AdaptInfer: Adaptive Token Pruning for Vision-Language Model Inference via Dynamical Text Guidance

Weichen Zhang (‡)[1,2], **Zhui Zhu** (‡)[2], **Ningbo Li**[2,3], **Shilong Tao**[4], **Hongzi Zhu**[5], **Jingao Xu**[1], **Kebin Liu** (✉)[2], **Yunhao Liu**[2]
[1]The University of Hong Kong, [2]Tsinghua University,
[3]The Hong Kong University of Science and Technology, [4]Peking University,
[5]Shanghai Jiao Tong University
weic.zhang@connect.hku.hk, z-zhu22@mails.tsinghua.edu.cn, nlibc@connect.ust.hk,
shilongtao@stu.pku.edu.cn, hongzi@sjtu.edu.cn, jingaoxu@hku.hk,
{kebinliu2021, yunhao}@tsinghua.edu.cn

Abstract

Vision–language models (VLMs) have achieved impressive performance on multimodal inference tasks, but the cost remains a significant challenge due to the large number of vision tokens processed during the prefill stage. Existing token pruning methods often rely on utilizing the static attention patterns directly, failing to exploit the dynamic internal signals within VLMs. To address the issue, we propose AdaptInfer, a novel plug-and-play framework for vision token pruning. First, we introduce a dynamic text-guided pruning mechanism that construct soft priors over text-token importance on each pruning layer, allowing more informed scoring of vision tokens at each stage. Second, we observe a highly consistent distribution of cross-modal attention shifts by architecture, which inspires us to introduce a efficient data-driven schedule which determines the pruning locations automatically. Experimental results have verified the effectiveness and generalization of the proposed method. Under the same token budget, AdaptInfer surpasses existing approaches in accuracy. For instance, AdaptInfer maintains averagely 99.4% accuracy on Qwen2-VL while 70% of the prefilling vision token overhead is reduced. The source code is available on: https://github.com/weiczh02/AdaptInfer-base.

1 Introduction

Vision–language models (VLMs) have achieved strong performance on multimodal tasks such as captioning (Lin et al., 2014), retrieval (Faghri et al., 2018), and visual question answering (Antol et al., 2015), by coupling visual encoders (Liu et al., 2022; Dosovitskiy et al., 2021) with large language model decoders (Touvron et al., 2023a; Brown et al., 2020; Du et al., 2022). Nevertheless, this architecture introduces a major inference bottleneck: visual inputs are represented by far more tokens than textual prompts. For example, a 672×672 image with a 14×14 patch size normally produces 2304 visual tokens (Radford et al., 2021; Zhang et al., 2025b), while the text prompt usually contains fewer than 100 tokens (Hudson and Manning, 2019; Fu et al., 2025). Since vision tokens are often redundant and semantically repetitive (Zhang et al., 2025a; Tong et al., 2025), reducing them has become a central direction for VLM acceleration.

Existing methods mainly prune, merge, or compress visual tokens either in the visual encoder (Li et al., 2024; Chen et al., 2025) or during the LLM prefill stage (Lin et al., 2025; Xing et al., 2025; Chen et al., 2024a; Li et al., 2025). Recent attention-based methods further show that not all attention signals are equally useful for token ranking (Zhao et al., 2026). VisPruner (Zhang et al., 2025a) observes that full attention patterns can be noisy, while SparseVLM (Zhang et al., 2025b) improves pruning by selecting salient text tokens as guidance before inference. Despite effectiveness, these methods still rely on static guidance. However, the informativeness of text tokens evolves as the model progressively refines its internal representations and activation patterns (Tenney et al., 2019; Clark et al., 2019; Zhu et al., 2026). As shown in Figure 1, the most important text tokens vary substantially across layers, suggesting that static text-token selection may lead to suboptimal pruning decisions and affect the performance.

Therefore, to truly harness the benefits of text-guided sparsification, it is crucial to develop a pruning strategy matched with dynamic fluidity of information, reflecting the effective cross-modal interaction throughout the inference process. Therefore, we propose to reconstruct the dynamic importance ranking of text tokens at each layer by utilizing the text-to-text (t2t) attention maps. These attentions provide a natural, layer-specific prior distribution for text-token importance, which we then use to

(‡)Equal contribution.
(✉)Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 19920–19938
October 24-29, 2026 ©2026 Association for Computational Linguistics

reweight text-to-vision (t2v) attention scores for vision token ranking. Since these attention blocks are already available from the standard forward pass and are queried only at selected pruning layers, AdaptInfer introduces minor additional overhead.

Another challenge is how to choose pruning locations. Prior methods often rely on empirical rules or extensive hyperparameter search (Xing et al., 2025; Chen et al., 2024a; Zhao et al., 2026), which can be computationally expensive and dataset-specific. We instead analyze layer-wise cross-modal attention shifts during inference and observe consistent shift patterns across datasets and architectures. For example, attention shifts concentrate around layers 1 and 10-20 in LLaVA-1.5-7B (Liu et al., 2023), and around layers 10-19 in Qwen2-VL-2B (Wang et al., 2024a). These observations provide a simple, data-driven basis for selecting effective pruning layers without exhaustive tuning.

The solutions we propose effectively address the limitations of existing works in the field of vision token sparsification for VLM acceleration. Our main contribution involves:

- We propose **AdaptInfer**, an adaptive vision token pruning method which dynamically determines text token guidance during inference. AdaptInfer is a plug-and-play solution.
- We identify consistent cross-modal attention-shift patterns and use them to derive a simple, data-driven pruning schedule.
- We implement and evaluate our proposed solution, AdaptInfer, across multiple benchmarks and VLMs under different vision token budget settings. Within the same token budget, our AdaptInfer outperforms state-of-the-art (SOTA) methods on the inference accuracy.

2 Related Work

In this section, we will briefly introduce the previous works that are correlated with ours.

2.1 Vision-Language Models

Early multi-modal systems paired convolutional vision backbones with recurrent language decoders (Karpathy and Fei-Fei, 2017; Vinyals et al., 2015). Modern VLMs instead follow the Transformer paradigm (Vaswani et al., 2017) by representing an image as a sequence of *visual tokens* that interact with textual tokens in a shared self-attention space (Liu et al., 2023; Chen et al., 2024b). BLIP-2 (Li et al., 2023a) and MiniGPT-4 (Zhu et al., 2023) introduce lightweight linear adapters that project features from a frozen CLIP encoder into the hidden space of a large language model, enabling efficient training. These explorations bridge the gap between frozen encoders and LLMs. The LLaVA family (Liu et al., 2024a, 2023, 2024b) refines this recipe with stronger instruction tuning, while a few other efforts such as Flamingo (Alayrac et al., 2022) and CogVLM (Wang et al., 2024b) scale the approach to billions of parameters.

Recent explorations also contribute to resolution and multi-modal extensions. Models leveraging hierarchical perception (e.g., LLaVA-NeXT (Liu et al., 2024b)) and adaptive patching (e.g., Qwen-VL (Bai et al., 2023)) permit high-resolution inputs, while large-scale vision encoders are adopted to generate richer hidden states of the vision tokens (Chen et al., 2024b; Zhu et al., 2025). However, these gains come at the cost of dramatically more vision tokens, which is a key bottleneck we target at.

2.2 Inference Acceleration of VLMs

Previous approaches mainly focus on vision token sparsification. This is because the number of vision tokens is often an order of magnitude (or more) larger than that of text tokens. In addition, visual embeddings are naturally much more sparse and repetitive than human-made texts (Marr, 2010). In this field, there are two research directions including efficient vision encoders and vision token pruning in LLM networks.

First, methods like LLaVA-PruMerge (Shang et al., 2025), VisionZip (Yang et al., 2024), and VisPruner (Zhang et al., 2025a) follow the first direction, which cuts the encoder outputs or redesign the projector to produce fewer vision tokens. Recoverable compression (Chen et al., 2025) is also introduced to repair the information loss from token pruning within vision encoders. Solutions follow the second direction not only to drop the vision tokens (Chen et al., 2024a; Lin et al., 2025; Xing et al., 2025; Li et al., 2025; Luan et al., 2025; Ye et al., 2025) during prefill, but also merge them to compress the numbers of the vision tokens (Bolya et al., 2023). SparseVLM (Zhang et al., 2025b) tries to go deeper in exploration of static text prompt guidance but ignoring the evolving inherent of token information. Our approach contributes to the second paradigm.

