# DAPRUNE: DUAL-ADAPTIVE PRUNING VIA ADAPTIVE RATIO AND LAYER SELECTION FOR LARGE VISION LANGUAGE MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Visual tokens in large vision language models (LVLMs) increase rapidly with image resolution and video inputs, resulting in substantial computational overhead. To reduce this cost, a pruning policy is needed to reduce the number of tokens involved in the attention computation. However, existing token pruning methods mostly adopt fixed pruning policies for all inputs, which are not sufficiently efficient and cannot adapt to input difficulty. To address this limitation, we propose DAPrune, a training-based dual-adaptive pruning framework that jointly learns *where* to prune and *how many* visual tokens to retain for each input. Specifically, DAPrune introduces two lightweight modules: a routing module for pruning layer selection and a ratio module for adaptive token retention. With the base LVLM frozen, these modules can adapt efficiently within 3 hours on a single H100 GPU for LLaVA-1.5. Empirically, DAPrune preserves over 98.5% of the baseline performance under high FLOPs reduction on LLaVA-1.5, LLaVA-NeXT, and Qwen-2.5-VL, consistently outperforming prior methods under comparable settings. Furthermore, theoretical analysis suggests that, under the same average budget, our adaptive policy achieves lower representation error than fixed pruning strategies.

## 1 INTRODUCTION

Large vision-language models (LVLMs) (Achiam et al., 2023; Liu et al., 2024b; 2023; 2024a) have achieved remarkable progress in multimodal understanding by combining visual encoders (Radford et al., 2021) with pretrained large language models. This design allows images to be represented as visual tokens and processed jointly with text, leading to strong performance across a wide range of vision-language tasks (Wang et al., 2023). However, it also introduces a fundamental efficiency bottleneck: visual inputs produce substantially more tokens than text, and the token count increases quadratically as image resolution increases (Hu et al., 2024a) or with video input (Lin et al., 2024). Because transformer cost scales rapidly with sequence length, these visual tokens incur significant memory and computation overhead in both training and inference, making efficient LVLM inference increasingly important for practical deployment.

To mitigate this issue, recent works have focused on pruning redundant visual tokens inside the language model, guided by a widely observed phenomenon: LVLMs tend to capture global visual information in shallow layers and focus on more localized regions in deeper layers. As a result, a dominant strategy is to rank visual tokens by an importance signal, typically derived from attention, and discard low-importance tokens at selected transformer layers so that subsequent layers operate on a shorter sequence. Specifically, FastV (Chen et al., 2024a) performs visual token pruning at an early layer, PDrop (Xing et al., 2024) extends this idea to a staged pruning scheme across decoder depth, and SparseVLM (Zhang et al., 2024) further introduces token scoring for visual token reduction. Despite their strong performance, they typically rely on manually fixed pruning configurations (policy), including pruning ratios and layers that are shared across all inputs. Although (Ye et al., 2025) further introduces learned pruning ratios, it requires fine-tuning from the Vicuna model (Chiang et al., 2023) and therefore incurs substantial additional training cost.

Particularly, sharing a fixed pruning policy across all inputs has two important drawbacks. First, it is difficult to identify the most efficient fixed policy. As shown in Figure 1, the pruning policy used in



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.

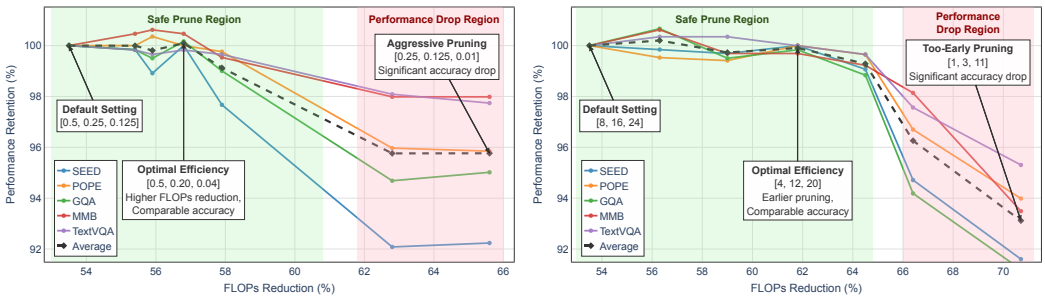


Figure 1: Performance-efficiency trade-off analysis for retain ratios (left) and pruning layers (right).

PDrop (Xing et al., 2024) is not optimal in terms of the FLOPs–performance trade-off, and a more aggressive fixed policy can achieve nearly the same performance with lower computational cost. The boundary between shallow layers and deep layers, and optimal pruning ratios cannot be determined in advance without an extensive grid search. Second, such a fixed policy ignores input heterogeneity, that inputs of different difficulty may require different pruning policies as shown in Figure 3. This naturally raises the following question: *can we design an efficient and fully adaptive pruning policy that automatically adjusts the pruning policy for each input without substantial training cost?*

In this work, we answer this question affirmatively by proposing *DAPrune*, a dual-adaptive pruning framework that jointly learns **where to prune** and **how many tokens to retain** conditioning on the input with marginal training cost. Concretely, DAPrune introduces two lightweight modules into the frozen LVLM decoders with only $0.53\%$ trainable parameters: routing modules and ratio modules. The routing module predicts pruning probabilities over $K$ candidate layers and selects the one with the highest probability. At the selected layer, visual tokens are first ranked by importance, such as using self-attention scores, and the ratio module then predicts the keep ratio to prune the low-ranked tokens. In this way, DAPrune adaptively prunes earlier and more aggressively for simpler inputs while preserving more tokens for harder ones. Moreover, even under the same average budget, our theoretical analysis suggests that adaptive pruning achieves lower representation error than fixed pruning. We summarize our contributions as follows:

- We reveal the importance of *dual-adaptive* visual token pruning in LVLMs, and formulate token pruning as a policy that should jointly determine both the pruning layer and the retention ratio.
- We propose *DAPrune*, a lightweight dual-adaptive pruning framework equipped with routing modules and ratio modules. Practical training objectives are developed, including balancing early pruning against difficulty-aware layer selection and aggressive pruning against target-budget control. On 10% sampled data, the modules can adapt within 3 hours on a single H100 GPU.
- Empirically, DAPrune achieves a strong and robust efficiency-performance trade-off, preserving 97.6%-98.7% of the baseline performance while reducing FLOPs by 73.1%-64.9%. Further analysis on training efficiency, layer selection frequency, and component ablation provides additional support for the effectiveness of DAPrune.
- We provide theoretical analysis suggesting that, under the same average budget, both adaptive ratio selection and adaptive layer selection reduce expected representation error relative to fixed policies. These results provide theoretical justification for the advantages of dual-adaptive pruning.

