drop} affects all subsequent generated tokens, the number of generation steps benefiting from the reduced KV cache is $T - t_{\text{drop}}$. The total attention FLOPs saved over the generation trajectory are therefore

$$\Delta F = 4LH_q d_h N \,(T - t_{\text{drop}}) \tag{6}$$

for eligible samples. We use the same eligibility criteria as for the visual-token reduction metric. Specifically, a sample contributes to the FLOP and memory reductions only when

$$T > t_{\text{drop}} \qquad \text{and} \qquad T \leq T_{\text{baseline}}, \tag{7}$$

where T_{baseline} is the number of tokens generated by the corresponding baseline run. The first condition ensures that the generation reaches the dropping point, while the second prevents assigning computational savings when dropping results in a longer generation than the baseline. Samples that do not satisfy either condition are assigned zero reduction.

We also report another metric which provides a more direct comparison of FLOPs saving with respect to the baseline. We compute $\Delta F_{actual} = F_{baseline} - F_{drop}$. That is, we take the difference between the total FLOPs of the baseline and the dropping. The baseline uses its actual generation length T_b, while the intervention uses its actual generation length T_d. This counts the computation avoided since all steps after t_{drop} never happened.

KV-cache memory reduction: Denote b as the number of bytes used to store each KV element (e.g., $b = 2$ for FP16/BF16). In multi-head attention, the KV cache stores H_q keys and H_q values for each token and transformer layer. For grouped-query attention used in Qwen3-VL, only H_{kv} keys and H_{kv} heads are stored per token, for each layer. Hence, the memory required by one cached token across all transformer layers is

$$M_{\text{token}} = 2LH_{kv}d_h b \quad \text{bytes}, \tag{8}$$

where the factor of 2 accounts for storing both the key and value representations. Dropping N visual KV tokens therefore reduces the KV-cache memory by

$$\Delta M = 2LH_{kv}d_h bN \quad \text{bytes}. \tag{9}$$

This quantity represents the instantaneous KV-cache memory reduction after the dropping operation.

To also quantify the cumulative amount of KV data removed from subsequent attention operations, we compute the logical KV-read reduction as

$$\Delta M_{cum} = \Delta M \,(T - t_{\text{drop}}) \tag{10}$$

bytes. This quantity represents the total amount of reduction in the KV data involved in attention over the complete generation trajectory. Without visual token dropping, the visual KV representations would continue to participate in every attention operation after generation step t_{drop}. Unlike ΔM, this quantity represents how much future reserved memory is removed from consideration (across all subsequent attention operations).

Results are shown in Table 2. ΔF and ΔF_{actual} are reported in GFLOPs, while ΔM in MB and ΔM_{cum} in GB.

B.2 Dropping Across Transformer Depth

Computation savings: Let P denote the original prompt length, including V visual tokens, L the total number of transformer layers, and K be the number of upper layers where visual tokens are removed. Let T_b and T_d denote the number of tokens generated by the baseline and dropping intervention, respectively. For the transformer architecture, let D denote the hidden dimension, I the MLP intermediate dimension, H_q the number of query heads, H_{kv} the number of key-value heads, and d_h the head dimension. The query dimensions are $Q = H_q d_h$, and the key-value dimensions are $Q_{kv} = H_{kv} d_h$. For a single token at a single transformer layer, we have the cost of the query projection ($2DQ$), key projection ($2DQ_{kv}$), value projection ($2DQ_{kv}$), and attention output projections ($2DQ$). Qwen uses a SwiGLU-style MLP containing gate, up, and down projections: $2DI + 2DI + 2DI$. Therefore, for all projections, we have $C = 4DQ + 4DQ_{kv} + 6DI$.

For a sequence of n tokens during the prefil, the matrx shapes of each of the queries, keys and values are $(H_q, n, d_h), (H_{kv}, n, d_h), (H_{kv}, n, d_h)$, respectively. For query-key matrix multiplications,

Table 2: Attention compute and memory savings across Qwen3-VL models. ΔF and ΔF_{actual} are reported in GFLOPs. ΔM is reported in MB, while ΔM_{cum} is reported in GB. Higher values indicate larger savings.