More broadly, structured representations have

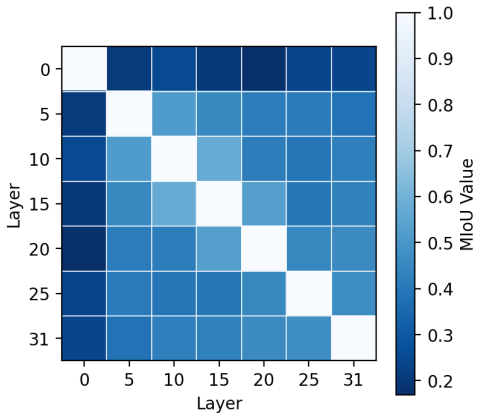
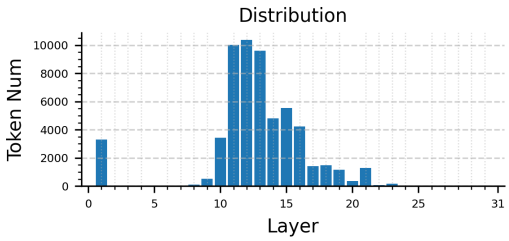
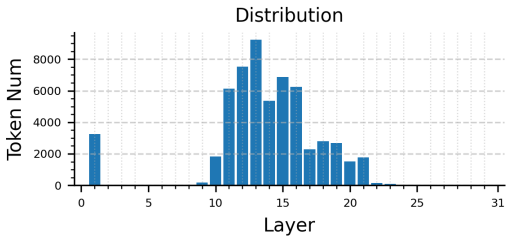


Figure 1: **MIoU of key text tokens in different layers.** The consistent low mIoU shows the key text token varies across layers.



(a) Attention shift distribution on MME.



(b) Attention shift distribution on TextVQA.

Figure 2: **Highly consistent distribution of attention shifts on MME and TextVQA.**

been explored as an effective way of capturing complex relationships and dependencies in learning-based systems (Tao et al., 2025, 2026). Recent studies have further leveraged such explicit structural modeling, like graph-structured representations, to guide visual token pruning in multimodal models (Jiang et al., 2025; Yang et al., 2025).

3 Method

In this section, we will introduce our observations and the methods we propose.

3.1 Observations

We present a few observations and preliminary experiments that have illuminated our insights.

3.1.1 Text Token Importance

SparseVLM (Zhang et al., 2025b) selects key text token offline for pruning guidance, which actually based on an assumption that the text token importance remains static across all prefilling layers. Since this assumption is not trivial, we examine it as follows.

Given an input question prompt, we first tokenize it as $Prompt = [t_1, \ldots, t_n]$. We then define the *importance* of a text token t_i at layer ℓ as the total attention weight it receives from all text tokens in the layer-ℓ text-to-text (t2t) attention map on average of attention heads:

$$\mathrm{Imp}_\ell(t_i) = \frac{1}{H} \sum_{h=1}^{H} \sum_{j=1}^{n} \mathbf{A}_{\text{t2t}}^{(\ell,h)}[j, i], \qquad (1)$$

where $\mathbf{A}_{\text{t2t}}^{(\ell,h)}[j, i]$ denotes the attention of head h directed from token t_j to t_i, H represents the number of the attention heads. Intuitively, tokens with high Imp_ℓ are the current key text tokens of the language stream and are therefore best suited to guide cross-modal pruning decisions at that layer.

3.1.2 Dynamics of Text Token Importance

To examine whether text-token importance remains stable across layers, we conduct an empirical study on LLaVA-1.5-7B (Liu et al., 2023) using 1,000 samples from TextVQA. For each selected layer, we compute text-token importance from the text-to-text attention map and select the top 20% tokens as key text tokens. We then measure the mean Intersection over Union (mIoU) of these key-token sets across layers, as shown in Figure 1.

The consistently low off-diagonal mIoU values indicate that key text tokens vary substantially during inference. For example, the mIoU between layer 0 and layer 24 is only 0.169, meaning that most selected key tokens differ across these two layers. This suggests that VLMs attend to different parts of the input question at different reasoning stages. Consequently, any static text-prompt–guided pruning approach is likely to fail in capturing this evolving semantic alignment. These findings motivate our dynamical, layer-wise text-guidance mechanism.

3.1.3 Cross-Attention Shifts of VLM

To derive pruning schedules from the model's own inference dynamics rather than from trial-and-error tuning, we analyze where cross-modal attention shifts occur across layers. Specifically, for each

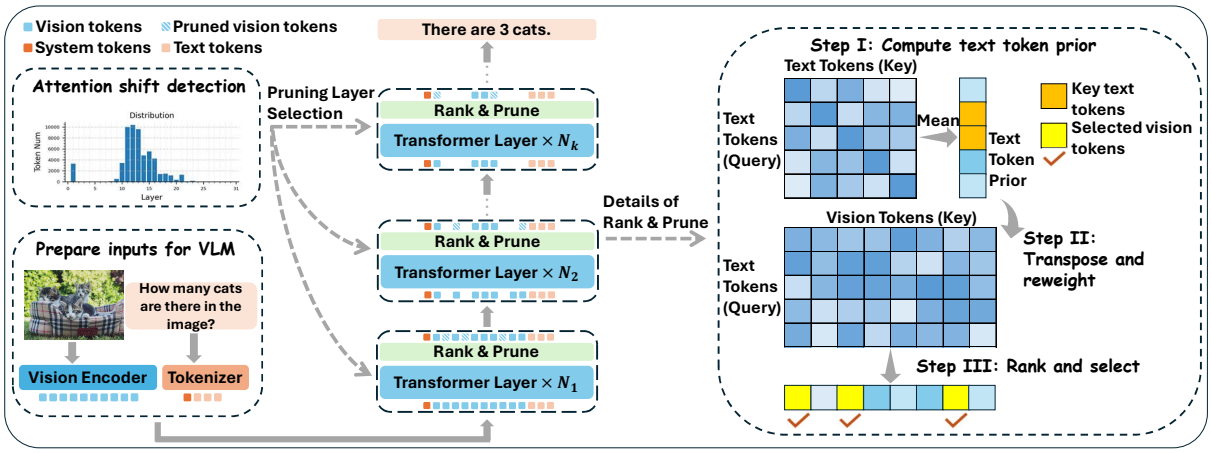


Figure 3: **The architecture of AdaptInfer**. Text token importance is computed and guides vision token selection at every pruning layer adaptively. The right panel illustrates the implementation details of the Rank & Prune module.

sample in LLaVA-1.5-7B (Liu et al., 2023), we compute cumulative text-to-vision (t2v) attention scores for visual tokens and focus on the top 10% vision tokens receiving the highest total attention in the prefill stage. These tokens are treated as the most task-relevant visual evidence for the corresponding multimodal input.

We then apply change-point detection (Truong et al., 2020) to the layer-wise cumulative attention trajectory of each selected token, identifying layers where its attention pattern changes significantly. Appendix B shows the technical details. Such shift points indicate layer-wise transitions in visual information usage, either when a token becomes salient or when its information has largely been consumed.

Figure 2 aggregates the detected shift locations over 1,000 samples from both MME (Fu et al., 2025) and TextVQA (Singh et al., 2019). Despite their different task formulations and data distributions, the two datasets exhibit highly consistent patterns: attention shifts concentrate around layer 1 and layers 10–20, while layers 2–9 and later layers are attention stable regions. This consistency suggests that the observed attention dynamics are more likely to reflect architecture-level behavior than dataset-specific artifacts. Therefore, these shift distributions provide a data-driven basis for selecting pruning layers according to stable attention regions, rather than relying solely on empirical heuristics.

3.2 Adaptive Token Pruning

Based on the two key observations above, we propose AdaptInfer for VLM inference acceleration.

3.2.1 Dynamic Text Guidance Pruning

To guide visual token pruning in a more informed and adaptive way, we propose a dynamic text-guidance mechanism that leverages the model's internal attention signals. Instead of statically selecting a fixed subset of text tokens before entering the language model (Zhang et al., 2025b), we dynamically infer their relative importance during inference at each predefined pruning layer. We reuse the attention maps already computed by the model. The architecture of Our AdaptInfer is shown in Fig 3, and our dynamic text guidance mechanism can be divided into three steps.

Firstly, in each pruning layer, we extract the *text-to-text* (t2t) attention matrix $\mathbf{A}^{h}_{\text{t2t}} \in \mathbb{R}^{T\times T}$, where T is the number of text tokens, h is the index of attention head with a total number of H. To estimate the importance of each text token, we aggregate attention scores across the query dimension:

$$\mathbf{w}^{(\mathbf{h})} = \sum_{i=1}^{T} \mathbf{A}^{(h)}_{\text{t2t}}[i,:] \in \mathbb{R}^{T}, \tag{2}$$

where $\mathbf{w}$ serves as a soft prior distribution over the text tokens, averaged across all attention heads, indicating how much attention each token receives from the rest of the sequence.