## 2 Related Work

### 2.1 Large Vision-Language Models

Recent advances in large language models, such as GPT (Achiam et al., 2023), Gemini, LLaMA (Touvron et al., 2023), Vicuna (Zheng et al., 2023), have driven the rapid development of large vision-language models (LVLMs) (Chen et al., 2023), which extend text-only reasoning to multimodal understanding by coupling a visual encoder (Radford et al., 2021) with a pretrained language model. Representative models such as LLaVA (Liu et al., 2024b;c; 2023), InstructBLIP (Dai et al., 2023),

MiniGPT-4 (Zhu et al., 2023), Qwen-VL (Bai et al., 2025a; Wang et al., 2024), and InternVL (Gao et al., 2024; Chen et al., 2024b) map images into visual tokens and process them jointly with text for multimodal instruction following and reasoning. As LVLMs move toward higher-resolution and multi-image settings (Bai et al., 2025a; Liu et al., 2024c), the number of visual tokens grows substantially. For example, LLaVA-1.5 uses 576 visual tokens for a 336×336 image, while higher-resolution variants such as LLaVA-NeXT (Liu et al., 2024c) produce far more. Because attention cost scales quadratically with sequence length, this rapid growth in visual tokens creates a major bottleneck in training and inference efficiency and occupies a large fraction of the model's context budget. This challenge has become increasingly important for practical deployment, especially in compute-sensitive and real-time settings (Ruan et al., 2025), motivating a growing line of work on improving the efficiency of visual token processing in LVLMs.

### 2.2 Vision Token Compression and Pruning

To improve the efficiency of large vision-language models, a growing body of work has focused on compressing or pruning visual tokens. Prior methods differ both in where token reduction is applied and in how token importance is determined. Pre-LLM approaches reduce visual representations before they are passed to the language model, for example, through learned query resampling, lightweight projectors, pooling, or encoder-side compression modules (Cha et al., 2024; Li et al., 2025; Hu et al., 2024b; Song et al., 2025; Zhang et al., 2025c). In contrast, in-LLM approaches exploit redundancy within the transformer layers of the language model itself, where token pruning or merging can be performed after multimodal interaction has begun. Within this setting, FastV (Chen et al., 2024a) prunes tokens at shallow LLM layers based on average attention received by each visual token, while PDrop (Xing et al., 2024) introduces a hierarchical pruning strategy motivated by progressively increasing redundancy across layers. SparseVLM (Zhang et al., 2024) further incorporates text-aware guidance through text-to-vision attention and adjusts pruning strength adaptively using attention structure. Other works explore alternative criteria beyond standard attention, such as token merging to preserve potentially useful local information (Yang et al., 2025) or diversity-based selection (Alvar et al., 2025; Wen et al., 2025; Lin et al., 2025) to retain a broader set of visual semantics. Recent methods also begin to explore where pruning should occur, rather than only how many tokens to retain. For example, Wu et al. (2026); Chen et al.; Lin et al. (2025) accelerates inference by removing visual tokens once they become less necessary for subsequent decoding. More recent studies also consider training-based pruning (Ye et al., 2025; Shao et al., 2025), where additional modules are introduced to predict instance-specific pruning thresholds. Overall, these works demonstrate that substantial visual redundancy exists in LVLMs, but they still leave open the challenge of determining both effective pruning locations and appropriate pruning ratios in a unified and adaptive manner.

## 3 Formulation and Motivation

We consider a standard LVLM decoder backbone composed of $L$ transformer layers. At each layer $l \in [L]$, the input hidden states are a concatenation of visual and text tokens, $H^l \in \mathbb{R}^{T\times d}$, where $T$ is the total number of tokens, and $d$ is the hidden dimension. The decoder adopts standard causal multi-head self-attention. For attention head $h$, the self-attention logits at layer $l$ are computed as $A^{h,l}_{\text{logits}} = \frac{Q^{h,l}(K^{h,l})^\top}{\sqrt{d_h}}$, where $Q^{h,l}, K^{h,l} \in \mathbb{R}^{T\times d_h}$ are the query and key matrices, and $d_h$ is the per-head dimension. To enforce causal attention, a lower triangular mask $M \in \mathbb{R}^{T\times T}$ is added to the logits, yielding the attention weights $A^{h,l} = \mathrm{Softmax}(A^{h,l}_{\text{logits}} + M)$. The attention outputs are then combined to produce the updated token features, followed by a feed-forward network (FFN).

**Problem Formulation.** The goal of visual token pruning is to reduce computational cost while preserving model performance. Let $\theta$ denote the frozen parameters of the vision-language model and $\pi$ denote a pruning policy. We seek a policy $\pi$ that optimizes the task objective subject to a computational budget: $\min_\pi \mathcal{F}(\theta, \pi),\ \text{s.t. } \mathcal{C}(\pi) \le \mathcal{B}$, where $\mathcal{F}(\theta, \pi)$ is the vision language modeling objective under policy $\pi$ plus model parameters $\theta$, $\mathcal{C}(\pi)$ measures the computational cost, such as FLOPs, and $\mathcal{B}$ denotes the target computational budget. In practice, the computational cost is reported by the FLOPs of the visual token part. For instance, the FLOPs of the multi-head attention and the FFN modules can be written as $\text{FLOPs}(n) = 4nd^2 + 2n^2d + 3ndm$ at each transformer layer in the LLaVA model, where $n$ is the number of visual tokens, $d$ is the hidden state size, and $m$ is the intermediate size of the FFN. Meanwhile, we follow the architecture in Xing et al. (2024) to divide

the decoder layers into $S$ stages that the problem can be formulated as

$$\min_{\pi} \mathcal{F}(\theta, \pi), \text{ s.t. } \mathcal{C}(\pi) = \frac{\sum_{s=0}^{S-1}(l_{s+1} - l_s) \cdot \text{Flops}(n_s)}{L \cdot \text{Flops}(n_0)} \leq \mathcal{B}, \tag{1}$$

where $l_s$ denotes the pruning layer at $s$-th stage, and $n_s$ denotes the number of token retained after $s$-th pruning. Naturally, the $\mathcal{C}(\pi)$ jointly depends on where to prune and how many tokens to retain.