		Qwen3-VL-2B				Qwen3-VL-4B			
Benchmark	**Drop**	$\Delta F \uparrow$	$\Delta F_{\text{actual}} \uparrow$	$\Delta M \uparrow$	$\Delta M_{\text{cum}} \uparrow$	$\Delta F \uparrow$	$\Delta F_{\text{actual}} \uparrow$	$\Delta M \uparrow$	$\Delta M_{\text{cum}} \uparrow$
WeMath	50	11.76	63.31	17.20	5.88	20.09	88.33	22.34	5.02
	100	13.95	50.72	17.84	6.98	19.45	73.63	21.12	4.86
	200	15.15	29.00	9.77	7.58	19.60	57.37	7.53	4.90
	300	15.09	18.38	5.90	7.54	17.02	45.13	4.76	4.25
LogicVista	50	36.38	127.39	28.40	18.19	45.51	203.12	34.67	11.38
	100	37.96	105.52	31.32	18.98	57.11	158.11	36.56	14.28
	200	47.50	68.85	26.59	23.75	57.26	118.53	20.29	14.31
	300	45.49	54.70	18.42	22.74	61.66	87.20	11.52	15.42
MMStar	50	9.35	52.77	18.73	4.67	21.91	62.97	24.20	5.48
	100	12.25	37.91	16.35	6.12	21.24	53.39	16.46	5.31
	200	15.69	19.49	6.00	7.85	21.18	32.38	4.92	5.29
	300	14.83	16.29	5.06	7.42	20.15	32.20	3.82	5.04
HallusionBench	50	17.52	43.30	77.44	8.76	16.65	23.12	85.13	4.16
	100	23.33	28.77	44.89	11.67	6.79	10.94	46.23	1.70
	200	20.65	22.69	7.13	10.32	1.99	5.39	1.43	0.50
	300	20.34	20.34	5.82	10.17	2.14	3.55	0.58	0.54

		Qwen3-VL-8B				Qwen3-VL-32B			
Benchmark	**Drop**	$\Delta F \uparrow$	$\Delta F_{\text{actual}} \uparrow$	$\Delta M \uparrow$	$\Delta M_{\text{cum}} \uparrow$	$\Delta F \uparrow$	$\Delta F_{\text{actual}} \uparrow$	$\Delta M \uparrow$	$\Delta M_{\text{cum}} \uparrow$
WeMath	50	18.86	91.72	22.50	4.71	79.33	323.42	36.16	9.92
	100	19.77	72.77	19.02	4.94	79.24	253.04	41.47	9.90
	200	19.22	51.28	7.57	4.81	66.80	190.57	21.25	8.35
	300	18.73	40.23	4.65	4.68	58.50	174.81	12.35	7.31
LogicVista	50	30.60	135.10	32.59	7.65	289.86	1073.66	63.91	36.23
	100	29.65	124.41	34.70	7.41	280.00	914.72	60.53	35.00
	200	31.31	89.37	15.41	7.83	291.34	828.12	58.87	36.42
	300	34.14	67.10	8.37	8.53	285.57	700.57	46.45	35.70
MMStar	50	14.94	55.93	23.77	3.73	54.72	176.05	39.66	6.84
	100	15.45	40.02	16.14	3.86	51.15	121.64	38.30	6.39
	200	13.44	37.20	4.10	3.36	39.38	89.39	12.04	4.92
	300	15.00	27.02	3.23	3.75	41.54	73.55	5.82	5.19
HallusionBench	50	12.43	15.72	82.23	3.11	84.28	110.70	165.52	10.54
	100	3.47	4.18	33.03	0.87	33.06	59.94	101.06	4.13
	200	0.21	0.26	1.12	0.05	14.20	21.49	8.79	1.77
	300	0.03	0.03	0.08	0.01	10.93	16.77	2.72	1.37