Then, we reweight the t2v attention matrix $\mathbf{A}^{(h)}_{\text{t2v}} \in \mathbb{R}^{T\times V}$ with this prior, where V is the number of visual tokens remained. $\mathbf{A}^{(h)}_{\text{t2v}}$ denotes the cross-attention matrix where text tokens are used as queries, and vision tokens serve as keys and values. The importance score of each visual token on average of all attention heads is computed as:

$$\mathbf{s} = \frac{1}{H}\sum_{h=1}^{H} \mathbf{w}^{(h)\top} \cdot \mathbf{A}_{\text{t2v}}^{(h)} \in \mathbb{R}^{V}. \tag{3}$$

Here, $\mathbf{s}_j$ reflects the aggregated and weighted attention from all text tokens to visual token j.

Finally, based on these scores, we rank all visual tokens and retain the top-k for the current layer:

$$\mathcal{I}_k = \text{TopK}(\mathbf{s}, k). \tag{4}$$

Importantly, all text tokens participate in visual token scoring, but contribute in proportion to their dynamically inferred importance. Moreover, because both $\mathbf{A}_{\text{t2t}}$ and $\mathbf{A}_{\text{t2v}}$ are natively computed in standard forward passes, our method introduces little additional computational overhead. Note that this solution follows a training-free paradigm and can be seamlessly integrated as a plugin into existing VLMs. The detailed FLOPs derivations are also provided in Appendix C.

3.2.2 Heuristic Pruning Schedule

Following the attention-shift analysis described above, we design a pruning schedule that aligns with the layer-wise evolution of cross-modal attention. In our implementation, pruning at layer ℓ indicates to perform token pruning after the forward computation of layer ℓ: we use the attention scores produced at this layer to rank vision tokens, remove low-ranked tokens, and feed only the remaining tokens into layer $\ell+1$. Therefore, a pruning location should be interpreted as the boundary after layer ℓ.

Prior works (Chen et al., 2024a; Xing et al., 2025; Zhang et al., 2025b) directly rely on attention scores to rank tokens and select pruning locations manually. However, we argue that attention scores from different transformer layers are not equally reliable as pruning evidence. Our attention-shift analysis reveals two typical regimes. In high-variation regions, token importance is actively being reassigned, with tokens either becoming newly important or having already been exploited by the model. Attention rankings in these regions are unstable, making pruning decisions less reliable. In stable regions, by contrast, token importance rankings remain relatively consistent and therefore provide more reliable pruning signals.

Formally, let the model have L transformer layers and let K denote the default number of pruning stages. From the offline attention-shift analysis, we detect the starting layers of stable regions:

$$\begin{aligned} &\mathcal{S} = \{s_1, s_2, \ldots, s_N\}, \\ &0 \le s_1 < s_2 < \cdots < s_N < L, \end{aligned} \tag{5}$$

where N is the number of detected stable-region starts. Since pruning at layer ℓ removes tokens before layer $\ell+1$, a stable region starting at s_i should be associated with the boundary right before it, i.e., after layer $s_i - 1$. Pruning at this boundary allows the entire following stable region to operate on fewer visual tokens, thus maximizing computation savings while still relying on a stable attention-ranking regime. We therefore convert the detected stable-region starts into a set of pruning layer candidates:

$$\tilde{\mathcal{S}} = \{\tilde{s}_i \mid \tilde{s}_i = \max(s_i - 1, 0),\ i = 1, \ldots, N\}. \tag{6}$$

If $N \ge K$, we choose the first K candidates as the pruning layers:

$$\mathcal{P} = \{\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_K\}. \tag{7}$$

If $N < K$, we keep all detected candidates and fill the remaining $K - N$ pruning layers by uniform interpolation between the earliest two pruning candidates. This interpolation aims to distribute pruning across depth. It places additional pruning stages early to maximize computation savings, while maintaining sufficient layer intervals for each pruning step to take effect. Notably, the interpolation strategy is an efficiency-oriented scheduling heuristic rather than another stable guarantee. Specifically, the additional pruning layers are

$$\begin{aligned} \mathcal{U} = \Big\{ \Big\lfloor \tilde{s}_1 + \frac{j}{K - N + 1}(\tilde{s}_2 - \tilde{s}_1) \Big\rfloor \\ \mid\ j = 1, 2, \ldots, K - N \Big\}. \end{aligned} \tag{8}$$

The final pruning schedule is then

$$\mathcal{P} = \text{Sort}(\tilde{\mathcal{S}} \cup \mathcal{U}). \tag{9}$$

We set $K = 3$ in our experiments, the ablation study on K is included in Table 7. In our experiments, the attention-shift analysis typically identifies two stable regions on Qwen, LLaVA, and InternVL models. Thus, under the default setting $K = 3$, the schedule reduces to selecting the two stable-region boundaries and their midpoint. For example, on LLaVA-1.5-7B, the detected stable starts correspond to pruning candidates layers

Table 1: **Comparison of methods under different pruning budgets on LLava-1.5-7B.**

Method	Avg. Tokens	MME	GQA	MMB	SQA	TextVQA	MMVet	Ratio (%)
Vanilla	576	1864	61.9	64.6	69.5	58.3	31.1	100
ToMe (ICLR 23)	128	1343	52.4	53.3	59.6	49.1	27.2	82.8 (↓ 17.2)
FastV (ECCV 24)	128	1490	49.6	56.1	68.6	52.5	26.3	87.9 (↓ 12.1)
PDrop (CVPR 25)	128	1761	57.1	61.6	68.4	56.6	28.1	94.7 (↓ 5.3)
SparseVLM (ICML 25)	128	1746	58.4	64.5	68.6	56.7	29.0	96.2 (↓ 3.8)
DART (EMNLP 25)	128	1804	57.7	62.3	69.4	55.9	29.8	96.3 (↓ 3.7)
A-Bias (arXiv 26.1)	128	1771	58.8	63.8	68.0	56.6	29.2	96.3 (↓ 3.7)
AdaptInfer (Ours)	128	1794	58.5	63.8	69.9	56.8	29.5	97.1 (↓ 2.9)
ToMe (ICLR 23)	64	1138	48.6	43.7	50.0	45.3	24.1	72.4 (↓ 27.6)
FastV (ECCV 24)	64	1255	46.1	47.2	68.7	45.9	19.6	75.9 (↓ 24.1)
PDrop (CVPR 25)	64	1561	47.5	58.8	69.0	50.6	22.5	85.0 (↓ 15.0)
SparseVLM (ICML 25)	64	1589	53.8	60.1	69.8	53.4	24.9	89.5 (↓ 10.5)
DART (EMNLP 25)	64	1533	51.2	55.8	68.7	50.4	25.1	86.2 (↓13.8)
A-Bias (arXiv 26.2)	64	1623	54.0	59.7	69.2	53.4	25.9	90.1 (↓ 9.9)
AdaptInfer (Ours)	64	1684	53.2	61.7	69.9	54.3	25.9	91.5 (↓ 8.5)

1 and 20, yielding the schedule $\{1, 10, 20\}$. For LLaVA-1.5-13B, we use $\{1, 11, 22\}$; for Qwen2-VL-2B (Wang et al., 2024a), we use $\{0, 9, 19\}$. Additional experiments on LLaVA-1.5-13B and Qwen2-VL-2B are presented in Appendix E and Appendix F. The proposed offline analysis is lightweight: it is performed only once for each VLM and typically finishes within minutes.

3.2.3 Discussion

AdaptInfer only accesses attention at the selected pruning layers, at most three in our experiments, and uses only the *text-to-text* ($t2t$) and *text-to-vision* ($t2v$) blocks for token ranking, without materializing full attention maps or the much larger *vision-to-vision* ($v2v$) block. This can be implemented with the FlashAttention-compatible dual-pass strategy used in SparseVLM (Zhang et al., 2025b). Since attention is queried only at a few pruning layers and only for the required sub-blocks, the resulting end-to-end latency overhead is modest in practice. We defer the implementation details and complexity discussion to Appendix D.

4 Experiment

In this section, we empirically evaluate our proposed approach through a series of experiments.

4.1 Experimental Settings

4.1.1 Datasets

For multimodal evaluation, we test our solutions on several widely used benchmarks, including MME (Fu et al., 2025), GQA (Hudson and Manning, 2019), MMBench (MMB) (Liu et al., 2024c), ScienceQA (SQA) (Lu et al., 2022), TextVQA (Singh et al., 2019), MMVet (Yu et al., 2024), and POPE (Li et al., 2023b). These datasets together provide a comprehensive evaluation, including vision-question-answering (VQA), optical character recognition (OCR), perception, reasoning, factual grounding and so on. In addition, we also test our solutions on three popular video-based multimodal benchmarks, TGIF-QA (TGIF) (Jang et al., 2019), MSVD-QA (MSVD) and MSRVTT-QA (MSRVTT) (Xu et al., 2017).