**Challenges.** Recent studies indicate that decoder layers in LVLMs exhibit differentiated roles in visual reasoning: earlier layers primarily encode global visual context, whereas deeper layers increasingly focus on localized, task-relevant regions. This suggests that a larger number of tokens should be preserved in the early layers, while more aggressive token pruning can be applied in later layers (Chen et al., 2024a; Xing et al., 2024; Ye et al., 2025). Existing approaches typically leverage this observation by selecting specific pruning layers and assigning fixed keep ratios. However, the transition between these regimes remains underexplored, and such design choices are often training-free and uniformly applied across all inputs, leading to two key limitations.

***Problem 1: Suboptimal trade-off.*** It is inherently difficult to identify the most efficient pruning policy, since the boundary between safe pruning and performance-degrading pruning is unknown a priori. As illustrated in Figure 1, varying the pruning layer indices and keep ratios in the PDrop policy (Xing et al., 2024) reveals two clearly distinguishable regimes: a safe pruning region where performance remains largely stable, and a performance drop region where accuracy degrades rapidly. This observation further suggests that the original policy is not Pareto-optimal in the FLOPs–performance trade-off, as a more aggressive fixed policy can achieve nearly the same performance while reducing FLOPs by more than 10%. Therefore, manually designed fixed pruning policies may be inherently suboptimal and leave significant efficiency gains unexplored.

***Problem 2: Ignoring input difficulty.*** Applying the same pruning policy to all inputs ignores the substantial variation in input difficulty across tasks. As illustrated in Figure 3, the two examples exhibit clearly different reasoning requirements: the giraffe question mainly relies on coarse object recognition, whereas the beer-brand question requires fine-grained localization and recognition of text regions. It can be seen that applying an identical pruning policy to both inputs may retain many unnecessary background tokens for simpler inputs, where only a small number of tokens are sufficient, while failing to preserve enough task-relevant tokens for more challenging inputs.

Therefore, *pruning policies should be adaptive in both layer indices and token keep ratios to achieve better efficiency and capture input difficulty.* To this end, we propose a training-based dual-adaptive pruning policy, DAPrune. By training only lightweight pruning modules with sampled data, DAPrune incurs marginal overhead while remaining strong and robust under a wide pruning-budget range.

# 4 METHOD

As illustrated in Figure 2, DAPrune adaptively determines both where token pruning should be applied and how many visual tokens to retain at each decoder stage. The dual-adaptive policy is parameterized with two lightweight learnable modules:

$$\pi = g(\phi_l, \phi_r) = \{(l_s, r_s)\}_{s=1}^{S},$$

where $\phi_l$ and $\phi_r$ are the parameters of the *router module* and *ratio module*, respectively. For each input, the router module selects the pruning layer $l_s$ within each stage, while the ratio module predicts a keep ratio $r_s \in (0, 1)$ such that $n_s = \lfloor r_s \cdot n_0 \rfloor$. We describe each module in detail below.

## 4.1 ADAPTIVE LAYER SELECTION

Router modules learn *where* token pruning should be applied by selecting the pruning layer within each decoder stage conditioned on the input. Given the hidden states at the beginning of stage $s$, where each stage contains $K$ candidate layers, the routing module first computes a pooled representation by mean-pooling over the sequence dimension, yielding $h_s \in \mathbb{R}^d$. A two-layer MLP then maps this representation to $K$-dimensional routing logits: $z_s = \text{MLP}_{\phi_l}(h_s) \in \mathbb{R}^K$.

**Training.** Since discrete layer selection is non-differentiable, we apply the Gumbel-softmax relaxation to obtain soft selection weights:

$$\tilde{z}_s = \text{Gumbel-Softmax}(z_s, \tau),$$

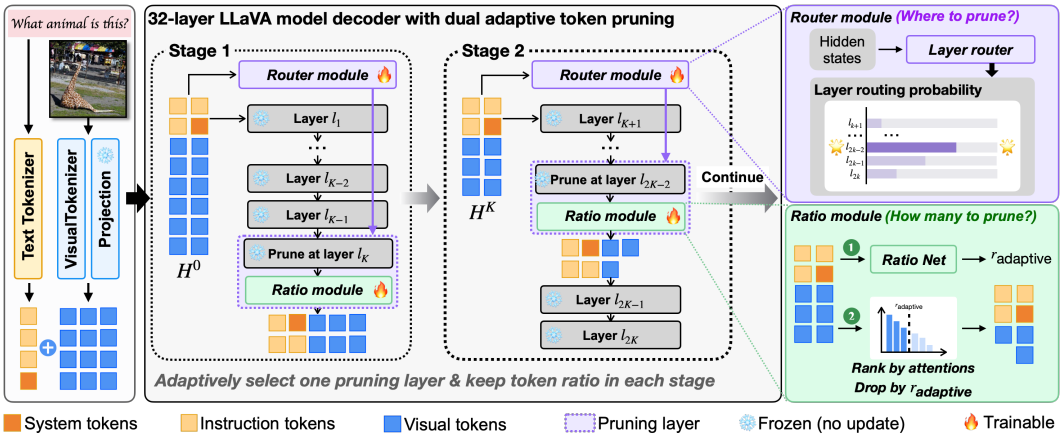


Figure 2: **Overview of DAPrune.** Our method introduces two lightweight modules into the model decoder (using 32-layer LLaVA as an example): a router module selects one pruning layer in each stage, and a ratio module predicts the adaptive token keep ratio at the selected layer. Based on ranked attention scores, DAPrune retains top-$r_{\text{adaptive}}$ tokens for subsequent computation.

where $\tau$ is a temperature parameter. The selected layer index is given by $\arg\max_j \tilde{z}_{s,j}$, while the soft weights $\tilde{z}_s$ are retained for gradient flow. Moreover, the task loss alone provides no direct supervision on which layer to select. We therefore introduce an auxiliary loss to encourage both layer diversity and a preference for earlier pruning among the $K$ candidates:

$$\mathcal{L}_{\text{router},s} = \frac{1}{K} \sum_{j=0}^{K-1} \left( \left| \tilde{z}_{s,j} - \frac{1}{K} \right| + \tilde{z}_{s,j} \cdot j \right).$$

This objective introduces a trade-off between early pruning and difficulty-aware layer selection. Without the first term, the router tends to collapse to the first layer in each stage. Without the second term, the router tends to utilize candidate layers more uniformly, regardless of inputs with different difficulty levels. By combining the two terms, we encourage the router to retain a preference for earlier pruning while still adapting the selected layer to the difficulty of the input.

**Inference.** At inference time, the pruning layer at stage $s$ is selected as $\arg\max_j z_{s,j}$.

### 4.2 Adaptive Token Retention

Ratio modules learn *how many* visual tokens should be retained at the selected pruning layer by predicting an adaptive keep ratio for each input, consisting of token ranking and ratio prediction.