since the multiplications is done independently across batched heads, the computation involved is $2d_h H_q n^2$. Taking into account also the key-value multiplication adds another $2d_h H_q n^2$, resulting in $4d_h H_q n^2$. Since prefil uses autoregressive causal attention, the computation involved reduces to $4d_h H_q[n(n+1)/2] = 2d_h H_q n(n+1)$. The complete prefill cost of one layer for n tokens is therefore:

$$\Phi(n) = nC + A(n), \tag{11}$$

where nC accounts for the token-wise computation, including the Q/K/V projections, attention output projection, and MLP computation, while $A(n)$ denotes the self-attention computation. Before visual-token dropping, an upper transformer layer processes all P prompt tokens and therefore

requires $\Phi(P)$ FLOPs. After dropping the V visual tokens, the same layer processes only $P - V$ tokens, resulting in a computational cost of $\Phi(P - V)$. In order to calculate ΔF, which are the FLOPs saved because these V visual tokens disappeared from the final K layers, we cannot simply perform $K\phi(V)$ because the prefil also includes text prompt tokens. Therefore, we perform:

$$\Delta F_{\text{prefill}} = K\left[\Phi(P) - \Phi(P - V)\right]. \tag{12}$$

During decoding, if the intervention generates T_d tokens, the first generated token g_1 is produced directly from the prefill pass. Therefore there are $T_d - 1$ tokens to consider during autoregressive decoding. At this stage, there is only causal attention. Across $T_d - 1$ decode forwards and K affected layers, the computational savings during decoding for a single query token is $4KH_q d_h V$, and for $T_d - 1$ tokens, we have:

$$\Delta F_{\text{decode}} = 4KH_q d_h V\,(T_d - 1). \tag{13}$$

Therefore, the final FLOPs savings is computed as: $\Delta F = \Delta F_{\text{prefill}} + \Delta F_{\text{decode}}$. Similarly as in B.1, we compute:

$$\Delta F_{\text{actual}} = F_{\text{baseline}}(T_b) - F_{\text{drop}}(T_d). \tag{14}$$

Therefore, ΔF measures how much computation did token dropping itself eliminate, while ΔF_{actual} measures how much less computation was actually performed compared with baseline.

Memory reduction: One visual token has $H_{kv}d_h$ dimensions for the keys and $H_{kv}d_h$ for the values. If each dimension uses b bytes, then the total memory occupied by keys and values for a single token is $2H_{kv}d_h b$. Therefore, removing V visual tokens entries from each of K layers results in memory savings of:

$$\Delta M = 2KVH_{\text{kv}}d_h b. \tag{15}$$

This memory saving is created during prefill and is maintained throughout the entire generation process, and ΔM does not increase as more generated tokens are produced.

For T_d generated tokens, the prefill stage produces the first token g_1, while the subsequent decoding steps produce the remaining tokens. Thus, there are $1 + (T_d - 1) = T_d$ states over which the removed KV-cache entries would otherwise occupy memory. Therefore, the cumulative memory saving is:

$$\Delta M_{\text{cum}} = T_d \Delta M = 2T_d KVH_{\text{kv}}d_h b. \tag{16}$$

As in Section B.1, this does not measure the amount of memory freed, but rather the cumulative memory occupancy avoided over the generation trajectory, which also implies the freed memory reserve. Results are presented in Table 3. ΔF and ΔF_{actual} are reported in GFLOPs. ΔM is reported in in MB, while ΔM_{cum} is reported in GB. Higher values indicate larger savings.

C Layer-Wise Heatmaps for Other Models

We provide the attention Dynamics across Transformer Depth for Qwen3-VL-2B in Figure 1 and for Qwen3-VL-32B in Figure 2.