4.1.2 Baselines

We select six representative vision-token sparsification methods that operate in a plug-and-play manner during the LLM forward pass, including FastV (Chen et al., 2024a), ToMe (Bolya et al., 2023), Pyramid Drop (PDrop) (Xing et al., 2025), SparseVLM (Zhang et al., 2025b), DART (Wen et al., 2025), and Attention-Bias (A-Bias) (Zhao et al., 2026). For PDrop, we adopt its training-free version to ensure a fair comparison under the same deployment setting. For Attention-Bias, we use its PDrop-based variant for evaluation.

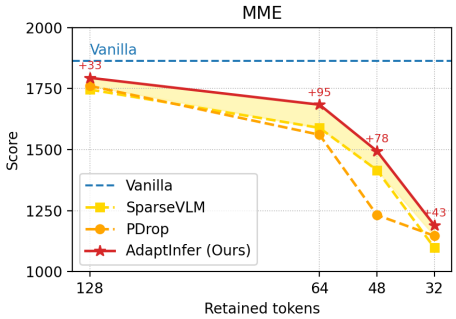
We focus the main comparison on methods in which visual token sparsification happens in the language model prefill stage in Section 4.2. Methods that reduce visual tokens at different locations, such as vision-encoder-side compression (Yang et al., 2024), are orthogonal to our prefill-stage pruning. To verify this complementarity, we further combine VisionZip with AdaptInfer and report the results in Section 4.3.

4.1.3 Implement Details

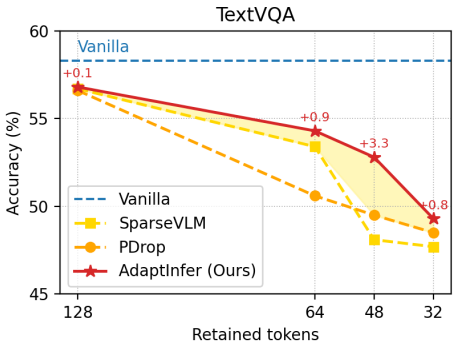
To ensure a comprehensive comparison between baselines and AdaptInfer, we report the results in

Table 2: **Comparison of Performance on Qwen2-VL-2B.**

Methods	Tokens	Image-Based					Video-Based			Ratio
		MME	TextVQA	GQA	MMB	POPE	TGIF	MSRVTT	MSVD	
Vanilla	100%	1901	77.8	60.4	71.7	86.9	9.9	30.9	42.3	100%
SparseVLM	50%	1900	76.9	59.9	71.0	86.4	11.7	31.2	42.9	102.1%
AdaptInfer	50%	1909	76.6	60.0	72.1	86.7	11.1	30.5	43.6	101.7%
SparseVLM	30%	1867	73.6	57.5	70.1	84.6	9.3	29.5	41.8	96.4%
AdaptInfer	30%	1885	73.9	58.5	71.4	85.8	10.4	30.2	43.5	99.4%
SparseVLM	10%	1460	51.3	40.4	57.3	40.3	5.0	28.1	30.2	68.6%
AdaptInfer	10%	1721	62.3	48.8	63.7	71.3	5.9	28.2	41.4	81.6%



(a) Performance trends on MME.



(b) Performance trends on TextVQA.

Figure 4: **Performance trends of AdaptInfer on two representative benchmarks.**

different average retained token budgets. AdaptInfer utilizes the pruning parameters described above while the baselines keep their original settings for all core components. Our experiments covers three different types of VLMs, including LLaVA-1.5-7B (Liu et al., 2023), Qwen2-VL-2B (Wang et al., 2024a), and InternVL3-2B (Zhu et al., 2025). Additional experiments on LLaVA-1.5-13B are presented in appendix E.2.

4.2 Main Results on LLaVA

Table 1 reports the results on LLaVA-1.5-7B under average vision-token budgets of 128 and 64. AdaptInfer achieves the best overall performance in both settings, retaining 97.1% and 91.5% of the full-token accuracy, respectively. Notably, with only 64 retained tokens, it reduces the prefill vision-token load by 88.9% while outperforming the strongest baselines SparseVLM and A-Bias by 2.0% and 1.4%. Although the absolute gains appear modest, they are achieved in a near-saturated regime where all strong baselines are already close to the full-token upper bound. Moreover, we shows the consistent gains of AdaptInfer over SparseVLM and PDrop across token budgets from 128 to 32 in figure 4.

Table 3: **Extreme pruning comparsions.**

Method	Tokens	MME	TextVQA	Ratio (%)
Vanilla	576	1864	58.3	100
SparseVLM	48	1416	48.1	79.2
AdaptInfer	48	1494	52.8	85.4
SparseVLM	32	1098	47.7	70.4
AdaptInfer	32	1190	49.3	74.2

We further evaluate AdaptInfer on a more recent VLM Qwen2-VL (Wang et al., 2024a) to examine its generalization across different VLM architectures. For Qwen2-VL, which dynamically adjusts the number of visual tokens and supports both image and video tasks, we compare AdaptInfer with SparseVLM under the same retained-token ratios of 50%, 30%, and 10%. As shown in Table 2, AdaptInfer consistently matches or outperforms SparseVLM across eight benchmarks, including three video QA datasets TGIF, MSRVTT, and MSVD. Notably, at the 50% retained ratio, both pruning methods slightly outperform the full-token model, suggesting that removing redundant visual tokens can reduce noise while improving efficiency. For more implementation details of AdaptInfer on Qwen2-VL, please refer to Appendix K.4.

We also examine our method under extreme pruning budgets on LLaVA-1.5-7B, which is pre-

Table 4: **Complementarity with vision-encoder-side compression.**

Method	Tokens	TVQA	MME	MMB	POPE	GQA	Ratio
Vanilla	576	58.3	1864	64.6	85.9	61.9	100%
VisionZip	64	55.5	1690	60.1	77.0	55.1	91.5%
VisionZip + AdaptInfer	64	56.5	1724	62.8	83.3	56.6	95.0%

Table 5: **Latency test of AdaptInfer.**

Method	Tokens	Metrics	MME	GQA	MMB	TextVQA	Average
Vanilla	576	FLOPs (T)	4.268	4.250	4.623	4.611	4.438
		Latency (ms)	82.0	76.5	91.3	91.2	85.3
PDrop	64	FLOPs (T)	0.975	0.958	1.316	1.305	1.138 (↓ 74.3%)
		Latency (ms)	33.0	32.0	36.4	36.6	34.5 (↓ 59.6%)
SparseVLM	64	FLOPs (T)	0.974	0.958	1.316	1.305	1.138 (↓ 74.3%)
		Latency (ms)	34.4	34.7	38.1	39.5	36.7 (↓ 57.0%)
AdaptInfer (Ours)	64	FLOPs (T)	0.975	0.959	1.317	1.305	1.139 (↓ 74.3%)
		Latency (ms)	31.0	30.9	34.5	35.5	33.0 (↓ 61.3%)

sented in Table 3. AdaptInfer still has considerable performance and consistently outperforms SparseVLM when retaining only 48 or 32 visual tokens on average, preserving 85.4% and 74.2% of the full-token accuracy respectively.

4.3 Complementarity with Vision-Encoder Compression

AdaptInfer performs token pruning during the LLM prefill stage and is therefore orthogonal to methods that compress visual tokens before they enter the language model. To verify this complementarity, we combine AdaptInfer with VisionZip (Yang et al., 2024), a vision-encoder-side token compression method, and evaluate whether dynamic text-guided pruning remains effective after upstream visual token reduction. As shown in Table 4, adding AdaptInfer on top of VisionZip consistently improves performance across all evaluated benchmarks under the same average token budget of 64.

4.4 Efficiency Analysis

To approximate a resource-constrained deployment setting, we evaluate the practical efficiency of AdaptInfer on LLaVA-1.5-7B using CUDA latency measurements on a single NVIDIA RTX 4090 GPU with FLOPs estimation, as reported in Table 5. To be specific, we evaluate the same method-specific 64-token settings here whose accuracy results are reported in Table 4.2. All latency results are averaged per sample. Under the same 64-token budget, AdaptInfer achieves an average latency of 33.0 ms, lower than both PDrop (Xing et al., 2025) and SparseVLM (Zhang et al., 2025b). Although the theoretical FLOPs of these methods are comparable, AdaptInfer attains lower real latency because it introduces little additional computation, requiring neither extra attention computation nor offline feature matching.