At stage $s$, given the hidden states at the selected pruning layer, we compute an importance score for each visual token by averaging the self-attention weights over all heads, yielding a score vector $a_s \in \mathbb{R}^{n_{s-1}}$, where $n_{s-1}$ is the number of visual tokens before pruning at stage $s$. The visual tokens are then ranked in descending order according to the $a_s$. Further, using the pooled hidden states $h_{\text{selected}}$ at the current layer, the ratio module predicts the keep ratio and the number of retained tokens,

$$r_s = \sigma(\text{MLP}_{\phi_r}(h_{\text{selected}})) \in (0,1), \quad n_s = \lfloor r_s \cdot n_0 \rfloor$$

where $\sigma(\cdot)$ denotes the sigmoid function and $\lfloor \cdot \rfloor$ denotes the floor operator.

**Training.** Since $r_s$ is continuous, the ratio module can be optimized directly through backpropagation. Similarly, the task loss alone provides an insufficient signal to control the token budget. We therefore introduce the following two auxiliary losses on $r_s$.

$$\mathcal{L}_{\text{budget},s} = (r_s - r_b)^2, \quad \mathcal{L}_{\text{sparsity},s} = r_s,$$

where $r_b$ represents a budget ratio. The budget loss penalizes deviation from a budget ratio $r_b$, and the sparsity loss encourages more aggressive pruning. Without the sparsity loss, the ratio module tends to stay near the budget ratio $r_b$ and lacks incentive to further reduce the token count. Without




Figure 3: **Visualization comparison of PDrop and DAPrune.** DAPrune adaptively selects pruning layers and retains fewer, task-relevant visual tokens for each input.

the budget loss, the sparsity loss drives $r_s$ toward excessively small values, leading to over-pruning. By combining the two losses, we encourage the model to prune aggressively while remaining close to the budget. Besides, we use MSE for the budget loss rather than $L_1$, since $\partial L_1/\partial r_s$ yields a constant gradient regardless of the distance from $r_b$, which may lead to instability near the target.

**Inference.** Similarly, $r_s$ is predicted, and the top-$n_s$ tokens are retained at the selected layer.

### 4.3 Overall Training Objective

The overall training objective of DAPrune consists of the task loss, the router, and ratio regularization:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \frac{1}{S}\sum_{s=0}^{S-1}\Big(\lambda_r \mathcal{L}_{\text{router},s} + \lambda_b \mathcal{L}_{\text{budget},s} + \lambda_s \mathcal{L}_{\text{sparsity},s}\Big), \tag{2}$$

where $\mathcal{L}_{\text{task}}$ denotes the original vision-language modeling loss. $\lambda_r$, $\lambda_b$, and $\lambda_s$ control the relative contributions of the router, budget, and sparsity regularization terms, respectively.

**Discussion.** In the current design, DAPrune uses attention weights to rank visual tokens before applying adaptive token pruning. This choice is simple and effective, but the proposed framework is not restricted to attention-based ranking. Since the routing module and ratio module only parameterize the pruning policy, DAPrune can be naturally combined with other token ranking criteria, such as similarity-based pruning or diversity-aware pruning (Zhang et al., 2025b; Wen et al., 2025). Lastly, the additional computational overhead is negligible, as shown in Table 5.

## 5 Experiments and Theoretical Takeaways

### 5.1 Experiment Setup

**Models.** We evaluate DAPrune on three representative LVLM backbones, namely LLaVA-1.5-7B (Liu et al., 2024b), LLaVA-NeXT-7B (Liu et al., 2024c) and Qwen-2.5-VL-7B (Bai et al., 2025b). LLaVA-1.5 is a widely used open-source LVLM that projects image features from the CLIP vision encoder (Goh et al., 2021) into the language model through a multimodal projector, yielding 576 visual tokens per image. LLaVA-NeXT is its high-resolution extension, which supports up to 2880 visual tokens and provides stronger high-resolution modeling capability. Qwen-2.5-VL-7B incorporates dynamic resolution processing to handle images of varying sizes, ranging from 4 to 16,384 visual tokens. We freeze the original LVLM backbone for all models and only optimize the proposed adaptive pruning modules. For training data, we follow the standard visual instruction tuning setup of LLaVA and use the **sampled** 665K instruction-following samples (Liu et al., 2024b)

Table 1: Performance comparisons on LLaVA-1.5-7B with different pruning ratios on 8 benchmarks.

| Method | Avg. Token↓ | FLOPs Red.↑ | GQA | MMB | MME | POPE | SEED | SQA$^I$ | MMB$^{CN}$ | TextVQA | Avg.↑ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Upper Bound Model* | | | | | | | | | | | |
| LLaVA-1.5 (Liu et al., 2023) | 576 | 0 | 62.0 | 64.3 | 1510.7 | 85.8 | 66.1 | 71.6 | 58.3 | 58.2 | 100% |
| ***Fix** Layer Index and **Fix** Pruning Ratio* | | | | | | | | | | | |
| FastV (ECCV24) (Chen et al., 2024a) | 192 | 67.0 | 52.7 | 61.2 | 1312.4 | 64.8 | 62.0 | 65.4 | 56.0 | 52.5 | 89.2% |
| | 128 | 77.9 | 49.6 | 56.1 | 1187.9 | 59.6 | 58.6 | 59.7 | 52.2 | 50.6 | 83.0% |
| PDrop (CVPR25) (Xing et al., 2024) | 192 | 67.8 | 60.1 | 63.2 | 1463.9 | 82.3 | 62.2 | **69.2** | 57.0 | 56.1 | 96.6% |
| | 128 | 78.8 | 56.0 | 61.1 | 1332.1 | 82.3 | 57.9 | 68.4 | 53.2 | 55.1 | 92.3% |
| ***Fix** Layer Index* and **Adaptive** Pruning Ratio | | | | | | | | | | | |
| SparseVLM (ICML25) (Zhang et al., 2024) | 192 | 67.8 | 57.6 | 62.5 | 1382.8 | 83.6 | 64.2 | 67.2 | 57.9 | **57.8** | 96.1% |
| | 128 | 78.8 | 56.0 | 60.0 | 1296.7 | 80.5 | 63.6 | 65.5 | 58.0 | 56.7 | 93.5% |
| AutoPrune (NEURIPS25) (Wang et al., 2025) | 210[1] | 63.9 | **60.3** | 64.5 | 1461.7 | **84.8** | 64.3 | 68.5 | 57.9 | 57.2 | 98.0% |
| | 146[1] | 75.0 | 59.8 | 63.5 | 1456.1 | 84.4 | 64.2 | 68.2 | 57.9 | 56.7 | 97.3% |
| ***Adaptive** Layer Index and **Adaptive**Pruning Ratio* | | | | | | | | | | | |
| **DAPrune** | 204 | 64.9 | 60.1 | **64.9** | **1494.2** | 84.7 | **65.6** | **69.2** | **58.0** | 57.4 | **98.7%** |
| | 151 | 73.1 | 59.3 | 63.0 | 1488.5 | 84.4 | 63.5 | 69.1 | **58.0** | 57.3 | 97.6% |