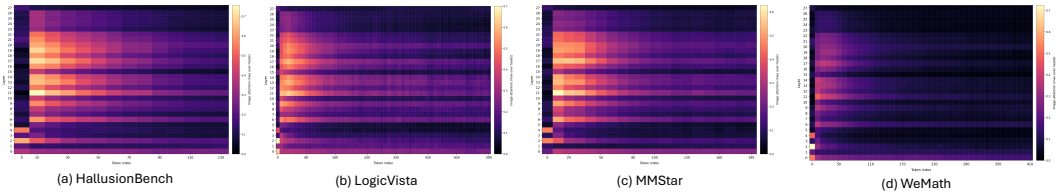
D Number of Generated Tokens

We report the number of generated tokens for all our experiments. Table 4 shows the average number of generated tokens across all samples, when visual token KV caches are dropped at different generation steps across four benchmarks and four models. In Table 5, we show the average number of generated tokens across all samples, when visual tokens are dropped at different layers across four benchmarks and four models. We drop at layer L - offset, where L is the final layer index. Finally,

Table 3: Attention compute and memory savings across Qwen3-VL models for dropping across transformer depth. ΔF and ΔF_{actual} are reported in GFLOPs. ΔM is reported in MB, while ΔM_{cum} is reported in GB. Higher values indicate larger savings.

		Qwen3-VL-2B				Qwen3-VL-4B			
Benchmark	**Offset**	$\Delta F \uparrow$	$\Delta F_{\text{actual}} \uparrow$	$\Delta M \uparrow$	$\Delta M_{\text{cum}} \uparrow$	$\Delta F \uparrow$	$\Delta F_{\text{actual}} \uparrow$	$\Delta M \uparrow$	$\Delta M_{\text{cum}} \uparrow$
WeMath	Offset 5	126.47	748.68	3.14	2.20	239.47	1188.67	3.00	1.36
	Offset 10	254.52	1181.48	5.18	4.21	489.69	1643.25	5.54	2.93
	Offset 15	395.53	1546.43	7.95	5.68	754.52	2143.59	6.42	5.44
LogicVista	Offset 5	207.53	1120.74	4.94	6.18	414.51	2736.99	4.85	2.97
	Offset 10	400.46	1542.04	7.84	13.18	804.64	3247.40	8.53	6.64
	Offset 15	634.81	2160.12	17.73	11.08	1182.39	3991.35	10.85	11.09
MMStar	Offset 5	135.76	716.48	3.08	2.06	270.02	897.59	3.02	1.60
	Offset 10	256.35	932.89	5.33	4.25	563.30	1324.43	5.98	3.36
	Offset 15	392.07	1301.43	8.89	4.01	800.10	1650.08	8.16	4.74
HallusionBench	Offset 5	452.22	741.63	9.83	2.94	883.75	1033.85	9.16	1.51
	Offset 10	991.83	1333.34	19.26	5.82	2125.57	2291.72	19.98	3.23
	Offset 15	1658.95	2095.24	35.98	5.30	3503.47	3772.87	30.80	5.11

		Qwen3-VL-8B				Qwen3-VL-32B			
Benchmark	**Offset**	$\Delta F \uparrow$	$\Delta F_{\text{actual}} \uparrow$	$\Delta M \uparrow$	$\Delta M_{\text{cum}} \uparrow$	$\Delta F \uparrow$	$\Delta F_{\text{actual}} \uparrow$	$\Delta M \uparrow$	$\Delta M_{\text{cum}} \uparrow$
WeMath	Offset 5	461.82	1943.32	3.35	1.33	1129.00	7294.21	3.17	1.56
	Offset 10	930.50	2980.13	5.59	2.86	2393.94	10070.32	6.31	3.35
	Offset 15	1397.63	3912.32	6.83	4.77	3654.42	13585.79	7.04	5.08
LogicVista	Offset 5	773.36	3329.26	4.85	2.24	1903.71	13570.48	4.58	6.61
	Offset 10	1456.84	4307.79	8.12	4.75	4024.35	18437.39	9.41	12.60
	Offset 15	2269.59	6129.48	10.98	7.56	6259.86	25812.80	10.63	23.92
MMStar	Offset 5	470.83	1391.14	3.12	1.26	1247.19	3366.47	3.27	1.32
	Offset 10	943.64	2438.58	5.73	2.32	2304.09	5523.84	5.58	2.81
	Offset 15	1441.21	3041.34	8.44	3.34	3456.41	6828.49	7.66	4.12
HallusionBench	Offset 5	1548.27	1691.72	9.62	1.38	4040.25	5119.77	9.99	1.92
	Offset 10	3028.45	3264.93	17.05	2.53	8359.61	9779.03	19.19	3.95
	Offset 15	5100.65	5438.55	27.91	4.05	13075.21	15007.77	29.26	5.75