Table 6: **Controlled ablation of different text priors on InternVL3-2B.**

Tokens	Text Prior	MME	TextVQA	SQA	POPE
30%	Single-token	1878	35.9	86.3	84.7
30%	Random	1990	48.6	88.6	86.4
30%	Static	1989	49.3	88.7	86.9
30%	Dynamic	2015	49.4	89.5	87.3

Table 7: **Performance of different pruning locations on LLava-1.5-7B.**

Tokens	K	Pruning Layers	MME	MMB
64	3	1,10,20	1684	61.7
64	2	1,20	1520	56.3
64	4	1,7,14,20	1645	60.7
64	3	0,10,20	1660	60.7
64	3	2,10,20	1589	60.1
64	3	1,9,20	1642	60.9
64	3	1,11,20	1656	61.7
64	3	1,10,19	1673	61.9
64	3	1,10,21	1675	61.5

4.5 Ablation Study

We first evaluate the necessity of dynamic text guidance through a controlled ablation study on InternVL3-2B (Zhu et al., 2025). We keep the pruning layers, token budget, per-stage pruning ratios, implementation, and evaluation protocol

identical, and change only the text-query weighting strategy. The results are shown in Table 6. In the table, *Random prior* indicates a prior sampled from a symmetric Dirichlet distribution (five runs averaged); *single-token prior* refers to a one-hot prior that assigns all weight to the first text token after the visual-token sequence; *static prior* is a SparseVLM-style fixed prior that remains unchanged across pruning stages, without the clustering-and-merging component of the full SparseVLM (Zhang et al., 2025b); and the last one is the proposed *dynamic prior*, which is recomputed at every pruning stage adaptively. Among all baselines, our dynamic prior reaches the best performance on all four benchmarks.

We further analyze the pruning schedule. Besides the default setting with $K = 3$, we compare variants with fewer or more pruning stages ($K = 2$ and $K = 4$) and perturb each pruning layer around the selected schedule. Table 7 shows that the observation-driven schedule $(1, 10, 20)$ achieves strong performance without exhaustive hyperparameter search. While nearby variants can occasionally obtain comparable scores on individual benchmarks, the selected schedule consistently lies in a high-performing region, supporting our claim that attention-shift analysis provides a simple and effective way to identify pruning locations.

5 Conclusion

This paper proposes AdaptInfer, a plug-and-play solution for accelerating VLM inference via dynamic text-guided token pruning. Moreover, we observe a highly consistent pattern of cross-modal attention shifts in VLMs and introduce a data-driven heuristic pruning schedule accordingly. AdaptInfer introduces minimal additional computational overhead while maintaining higher accuracy than existing approaches. For example, AdaptInfer reduces CUDA latency by 61.3% while retaining an average accuracy of 91.5% on LLaVA-1.5-7B. On Qwen2-VL-2B, it retains an average accuracy of 99.4% even when only 30% of the vision tokens are retained on average across layers.

Limitations

Attention Accessibility. AdaptInfer relies on accessing selected attention blocks during inference, including t2t and t2v attentions. Therefore, AdaptInfer can be directly applicable to open-source VLMs or deployment frameworks in principle (we have evaluated its generalization on LLaVA-1.5, Qwen2-VL, and InternVL3), but it is hard to be applicable to closed-source or API-only systems.

Pruning Schedule Transferability. Although AdaptInfer avoids expensive grid search for pruning locations, its pruning schedule is still derived from an offline attention-shift analysis. As a result, applying the method to a new architecture may also require a lightweight calibration step.

Aggressive Pruning. AdaptInfer improves the accuracy-efficiency trade-off, but aggressive pruning can still remove fine-grained visual evidence. This can affect tasks that rely on dense or localized visual information, such as counting and small-object recognition. Moreover, multi-turn dialogue may present an extra challenge: visual regions that are irrelevant to the current query could become important in subsequent turns, while their corresponding tokens may have already been removed.

Attention as Importance Proxy. Our method uses attention scores as an efficient proxy for token importance. While this proxy works well empirically, attention weights may not always reflect the causal contribution of each token to the final prediction, which remains a broader limitation of attention-based pruning methods.

Ethical Considerations

This work aims to advance the grounding of LLMs in vision. We do not introduce new datasets, collect human-subject data, or use private or sensitive data. This work may make the multimodal LLMs easier and cheaper to deploy. Our method builds on existing VLMs and does not introduce additional ethical risks beyond those already associated with the underlying models.

We use publicly available models, datasets, and baseline implementations in accordance with their respective licenses and terms of use.

Acknowledgment

This work was supported in part by the National Science and Technology Major Project under Grant 2022ZD0114903, and by National Natural Science Foundation of China under Grants No. 62432008 and 62472248.

The authors declare that they use generative AI tools for grammar refinement and language polishing only. The authors take full responsibility for the entire content of this paper.

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob L Menick, Sebastian Borgeaud, and 8 others. 2022. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 23716–23736. Curran Associates, Inc.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2425–2433.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *Preprint*, arXiv:2308.12966.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2023. Token merging: Your vit but faster. In *The Eleventh International Conference on Learning Representations*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 2024a. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LXXXI*, page 19–35, Berlin, Heidelberg. Springer-Verlag.

Yi Chen, Jian Xu, Xu-Yao Zhang, Wen-Zhuo Liu, Yang-Yang Liu, and Cheng-Lin Liu. 2025. Recoverable compression: A multimodal vision token recovery mechanism guided by text information. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(2):2293–2301.

Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, Ji Ma, Jiaqi Wang, Xiaoyi Dong, Hang Yan, Hewei Guo, Conghui He, Botian Shi, Zhenjiang Jin, Chao Xu, and 16 others. 2024b. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. *Preprint*, arXiv:2404.16821.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pages 16344–16359. Curran Associates, Inc.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Yifan Du, Zikang Liu, Junyi Li, and Wayne Xin Zhao. 2022. A survey of vision-language pre-trained models. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 5436–5443. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2018. Vse++: Improving visual-semantic embeddings with hard negatives. In *Proceedings of BMVC*.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, Rongrong Ji, Caifeng Shan, and Ran He. 2025. Mme: A comprehensive evaluation benchmark for multimodal large language models. *Preprint*, arXiv:2306.13394.

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

Yunseok Jang, Yale Song, Chris Dongjoo Kim, Youngjae Yu, Youngjin Kim, and Gunhee Kim. 2019. Video question answering with spatio-temporal reasoning. *Int. J. Comput. Vision*, 127(10):1385–1412.

Yutao Jiang, Qiong Wu, Wenhao Lin, Wei Yu, and Yiyi Zhou. 2025. What kind of visual tokens do we need? training-free visual token pruning for multi-modal large language models from the perspective of graph. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press.

Andrej Karpathy and Li Fei-Fei. 2017. Deep visual-semantic alignments for generating image descriptions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):664–676.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023a. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19730–19742. PMLR.

Yanwei Li, Chengyao Wang, and Jiaya Jia. 2024. Llama-vid: An image is worth 2 tokens in large language models. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part XLVI*, page 323–340, Berlin, Heidelberg. Springer-Verlag.

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

Yucheng Li, Huiqiang Jiang, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Amir H. Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, and Lili Qiu. 2025. MMInference: Accelerating pre-filling for long-context visual language models via modality-aware permutation sparse attention. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 34998–35020. PMLR.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755, Cham. Springer International Publishing.

Zhihang Lin, Mingbao Lin, Luxi Lin, and Rongrong Ji. 2025. Boosting multimodal large language models with visual tokens withdrawal for rapid inference. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(5):5334–5342.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024a. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26296–26306.

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

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, pages 34892–34916. Curran Associates, Inc.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. 2024c. Mmbench: Is your multi-modal model an all-around player? In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part VI*, page 216–233, Berlin, Heidelberg. Springer-Verlag.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. 2022. A convnet for the 2020s. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11966–11976.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*.

Bozhi Luan, Wengang Zhou, Hao Feng, Zhe Wang, Xiaosong Li, and Houqiang Li. 2025. Multi-cue adaptive visual token pruning for large vision-language models. *Preprint*, arXiv:2503.08019.

David Marr. 2010. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. The MIT Press.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

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

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. 2019. Towards vqa models that can read. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8309–8318.

Shilong Tao, Zhe Feng, Shaohan Chen, Weichen Zhang, Zhanxing Zhu, and Yunhuai Liu. 2026. Neural latent arbitrary lagrangian-eulerian grids for fluid-solid interaction. In *The Fourteenth International Conference on Learning Representations*.