while we use **sampled** RefCOCO, RefCOCO+, and RefCOCOg training splits (Kazemzadeh et al., 2014; Mao et al., 2016). In practice, we find that training on only $10\%$ of the data is already sufficient to learn an effective adaptive pruning policy. This design further reduces the training cost while maintaining strong pruning performance.

**Benchmarks.** We evaluate our method on a diverse set of widely used visual understanding benchmarks for LVLM efficiency studies, including GQA (Hudson & Manning, 2019), MMBench (Liu et al., 2024d), MMBench-CN (Liu et al., 2024d), MME (Fu et al., 2023), POPE (Li et al., 2023b), SEED (Li et al., 2023a), ScienceQA (image-only setting) (Lu et al., 2022), TextVQA (Singh et al., 2019), RefCOCO, RefCOCO+, and RefCOCOg (Kazemzadeh et al., 2014; Mao et al., 2016). Together, these benchmarks cover a broad range of multimodal reasoning skills, such as object recognition, visual grounding, hallucination robustness, general visual question answering, and scientific reasoning. Following standard practice in prior work, we assess the effect of token pruning by comparing model performance on these benchmarks before and after pruning. Unless otherwise specified, we adopt the same evaluation protocol used in recent vision-token pruning studies. Implementation details and additional results are provided in Section A.

### 5.2 Main Empirical Results

**Comparison with State-of-the-Art Methods.** Table 1 compares DAPrune with recent visual token pruning methods on LLaVA-1.5 under representative pruning budgets and highlights the top-3 performances in red, yellow and green. DAPrune consistently delivers the best efficiency–performance trade-off across eight benchmarks. At 204 average tokens, it retains $98.7\%$ of the baseline performance with $64.9\%$ FLOPs reduction, outperforming PDrop ($96.6\%$), SparseVLM ($96.1\%$), and AutoPrune ($98.0\%$). When pruning more aggressively, DAPrune still retains $97.6\%$ of the baseline performance at only 151 tokens and $73.1\%$ FLOPs reduction, exceeding all prior methods under similar settings. This demonstrates the advantage of jointly adapting both pruning layers and token retention levels. Moreover, table 2 shows that DAPrune also generalizes well to LLaVA-NeXT, achieving 98.5% average retention, outperforming FastV (97.0%), PDrop (95.9%), and AutoPrune (97.3%) at the same budget level. We further evaluate DAPrune on Qwen-2.5-VL-7B in Table 3. At 50% token retention, DAPrune achieves the highest average retention of $96.3\%$, slightly outperforming VScan ($96.1\%$), and obtains the best results on five of the eight RefCOCO-family evaluation splits. Under 75% token retention, it shows the same competitive trend.

**Training Overhead.** Table 5 shows that DAPrune introduces only lightweight training overhead. With the LLaVA-1.5-7B backbone frozen, DAPrune trains 37.8M additional parameters from the router and ratio modules, accounting for only 0.53% of the backbone under the three-pruning-stage setting. These MLP modules add negligible computation, with 0.0025% FLOPs overhead, and training takes around 3 hours on a single H100 GPU.

[1]For AutoPrune, Avg. Token is computed from the average number of visual tokens used in attention computation, whereas the original paper reports the average number of tokens after pruning.

Table 2: Performances on LLaVA-NeXT-7B with the same pruning ratios on 7 benchmarks.

| Method | Avg. Token↓ | FLOPs Red.↑ | GQA | MMB | MME | POPE | SEED | SQA$^I$ | TextVQA | Avg.↑ |
|---|---|---|---|---|---|---|---|---|---|---|
| LLaVA-NeXT (Liu et al., 2024c) | 2880 | 0 | 64.2 | 68.7 | 1534.1 | 86.1 | 71.1 | 70.4 | 67.2 | 100% |
| FastV (Chen et al., 2024a) | 862 | 64.5 | 61.9 | 66.3 | 1496.4 | **86.3** | 67.2 | 69.0 | 64.3 | 97.0% |
| PDrop (Xing et al., 2024) | 862 | 63.5 | 60.5 | 66.1 | 1451.7 | 85.2 | 67.0 | 68.3 | 64.5 | 95.9% |
| AutoPrune (Wang et al., 2025) | 862 | 62.0 | 62.0 | 66.5 | **1508.4** | 85.1 | 67.8 | 69.5 | 64.8 | 97.3% |
| **DAPrune** | 862 | 62.4 | **62.5** | **68.3** | 1505.2 | 86.2 | **68.6** | **69.7** | **66.5** | **98.5%** |

Table 3: Performance comparisons on Qwen-2.5-VL-7B (Bai et al., 2025b) across 3 referring expression comprehension benchmarks: RefCOCO, RefCOCO+, and RefCOCOg.