(a) HallusionBench (b) LogicVista (c) MMStar (d) WeMath

Figure 1: Attention Dynamics across Transformer Depth for Qwen3-VL-4B for (a) Hallusionbench, (b) LogicVista, (c) MMStar, and (d) WeMeth.

we also report the number of generated tokens in Table 6 for the combined dropping setting across both generation time and depth.

E Plotting Details

We describe the procedure used to generate the attention plots in Section 3.1 of the main manuscript. Since the generated reasoning sequences may differ in length across samples, we first synchronize them according to their token positions using padding. Statistical quantities are then computed only over non-padded entries, ensuring that each estimate reflects contributions from valid tokens only.

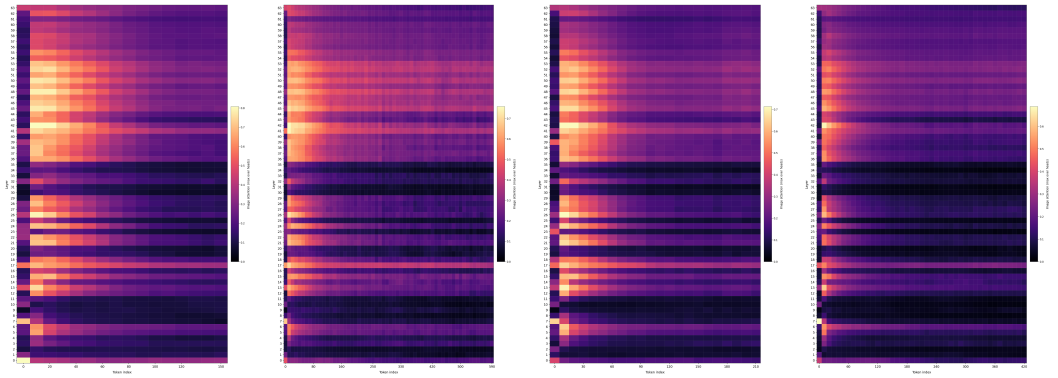


Figure 2: Attention Dynamics across Transformer Depth for Qwen3-VL-32B for (a) Hallusionbench, (b) LogicVista, (c) MMStar, and (d) WeMeth.

Table 4: Average number of generated tokens across all samples, when visual token are dropped at different generation steps across four benchmarks and four models.

Benchmark	Drop	Qwen3-VL-2B	Qwen3-VL-4B	Qwen3-VL-8B	Qwen3-VL-32B
WeMath	Baseline	499	370	345	409
	50	532	452	403	449
	100	525	406	364	436
	200	499	376	357	415
	300	500	368	350	404
LogicVista	Baseline	720	475	354	667
	50	811	574	494	700
	100	786	588	423	730
	200	732	478	373	652
	300	729	471	371	656
MMStar	Baseline	363	261	242	253
	50	368	289	269	273
	100	345	273	256	275
	200	368	274	236	263
	300	364	262	247	258
HallusionBench	Baseline	174	109	96	139
	50	156	115	102	142
	100	172	114	96	136
	200	173	116	96	137
	300	174	109	96	139

To avoid unreliable behavior in the long-tail region, we restrict the analysis to token positions supported by a sufficient number of samples. Specifically, a token index is included only if it is present in at least 20% of the samples and corresponds to a minimum of 5 valid generations. Consequently, different curves may end at different token positions, depending on when the available sample count falls below this support criterion, which can occur due to shorter reasoning traces or decoding length limits.