Shilong Tao, Zhe Feng, Haonan Sun, Zhanxing Zhu, and Yunhuai Liu. 2025. Unisoma: A Unified Transformer-Based Solver for Multi-Solid Systems. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pages 59132–59153.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Jintao Tong, Wenwei Jin, Pengda Qin, Anqi Li, Yixiong Zou, Yuhong Li, Yuhua Li, and Ruixuan Li. 2025. Flowcut: Rethinking redundancy via information flow for efficient vision-language models. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 94946–94973. Curran Associates, Inc.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Charles Truong, Laurent Oudre, and Nicolas Vayatis. 2020. Selective review of offline change point detection methods. *Signal Processing*, 167:107299.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024a. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *Preprint*, arXiv:2409.12191.

Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, Jiazheng Xu, Bin Xu, Juanzi Li, Yuxiao Dong, Ming Ding, and Jie Tang. 2024b. Cogvlm: Visual expert for pretrained language models. *Preprint*, arXiv:2311.03079.

Zichen Wen, Yifeng Gao, Shaobo Wang, Junyuan Zhang, Qintong Zhang, Weijia Li, Conghui He, and Linfeng Zhang. 2025. Stop looking for "important tokens" in multimodal language models: Duplication matters more. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 9961–9980, Suzhou, China. Association for Computational Linguistics.

Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, and Dahua Lin. 2025. Pyramidrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. *Preprint*, arXiv:2410.17247.

Dejing Xu, Zhou Zhao, Jun Xiao, Fei Wu, Hanwang Zhang, Xiangnan He, and Yueting Zhuang. 2017. Video question answering via gradually refined attention over appearance and motion. In *ACM Multimedia*.

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. 2024. Visionzip: Longer is better but not necessary in vision language models. *Preprint*, arXiv:2412.04467.

Zhongyu Yang, Dannong Xu, Wei Pang, and Yingfang Yuan. 2025. Script: Graph-structured and query-conditioned semantic token pruning for multimodal large language models. *Transactions on Machine Learning Research*.

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

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. 2024. MM-vet: Evaluating large multimodal models for integrated capabilities. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 57730–57754. PMLR.

Qizhe Zhang, Aosong Cheng, Ming Lu, Renrui Zhang, Zhiyong Zhuo, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. 2025a. Beyond text-visual attention: Exploiting visual cues for effective token pruning in vlms.

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. 2025b. SparseVLM: Visual token sparsification for efficient vision-language model inference. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 74840–74857. PMLR.

Kai Zhao, Wubang Yuan, Yuchen Lin, Liting Ruan, Xiaofeng Lu, Deng-Ping Fan, Ming-Ming Cheng, and Dan Zeng. 2026. Attention debiasing for token pruning in vision language models. *Preprint*, arXiv:2508.17807.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2023. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *Preprint*, arXiv:2304.10592.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Xuehui Wang, Yue Cao, Yangzhou Liu, Xingguang Wei, Hongjie Zhang, Haomin Wang, Weiye Xu, and 32 others. 2025. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *Preprint*, arXiv:2504.10479.

Zhui Zhu, Weichen Zhang, Zhenghan Zhou, Yunhao Liu, and Fan Dang. 2026. Dynamicinfer: Runtime-aware sparse offloading for LLMs inference on a consumer-grade GPU. In *The Fourteenth International Conference on Learning Representations*.

A Appendix Statement

The following appendix sections provide supplementary technical materials intended to support reproducibility, including implementation details, experimental configurations, and auxiliary analyses not essential to the main claims.

B Change-Point Detection

To identify shifts in cross-attention across transformer layers, we employ change-point detection (Truong et al., 2020) on cumulative cross-attention curves. The intuition is that a significant change in attention behavior corresponds to a noticeable inflection in the cumulative attention distribution over layers.

Given a token's cross-attention curve across L layers, denoted as:

$$\mathbf{a} = [a_1, a_2, \ldots, a_L], \quad \text{where } a_\ell \in \mathbb{R} \tag{10}$$

we first compute the cumulative attention:

$$\mathbf{c} = \left[\sum_{i=1}^{1} a_i, \sum_{i=1}^{2} a_i, \ldots, \sum_{i=1}^{L} a_i \right] \tag{11}$$

This sequence $\mathbf{c}$ captures the aggregate build-up of attention, which we treat as a univariate time series. We then apply change-point detection to identify the most likely layer index where a shift in trend occurs.

We adopt the `ruptures` Python library[1] for offline detection. Specifically, we use the Binary Segmentation (Binseg) algorithm with an ℓ_2 cost model. This algorithm aims to find a segmentation point that minimizes the within-segment variance. Formally, given the cumulative curve $\mathbf{c} = [y_1, y_2, \ldots, y_L]$, we solve:

$$\min_b \left(\sum_{t=1}^{b} (y_t - \mu_1)^2 + \sum_{t=b+1}^{L} (y_t - \mu_2)^2 \right), \tag{12}$$

where μ_1 and μ_2 are the mean values of the first and second segments, respectively. This corresponds to the ℓ_2 cost model used in ruptures, which measures the homogeneity of each segment by its squared deviation from the mean.

We restrict the number of change points to 1, making the detection both efficient and robust. The returned breakpoint $b \in \{1, \ldots, L\}$ indicates the first significant shift in attention accumulation for the given token. This layer index is treated as the token's *cross-attention shift point*, which guides our downstream pruning strategy.

C Analysis of Computational Complexity

Assuming $n = T + V$ denotes the current sequence length, d denotes the VLM's hidden state dimension, and m denotes the hidden size of projection layer in the FFN network. For each transformer layer in the prefill stage, the FLOPs can be estimated by

$$FLOPs^{\text{prefill}} = 4nd^2 + 2n^2d + 3ndm. \tag{13}$$

For each pruning layer, the additional FLOPs of are computed below:

$$FLOPs^{\text{prune}} = T^2 + 2TV. \tag{14}$$

Then, during the decode stage, the FLOPs of each layer can be estimated by

$$FLOPs^{\text{decode}} = 4d^2 + 2nd + 3dm. \tag{15}$$

From the above analysis, the computational bottleneck of VLM inference primarily arises in the prefill stage, where the self-attention operation incurs a quadratic complexity term $O(n^2d)$. Since the introduction of a large number of vision tokens substantially increases the sequence length n, the prefill stage becomes significantly more expensive than the decode stage, whose complexity scales only linearly with n. Therefore, to effectively reduce overall inference cost, vision token pruning should be performed during the prefill stage rather than deferred to the decode stage.

[1] https://github.com/deepcharles/ruptures

D Compatibility with FlashAttention

Many modern VLMs rely on FlashAttention (Dao et al., 2022) for efficient prefill and therefore do not explicitly materialize attention matrices. Since AdaptInfer uses attention for token ranking, a natural concern is whether this would introduce substantial overhead and offset the inference gains brought by pruning. Our answer is negative. AdaptInfer does *not* require attention extraction from all transformer layers, nor does it require the full attention matrix at any queried layer. In practice, attention is queried only at the selected pruning layers (at most three in our experiments), and only the *text-to-text* ($t2t$) and *text-to-vision* ($t2v$) blocks are needed. The expensive *vision-to-vision* ($v2v$) block is never used.

We build on the dual-pass FlashAttention strategy introduced by SparseVLM (Zhang et al., 2025b), but adapt it to our dynamic text-guidance setting. The first pass is the standard FlashAttention forward pass, which produces the hidden states used by the model. At a queried pruning layer, we then perform an auxiliary pass that reuses the same queries and keys, but replaces the value matrix with task-specific selector matrices so that only the required attention sub-blocks are returned. This preserves FlashAttention compatibility while avoiding explicit materialization of the full attention map.

Required attention blocks. Let the input sequence at a queried layer contain T text tokens and V vision tokens, with total length $L = T + V$. For head h, let

$$A^{(h)} = \mathrm{Softmax}\left(\frac{Q^{(h)}K^{(h)\top}}{\sqrt{d}}\right) \in \mathbb{R}^{L\times L} \quad (16)$$

denote the self-attention matrix, and let

$$\bar{A} = \frac{1}{H}\sum_{h=1}^{H} A^{(h)} \quad (17)$$

be the attention matrix averaged over heads. Since the FlashAttention-compatible dual-pass returns mean attention statistics, AdaptInfer operates on $\bar{A}$ in this appendix.