| Method | RefCOCO | | | RefCOCO+ | | | RefCOCOg | | Average |
|---|---|---|---|---|---|---|---|---|---|
| | val | testA | testB | val | testA | testB | val | test | |
| *Upper Bound, 4∼16384 Tokens (100%)* | | | | | | | | | |
| Qwen-2.5-VL-7B (arXiv'25) (Bai et al., 2025b) | 89.45 | 92.56 | 85.16 | 83.50 | 89.02 | 79.15 | 86.76 | 87.24 | 100.0% |
| *Retain 75% Tokens in Average (↓ 25%)* | | | | | | | | | |
| FastV (ECCV'24) (Chen et al., 2024a) | 85.27 | 87.84 | 82.28 | 79.02 | 82.95 | 72.86 | 82.95 | 83.32 | 94.8% |
| PDrop (CVPR'25) (Xing et al., 2024) | 87.79 | 91.00 | 83.22 | 81.48 | 86.55 | 74.02 | 84.62 | 85.10 | 97.2% |
| VScan (TMLR'26) (Zhang et al., 2025a) | 88.75 | **91.94** | 83.96 | **82.39** | **87.90** | 74.15 | 85.54 | 86.55 | 98.3% |
| **DAPrune (ours)** | **88.82** | 91.74 | **84.07** | 82.30 | 87.84 | **74.20** | **85.66** | **86.71** | **98.4%** |
| *Retain 50% Tokens in Average (↓ 50%)* | | | | | | | | | |
| FastV (ECCV'24) (Chen et al., 2024a) | 73.85 | 73.38 | 74.21 | 66.75 | 68.88 | 62.65 | 71.06 | 71.86 | 81.2% |
| PDrop (CVPR'25) (Xing et al., 2024) | 77.52 | 80.82 | 72.07 | 70.27 | 75.48 | 63.33 | 74.86 | 75.65 | 85.1% |
| VScan (TMLR'26) (Zhang et al., 2025a) | 86.78 | 90.74 | 82.37 | 79.99 | 86.12 | **71.67** | **84.03** | **84.44** | 96.1% |
| **DAPrune (ours)** | **87.08** | **90.91** | **82.50** | **80.07** | **86.40** | 71.60 | 84.00 | 84.27 | **96.3%** |

**Visualization Comparison.** Figure 3 shows that DAPrune prunes more selectively than PDrop by removing more unnecessary visual tokens while adapting its pruning behavior to input difficulty. From a spatial-retention perspective, DAPrune learns retention ratios and concentrates the tokens on question-relevant objects, such as the giraffe body or beer-label regions. Quantitatively, DAPrune can retain 20%-86% fewer tokens at each pruning stage. From an input-adaptivity perspective, DAPrune applies a more aggressive configuration to the easier giraffe example, pruning 7 layers earlier in total than PDrop and finally retaining only 10 tokens. For the beer-brand example, it uses a more conservative configuration, pruning only 4 layers earlier in total and retaining 33 final tokens.

### 5.3 Empirical Analysis and Ablation

**Component Ablation.** Table 4 studies the contribution of the two adaptive components in DAPrune. Compared with the fixed policy, using either adaptive ratio or adaptive layer selection alone reduces the average number of retained visual tokens, from 270 to 232 and 220, respectively, and improves FLOPs reduction. The adaptive-ratio-only variant learns to remove redundant tokens while preserving task-relevant regions. In contrast, the adaptive-layer-only variant prunes at earlier layers, which can remove useful visual information before token importance is sufficiently established. The DAPrune avoids this limitation by jointly deciding when to prune and how many tokens to retain, allowing earlier pruning while keeping an input-appropriate token keep count.

**Training data efficiency.** Table 6 reports the effect of using different fractions of data to train the adaptive pruning modules. While training time increases roughly linearly with the amount of data, the performance differences are relatively small once enough data is provided. In particular, we find that $5\%$ of the data is insufficient, but $10\%$ already offers a favorable trade-off between training cost and pruning performance. Further increasing the training data does not yield improvement.

**Budget Flexibility.** DAPrune provides flexible control over the computation budget through several design choices, including the sparsity coefficient $\lambda_s$, the target budget ratios $r$ at different stages, the number of pruning stages $S$, and the number of routing candidates $K$ per stage. By adjusting these factors, we can obtain pruning policies under different FLOPs budgets without manually specifying aggressive pruning schedules. As shown in Table 8-9, the resulting FLOPs reduction ranges from $60.7\%$ to $72.7\%$, while the performance varies from 95.2% to 98.1% with 3-stage pruning. Notably, Autoprune and DAPrune remain robust for over 73% FLOPs reduction with more pruning stages.

Table 4: Ablation study on the effect of adaptive ratio and adaptive layer selection.

| Variant | Adaptive Layer | Adaptive Ratio | FLOPs Red.↑ | Avg. Retention↑ | Avg. Tokens↓ |
|---|---|---|---|---|---|
| Fixed policy | ✗ | ✗ | 53.5 | 98.6% | 270 |
| Adaptive ratio only | ✗ | ✓ | 59.0 | 97.9% | 232 |
| Adaptive layer only | ✓ | ✗ | 61.4 | 97.0% | 220 |
| DAPrune | ✓ | ✓ | 64.9 | 98.7% | 204 |

Table 5: Training overhead analysis of DAPrune on LLaVA-1.5-7B.

| Component | Frozen Params | Trainable Params | Param Ovhd. (%) | Train Time Ovhd. | MLP Flops Ovhd. |
|---|---|---|---|---|---|
| Router module | 7,100,673,039 | 12,598,284 | 0.18% | – | 0.0008% |
| Ratio module | 7,100,673,039 | 25,171,971 | 0.35% | – | 0.0016% |
| **DAPrune (Total)** | 7,100,673,039 | **37,770,255** | **0.53%** | **3h** | **0.0025%** |

**Layer selection frequency.** Figure 4 shows the routing frequency of candidate layers in each stage when $K = 4$. Across all stages and benchmarks, the selected-layer frequency consistently decreases as the layer index increases. This indicates that the routing module generally prefers earlier layers within each stage, while still maintaining input-dependent variation across different tasks.

### 5.4 Theoretical Takeaways

We further provide a theoretical view of why adapting both the pruning ratio and pruning layer can benefit token pruning. **We summarize the main results here and defer the formal definitions, assumptions, and proofs to Section B.** We provide a theoretical explanation for why adapting both the pruning ratio and pruning layer can improve performance. When visual tokens are removed, the attention outputs are perturbed, resulting in a representation error $\mathcal{E}(\mathcal{R}, x)$. We show that this error is upper bounded by the corresponding pruning distortion $\mathcal{D}^l(k, x)$ as shown in Lemma 1:

$$\mathcal{E}(\mathcal{R}, x) \leq C \cdot \mathcal{D}^l(k, x), \tag{3}$$

where $C$ depends on the maximum norm of the value vectors. Therefore, we reduce the problem to comparing the pruning distortion induced by different pruning policies.

**Adaptive pruning ratio [Theorem 1].** Consider a fixed keep count $k_{\text{fix}}$ and an input-dependent keep count $k_{\text{adaptive}}(x)$ under the same average token budget, $\mathbb{E}_x[k_{\text{adaptive}}(x)] = k_{\text{fix}}$. We show that an adaptive allocation can achieve

$$\mathbb{E}_x[\mathcal{D}^l(k_{\text{adaptive}}(x), x)] \leq \mathbb{E}_x[\mathcal{D}^l(k_{\text{fix}}, x)]. \tag{4}$$

Intuitively, instead of assigning the same number of tokens to every input, the adaptive policy reallocates the budget by retaining fewer tokens for easier inputs and more for harder ones.