Table 5: Average number of generated tokens across all samples, when visual tokens are dropped at different layers across four benchmarks and four models. We drop at layer L - offset, where L is the final layer index.

Benchmark	Drop	Qwen3-VL-2B	Qwen3-VL-4B	Qwen3-VL-8B	Qwen3-VL-32B
WeMath	Baseline	499	370	345	409
	Offset 5	514	384	354	399
	Offset 10	595	422	381	401
	Offset 15	555	694	556	707
LogicVista	Baseline	720	475	354	667
	Offset 5	766	416	309	685
	Offset 10	1006	530	494	680
	Offset 15	587	818	665	1142
MMStar	Baseline	363	261	242	253
	Offset 5	364	271	235	263
	Offset 10	457	312	273	268
	Offset 15	330	372	323	384
HallusionBench	Baseline	174	109	96	139
	Offset 5	197	120	102	140
	Offset 10	324	230	172	161
	Offset 15	168	268	177	289

Table 6: Average number of generated tokens for the combined dropping across both time and depth, at different layer offsets and generation steps.

Benchmark	Offset	Drop after	Qwen3-VL-2B	Qwen3-VL-4B	Qwen3-VL-8B	Qwen3-VL-32B
WeMath	Baseline		532	376	350	416
	5	200	517	389	358	416
	5	300	527	385	360	401
	10	200	567	427	387	421
	10	300	565	420	383	409
LogicVista	Baseline		746	438	373	664
	5	200	748	501	358	648
	5	300	750	425	327	682
	10	200	1020	577	507	698
	10	300	1003	548	514	697
MMStar	Baseline		351	266	227	251
	5	200	375	276	242	265
	5	300	371	277	242	258
	10	200	455	331	296	278
	10	300	452	330	286	272
HallusionBench	Baseline		182	110	96	136
	5	200	216	118	102	134
	5	300	216	117	101	136
	10	200	344	223	161	164
	10	300	346	222	161	166

F PROMPTS

We provide the prompt used for all our benchmarks in Table 7.

Table 7: Prompt templates used for evaluation on the four multimodal benchmarks. Dataset-specific content is represented using curly-braced placeholders. The prompts were kept fixed for all samples within each benchmark.

Benchmark	Task / Output	Prompt
HallusionBench	Visual yes/no question. Output: `yes` or `no`.	Now, we require you to solve a yes/no question shown in the image. Please briefly describe your thought process and provide the final answer (yes or no). **Question:** `{question}` Regarding the format, please answer following the template below, and be sure to include two <> symbols: `<Thought process>: «your thought process» <Answer>: «your option»`
WeMath	Multiple-choice mathematical reasoning. Output: option.	Now, we require you to solve a multiple-choice math question. Please briefly describe your thought process and provide the final answer (option). **Question:** `{question}` **Option:** `{options}` Regarding the format, please answer following the template below, and be sure to include two <> symbols: `<Thought process>: «your thought process» <Answer>: «your option»`
LogicVista	Visual multiple-choice reasoning. Output: option.	Now, we require you to solve a multiple-choice question shown in the image. Please briefly describe your thought process and provide the final answer (option). **Question:** `{question}` Regarding the format, please answer following the template below, and be sure to include two <> symbols: `<Thought process>: «your thought process» <Answer>: «your option»`
MMStar	Multiple-choice multimodal reasoning. Output: option.	Now, we require you to solve a multiple-choice question. Please briefly describe your thought process and provide the final answer (option). **Question:** `{question}` Regarding the format, please answer following the template below, and be sure to include two <> symbols: `<Thought process>: «your thought process» <Answer>: «your option»`

G AI Assistance Disclosure

LLMs were used solely for language editing, including improving grammar, clarity, readability, sentence structure, and overall writing quality. These tools were not used to generate the scientific ideas, hypotheses, methodological contributions, results, and figures presented in this work. All technical content, scientific claims, and interpretations were developed and verified by the authors, who take full responsibility for the content of the manuscript.