Let $S_q \in \{0,1\}^{T\times L}$ select the rows corresponding to text queries, let $S_t \in \{0,1\}^{L\times T}$ select text-key columns, and let $S_v \in \{0,1\}^{L\times V}$ select vision-key columns. Then the only two attention blocks required by AdaptInfer are

$$\bar{A}_{t2t} = S_q\bar{A}S_t \in \mathbb{R}^{T\times T}, \quad (18)$$

$$\bar{A}_{t2v} = S_q\bar{A}S_v \in \mathbb{R}^{T\times V} \quad (19)$$

The dynamic text prior is computed as

$$w = \mathbf{1}^\top \bar{A}_{t2t} \in \mathbb{R}^T, \quad (20)$$

and the visual-token importance scores are then

$$s = w^\top \bar{A}_{t2v} \in \mathbb{R}^V. \quad (21)$$

Equivalently,

$$s = \left(\sum_{i=1}^{T} \bar{A}_{t2t}[i,:]\right)^\top \bar{A}_{t2v}. \quad (22)$$

This is the mean-attention version of Eq. 2 and Eq. 3 in the main paper.

Dual-pass extraction of the required blocks. In the auxiliary pass, we keep the same queries and keys as in the original layer, but replace the value matrix by selector matrices. Specifically, if we set the value matrix to

$$V_t = S_t, \quad (23)$$

then the FlashAttention output restricted to text-query rows becomes

$$O_t = S_q\bar{A}V_t = S_q\bar{A}S_t = \bar{A}_{t2t}. \quad (24)$$

Likewise, if we set the value matrix to

$$V_v = S_v, \quad (25)$$

the output restricted to text-query rows is

$$O_v = S_q\bar{A}V_v = S_q\bar{A}S_v = \bar{A}_{t2v}. \quad (26)$$

Therefore, the two auxiliary queries recover exactly the two sub-blocks required by AdaptInfer, without constructing the full attention matrix. In other words, our modification of the SparseVLM dual-pass is to query *block-restricted mean attention* tailored to dynamic text guidance, rather than extracting generic attention statistics for all tokens.

Complexity. A full attention matrix at one layer has size

$$L^2 = (T+V)^2 = T^2 + 2TV + V^2, \quad (27)$$

which includes the expensive $v2v$ block of size V^2. By contrast, AdaptInfer only needs the two blocks $\bar{A}_{t2t}$ and $\bar{A}_{t2v}$, whose total size is

$$T^2 + TV. \quad (28)$$

Hence, the auxiliary extraction cost per queried layer scales with the required block size

$$O(T^2 + TV), \tag{29}$$

rather than

$$O((T+V)^2) = O(T^2 + 2TV + V^2). \tag{30}$$

Since in VLM inference we typically have $T \ll V$, avoiding the V^2 term is crucial, especially for high-resolution images and video inputs. Moreover, this auxiliary extraction is performed only at a small number of pruning layers, instead of every transformer layer. If the number of queried pruning layers is M, the total auxiliary extraction cost is

$$O\big(M(T^2 + TV)\big), \tag{31}$$

with $M \leq 3$ in all our experiments.

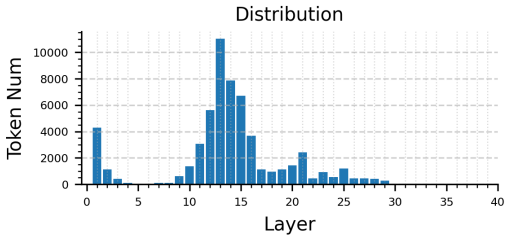
In summary, AdaptInfer remains compatible with FlashAttention by extending the dual-pass design of SparseVLM to the specific needs of dynamic text guidance. The method only queries the $t2t$ and $t2v$ blocks at a few selected layers and never uses the dominant $v2v$ block. As a result, the additional overhead is modest in practice and does not materially offset the efficiency gains from pruning.

E Experiments on LLaVA-1.5-13B

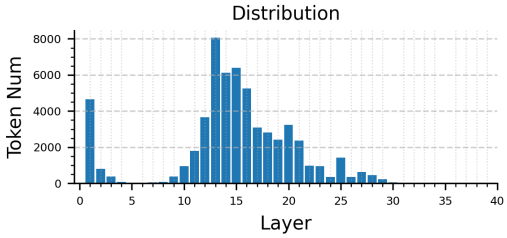
E.1 Attention Shift Detection on LLaVA-1.5-13B

To validate the transferability of our proposed pruning schedule, we conduct attention shift detection on LLaVA-1.5-13B (Liu et al., 2023), which is built upon a 40-layer LLaMA-2-13B backbone (Touvron et al., 2023b). The resulting distributions on two benchmark datasets, MME and TextVQA, are presented in Figure 5. We observe that the distribution of attention shift points still remains highly consistent on MME and TextVQA. The early-stage concentration of attention shifts consistently occurs at layer 1. Meanwhile, in later layers, dense change points are slightly delayed compared to those in LLaVA-1.5-7B, which typically occurs around layers 10 to 22. This suggests that for LLaVA-1.5-13B and other VLMs built on the LLaMA-2-13B backbone, an appropriate pruning schedule should select representative layers such as 1, 11, and 22 accoring to our proposed schdule.

Moreover, our attention shift detection pipeline is both simple and lightweight. For example, on the MME dataset, extracting 1000 attention maps and performing change-point detection takes only about 8 minutes in total. This confirms the efficiency and transferability of our pruning schedule across different model scales and architectures.



(a) Attention shift distribution on MME.



(b) Attention shift distribution on TextVQA.

Figure 5: **Layer-wise distribution of attention shifts on LLava-1.5-13B.**

Tokens	MME	MMB	TextVQA	Ratio (%)
576	1826	61.2	68.5	100
128	1813	59.8	67.4	98.5
64	1735	58.1	66.2	95.5

Table 8: **Performance of AdaptInfer on LLava-1.5-13B.** Our method still maintain high overall accuracy.

E.2 Performance of AdaptInfer on LLava-1.5-13B

Using the parameters above, we test our AdaptInfer on LLava-1.5-13B to further validate the effectiveness of the proposed pruning schedule. The experiments are based on three widely used dataset MME (Fu et al., 2025), MMBench (MMB) (Liu et al., 2024c), and TextVQA (Singh et al., 2019) under 128 and 64 vision token budgets. The results are shown in Table 8.

Under the more demanding 13-billion-parameter setting of LLaVA-1.5, AdaptInfer continues to deliver high performance. As Table 8 shows, pruning the vision token numbers from 576 to 128 on average per layer results in only a 1.5% drop in overall accuracy (from 100% to 98.5%). Even with an aggressive 64 vision token budget, AdaptInfer achieves an 8.9× reduction in vision-token count,

the model still retains 95.5% of its full-token accuracy. These numbers demonstrate that our token-adaptive pruning solution scales gracefully with model size, confirming AdaptInfer's robustness.

F Attention Shift Detection on Qwen2-VL-2B

We also conduct our attention shift detection on a different VLM framework, Qwen2-VL-2B (Wang et al., 2024a), which contains 28 transformer layers in its LLM backbone. The attention shift distribution on MME is shown in Fig 6. The high-frequency regions are among layer 11-19, where stable regions are among layer 0-10 and 20-28. Based on our proposed pruning schedule, we select the token pruning locations at layer 0, 9, and 19. Moreover, conducting the attention shift analysis on Qwen2-VL- 2B takes only about 5 minutes in total, which illustrates the lightweightness and transferability of the proposed attention shift analysis.

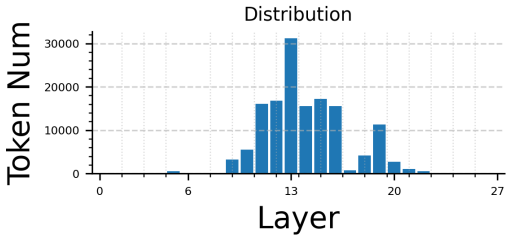


Figure 6: **Layer-wise distribution of attention shifts on Qwen2-VL-2B on MME.**

G Pruning Tolerance

To further understand the vision token pruning, we conduct a pruning-tolerance evaluation. Specifically, we apply AdaptInfer to Qwen2-VL-2B and report the performance across 14 multimodal tasks in the MME benchmark, comparing the scores with and without pruning. The results presented in Table 9 indicate that code reasoning, count and color-based tasks have a lower pruning tolerance relatively. Meanwhile, calculation and OCR tasks outperforms the original Qwen due to higher pruning tolerance and the noise reduction effect by token pruning.

H Instance-Level Decision Analysis

There is an inherent trade-off between efficiency and performance in vision token pruning. we admit that aggressive token pruning can induce instance-level decision changes. To examine this effect, we perform a simple instance-level analysis comparing the vanilla Qwen2-VL-2B and 30% token retained AdaptInfer on MME. For each example, we record whether the prediction is (correct/error) before and after pruning. We tabulate the transition, and report the change ratios in Table 10. We find that the vast majority of predictions remain unchanged, while the fraction of Correct to Error cases is small and comparable to the number of Error to Correct cases.