**Adaptive pruning layer [Theorem 2].** Similarly, let $l_{\text{fix}}$ be a fixed pruning layer and $l_{\text{adaptive}}(x)$ an input-dependent layer satisfying $\mathbb{E}_x[l_{\text{adaptive}}(x)] = l_{\text{fix}}$. Under the mild assumption that the marginal benefit of delaying pruning is non-negative and diminishes with depth, we show

$$\mathbb{E}_x[\mathcal{D}^{l_{\text{adaptive}}(x)}(k, x)] \leq \mathbb{E}_x[\mathcal{D}^{l_{\text{fix}}}(k, x)]. \tag{5}$$

The adaptive policy can therefore prune earlier for simpler inputs while delaying pruning for harder ones without increasing the average pruning depth.

## 6 Conclusion

In this paper, we propose *DAPrune*, a dual-adaptive visual token pruning framework for LVLMs that jointly learns where to prune and how many visual tokens to retain for each input. Empirically, DAPrune achieves a better efficiency–performance trade-off than existing methods with only marginal training overhead. Furthermore, our theoretical analysis suggests that, under the same average computational budget, adaptive pruning can achieve lower representation error than fixed pruning strategies. For future work, it would be interesting to extend the framework to broader multimodal reasoning settings and adaptive decoding strategies.

## AI USE STATEMENT

In this work, we used generative AI tools only for language editing and revision of author-written text, including improving grammar, clarity, and presentation. We did not use generative AI tools for developing research ideas or designing the methodology. All AI-assisted language revisions were carefully reviewed by the authors to ensure that they preserved the intended technical meaning and did not introduce unsupported claims or inaccuracies. We take full responsibility for the final content of this work, including any text revised with the aid of generative AI.

## REFERENCES

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

Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. Divprune: Diversity-based visual token pruning for large multimodal models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 9392–9401, 2025.

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

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

Junbum Cha, Wooyoung Kang, Jonghwan Mun, and Byungseok Roh. Honeybee: Locality-enhanced projector for multimodal llm. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13817–13827, 2024.

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 and Long Xing. Open-llava-next: An open-source implementation of llava-next series for facilitating the large multi-modal model community. *GitHub-xiaoachen98/Open-LLaVA-NeXT: Anopen-sourceimplementationfortrainingLLaVA-NeXT*, 2024.

Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, et al. Pali-x: On scaling up a multilingual vision and language model. *arXiv preprint arXiv:2305.18565*, 2023.

Xiaoyue Chen, Chen Gao, Yong Li, Xinlei Chen, et al. Balanced token pruning: Accelerating vision language models beyond local optimization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 24185–24198, 2024b.

Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6, 2023.

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

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, et al. Mme: A comprehensive evaluation benchmark for multimodal large language models. *arXiv preprint arXiv:2306.13394*, 2023.

Zhangwei Gao, Zhe Chen, Erfei Cui, Yiming Ren, Weiyun Wang, Jinguo Zhu, Hao Tian, Shenglong Ye, Junjun He, Xizhou Zhu, et al. Mini-internvl: a flexible-transfer pocket multi-modal model with 5% parameters and 90% performance. *Visual Intelligence*, 2(1):32, 2024.

Chris Hallacy Aditya Ramesh Gabriel Goh, Sandhini Agarwal Girish Sastry Amanda Askell, Pamela Mishkin Jack Clark Gretchen Krueger, Ilya Sutskever Alec Radford, and Jong Wook Kim. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2, 2021.

Anwen Hu, Haiyang Xu, Jiabo Ye, Ming Yan, Liang Zhang, Bo Zhang, Ji Zhang, Qin Jin, Fei Huang, and Jingren Zhou. mplug-docowl 1.5: Unified structure learning for ocr-free document understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 3096–3120, 2024a.

Wenbo Hu, Zi-Yi Dou, Liunian H Li, Amita Kamath, Nanyun Peng, and Kai-Wei Chang. Matryoshka query transformer for large vision-language models. *Advances in Neural Information Processing Systems*, 37:50168–50188, 2024b.

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

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

Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. Seed-bench: Benchmarking multimodal llms with generative comprehension. *arXiv preprint arXiv:2307.16125*, 2023a.

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

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

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

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

Hao Liu, Wilson Yan, Matei Zaharia, and Pieter Abbeel. World model on million-length video and language with blockwise ringattention. *arXiv preprint arXiv:2402.08268*, 2024a.

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

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

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

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

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

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 11–20, 2016.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Ciyu Ruan, Ruishan Guo, Zihang Gong, Jingao Xu, Wenhan Yang, and Xinlei Chen. Pre-mamba: A 4d state space model for ultra-high-frequent event camera deraining. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9169–9180, 2025.

Zhenwei Shao, Mingyang Wang, Zhou Yu, Wenwen Pan, Yan Yang, Tao Wei, Hongyuan Zhang, Ning Mao, Wei Chen, and Jun Yu. Growing a twig to accelerate large vision-language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 20064–20074, 2025.

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

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothee Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. 2023.

Hanshi Wang, Yuhao Xu, Zekun Xu, Jin Gao, Yufan Liu, Weiming Hu, Ke Wang, and Zhipeng Zhang. Autoprune: Each complexity deserves a pruning policy. *arXiv e-prints*, pp. arXiv–2509, 2025.

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

Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, et al. Image as a foreign language: Beit pretraining for vision and vision-language tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19175–19186, 2023.

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

Hao Wu, Yingqi Fan, Jinyang Dai, Junlong Tong, Yunpu Ma, and Xiaoyu Shen. Hidrop: Hierarchical vision token reduction in mllms via late injection, concave pyramid pruning, and early exit. *arXiv preprint arXiv:2602.23699*, 2026.

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

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

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

Ce Zhang, Kaixin Ma, Tianqing Fang, Wenhao Yu, Hongming Zhang, Zhisong Zhang, Haitao Mi, and Dong Yu. Vscan: Rethinking visual token reduction for efficient large vision-language models. *arXiv preprint arXiv:2505.22654*, 2025a.

Qizhe Zhang, Mengzhen Liu, Lichen Li, Ming Lu, Yuan Zhang, Junwen Pan, Qi She, and Shanghang Zhang. Beyond attention or similarity: Maximizing conditional diversity for token pruning in mllms. *arXiv preprint arXiv:2506.10967*, 2025b.