Table 9: **Pruning-tolerance evaluation on MME.** For AdaptInfer, 30% tokens are retained averagely.

Task	Vanilla	AdaptInfer	Ratio
Existence	200	200	100%
Count	145	135	93.1%
Position	158	158	100%
Color	190	180	94.7%
Posters	125	123	98.4%
Celebrity	138	136	98.6%
Scene	159	159	100%
Landmark	174	172	98.9%
Artwork	135	135	100%
OCR	88	103	117%
Commonsense	112	111	99.1%
Calculation	23	33	143%
Translation	163	155	95.1%
Code Reasoning	93	85	91.4%

Table 10: **Instance-Level change before and after pruning.** C: Correct; E: Error.

	C to C	C to E	E to C	E to E
Percentage	78.2%	1.7%	1.3%	18.8%

I Visualization

To qualitatively assess the pruning effect of the proposed AdaptInfer, we conduct an additional visualization study. In line with the main paper, vision tokens are pruned after the 1st, 10th, and 20th transformer layers. The resulting figures depict the spatial distribution of the remaining tokens at each stage: transparent regions correspond to tokens that survive pruning, whereas semi-transparent patches indicate positions that have been discarded.

The experiment is carried out on the TextVQA (Singh et al., 2019) benchmark, from which we sample a representative set of examples for display. As the layers deepen, ADAPTINFER rapidly suppresses tokens associated with semantically irrelevant background areas while consistently preserving those aligned with



Figure 7: **Visualization of AdaptInfer on different examples.**

question-critical cues. The visualization is shown in Figure 7.

J Performance Analysis

We provide further analysis of the advantages of our dynamic text guidance method, AdaptInfer, compared with the static text guidance solution, SparseVLM (Zhang et al., 2025b). We extract some VQA examples from the TextVQA dataset (Singh et al., 2019), and the answers of both methods are presented in Figure 8.

Within each question, blue highlights denote the static key text tokens selected offline by SparseVLM, whereas red highlights trace the key text tokens adaptively chosen from various layers by AdaptInfer. Close inspection reveals a consistent pattern: SparseVLM almost exclusively locks onto the text tokens that bear the strongest direct semantic link to the visual scene (e.g., *printer* or *players*), but often discards adjectival or adverbial modifiers that disambiguate entities (e.g., *pointing*, *two* or *behind*). When the prompt becomes syntactically richer and contains multiple content nouns and stacked modifiers static text guidance leads to an under-specified textual context and, consequently, a notable drop in answer accuracy. Conversely, AdaptInfer updates its key text token set dynamically during inference, rarely missing any

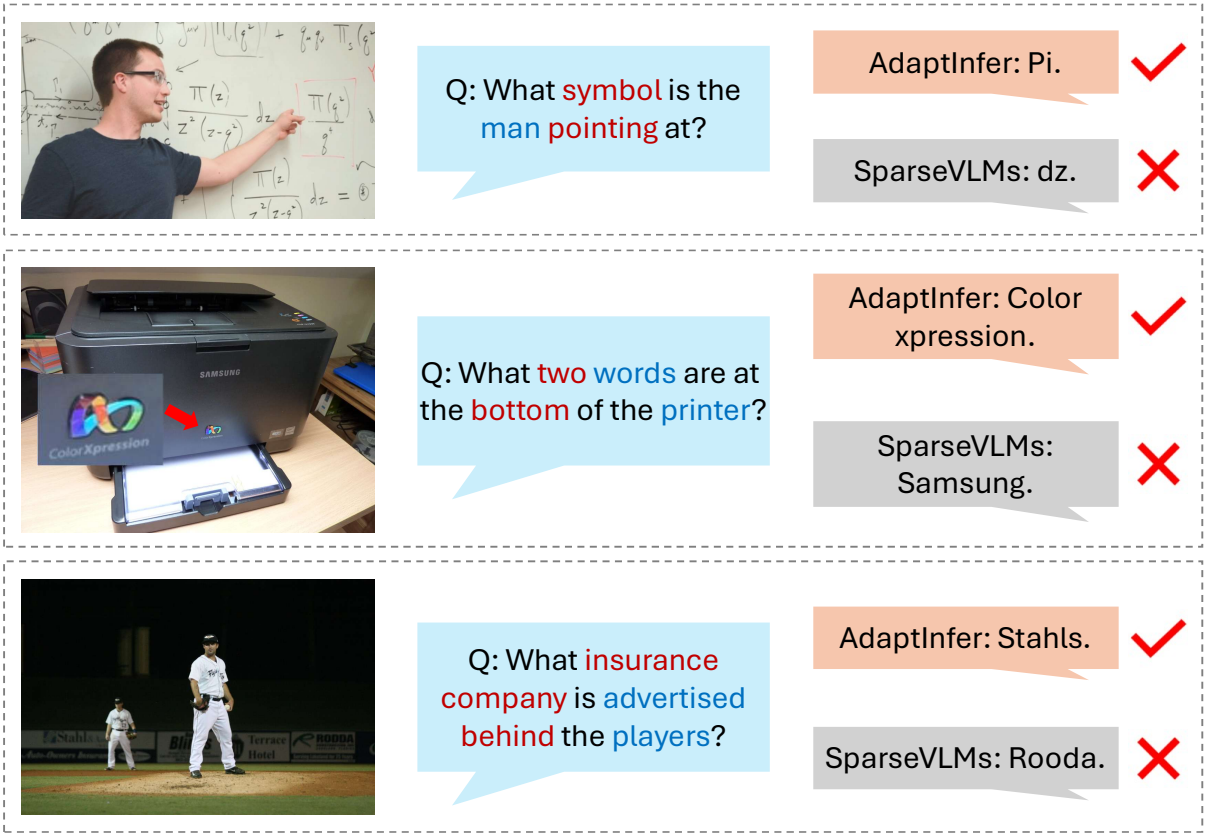


Figure 8: **Examples where AdaptInfer outperforms SparseVLM.** AdaptInfer performs better when text prompts are rich in descriptive modifiers.

prominent guidance. This dynamical adjustment make AdaptInfer overcome such limitation.

K Experiment Setup

In this appendix section, we will briefly introduce the details of our experiment setups.

K.1 Hardware Environment

To facilitate reproducibility, we ran every experiment on a single workstation that hosts an Intel Xeon Platinum 8358P processor together with eight NVIDIA RTX 4090 GPUs, each furnished with 24 GB of VRAM. The system operates under Ubuntu Linux 24.04, and all code was compiled using GCC 13.2.0 with Binutils 2.42 as the linker.

K.2 General Implementation Details

In our evaluation of AdaptInfer, the pruning ratios at each pruning location are the hyperparameters need to be selected as well. For these hyperparameters, we follow the same rank-based sparsification-level adaptation method which SparseVLM (Zhang et al., 2025b) introduced.

In the paper, we evaluated four vision–language models in total, including LLaVA-1.5-7B, LLaVA-1.5-13B, InternVL3-2B (Zhu et al., 2025) and Qwen2-VL-2B. To stay within GPU-memory limits, we ran the LLava-7B variant, Qwen2-VL-2B, and InternVL3-2B on one GPU, and the LLava-13B variant on three GPUs. All training and inference are performed in torch.float16 or torch.bfloat16 precision to further conserve memory usage. All the results of AdaptInfer reported in the context are the average of five runs.

K.3 Experiments on DART

DART (Wen et al., 2025) reports performance on LLaVA-v1.5-7B under different retained token counts in the paper. However, these values refer to the number of remaining vision tokens after pruning, whereas our work defines the token budget as the average number of vision tokens retained per layer. Accordingly, our token budget measures the mean token preservation across all transformer layers in the prefill stage rather than the final post-pruning token count.

To enable a fair and consistent comparison, we re-implemented DART using the official code-

base while keeping all default hyperparameters unchanged, including pruning location K is set to 2, $pivot_image_token$ is set to 4, and $pivot_text_token$ is set to 4. We only change the $max_num_trunction$ to 98 for a token budget of 128 and to 26 for a token budget of 64 respectively, ensuring that the resulting average retained token numbers match our budget definition.

K.4 Implementation Details of AdaptInfer on Qwen2-VL-2B

Qwen2-VL-2B employs a dynamic number of vision tokens, ranging from 4 to 16,384 per image. To ensure stable performance and prevent GPU memory overflow, we constrain the vision token count within a bounded range, setting the minimum to 256 and the maximum to 1,280. For video-based multimodal evaluation, we follow FastV (Chen et al., 2024a) and SparseVLM (Zhang et al., 2025b) and use the first 1,000 test samples from each benchmark to reduce evaluation time. The frame rate for all videos is fixed at 4 FPS.