Shaolei Zhang, Qingkai Fang, Zhe Yang, and Yang Feng. Llava-mini: Efficient image and video large multimodal models with one vision token. *arXiv preprint arXiv:2501.03895*, 2025c.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P Xing, et al. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. *arXiv preprint arXiv:2309.11998*, 2023.

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

Table 6: Effect of training data fraction on DAPrune. We report benchmark performance, average tokens, FLOPs reduction, and average retention relative to the vanilla LLaVA-1.5-7B model.

| Data (%) | Avg. Token↓ | FLOPs Red.↑ | GQA | MMB | MME | POPE | SEED | $SQA^I$ | $MMB^{CN}$ | TextVQA | Avg.↑ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 5% | 204 | 64.9 | 59.4 | 62.9 | 1477.5 | 84.1 | 63.2 | 69.2 | 56.4 | 57.2 | 97.1% |
| 10% | 204 | 64.9 | 60.1 | 64.9 | 1494.2 | 84.7 | 65.6 | 69.2 | 58.0 | 57.4 | 98.7% |
| 20% | 204 | 65.0 | 59.3 | 63.2 | 1463.3 | 84.2 | 63.2 | 69.2 | 57.0 | 57.1 | 97.1% |
| 30% | 203 | 65.1 | 59.8 | 63.4 | 1480.3 | 84.4 | 63.3 | 69.2 | 57.6 | 57.2 | 97.6% |
| 100% | 206 | 64.4 | 59.5 | 64.0 | 1493.3 | 84.4 | 63.4 | 69.1 | 57.4 | 57.1 | 97.7% |

# A Experimental Details

## A.1 Implementation details

**Implementation Details.** All experiments are conducted using the HuggingFace Transformers library on NVIDIA A100 and H100 GPUs with FlashAttention enabled. Rather than training from scratch, DAPrune is built on top of pretrained LVLM checkpoints, LLaVA-1.5-7B and Open-LLaVANeXT (Chen & Xing, 2024), and Qwen-2.5-VL-7B, thereby significantly reducing training costs while preserving strong baseline performance.

For LLaVA training data, we follow the standard LLaVA visual instruction-tuning setup using the 665K instruction-following mixture, a CLIP ViT-L/14-336 visual encoder, and the pretrained multimodal projector. Notably, we find that training on only 10% of this data is sufficient to learn an effective adaptive pruning policy, further reducing overhead. For Qwen training, we use 10% sampled RefCOCO, RefCOCO+, and RefCOCOg training splits (Kazemzadeh et al., 2014; Mao et al., 2016).

To enable adaptive pruning, we introduce two lightweight modules, a router module and a ratio module, in each pruning stage into the frozen LLM backbone. For LLaVA training, these modules are trained for one epoch with the backbone fully frozen, using a learning rate of $2 \times 10^{-5}$, batch size 4, cosine learning rate decay, and BF16 precision. The router dynamically selects a pruning layer within each stage from three fixed candidate groups: $[\![5, 6, 7, 8]\!]$, $[\![13, 14, 15, 16]\!]$, and $[\![21, 22, 23, 24]\!]$, corresponding to early, middle, and late decoder stages, respectively. Training DAPrune on 10% of the data takes approximately 3 hours on a single H100 GPU, with only 0.53% additional trainable parameters relative to the frozen backbone. Hyperparameters are summarized in Table 7. All pruning operations are designed to be fully compatible with FlashAttention, ensuring that the theoretical FLOPs savings from adaptive token reduction translate into measurable wall-clock speedups during inference. We implement the Qwen2.5-VL-7B version on the VScan codebase (Zhang et al., 2025a), using a single pruning stage whose router selects among a window of candidate layers around VScan's own validated middle layer.

## A.2 Additional Information

To further verify that the router module learns data-dependent layer preferences rather than a fixed architectural bias, we compare its layer-selection distribution across models trained on data of different difficulty. Following the confidence-and-correctness procedure used to construct our evaluation splits, we label each TextVQA training sample using the vanilla (unpruned) LLaVA-1.5-7B baseline's confidence and correctness. Confidence is defined as the mean probability assigned to the generated tokens. An sample is labeled EASY if the baseline assigns it confidence within the top 25% of the training samples and answers it correctly, and HARD if its confidence falls within the bottom 25% and the baseline answers it incorrectly. We then train three models on 6,987-sample EASY-only, HARD-only, and combined 13,974-sample MIXED (EASY+HARD) subsets, respectively.

We visualize the resulting layer-selection distributions in Figure 5. The most pronounced difference appears at Stage 2, where the EASY-trained router overwhelmingly favors the earliest candidate layer, while the HARD-trained router distributes its selections much more evenly across the candidate layers. The MIXED-trained router exhibits an intermediate distribution between these two extremes. This pattern indicates that the router adapts its average layer preference to the difficulty statistics of the training data, rather than converging to a fixed selection behavior imposed by the architecture.

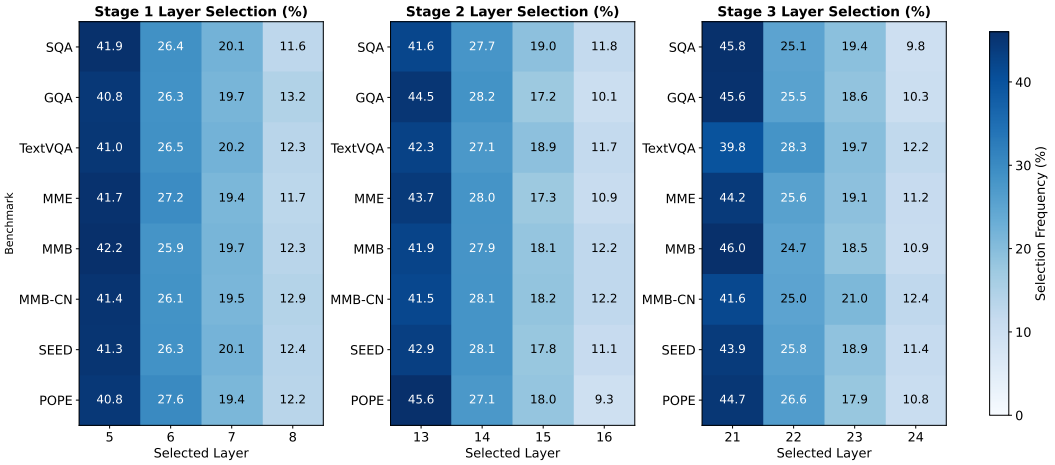


Figure 4: Benchmark-wise distribution of selected pruning layers across benchmarks and pruning stages. Each cell shows the percentage of samples for which the corresponding candidate layer is selected at that stage.

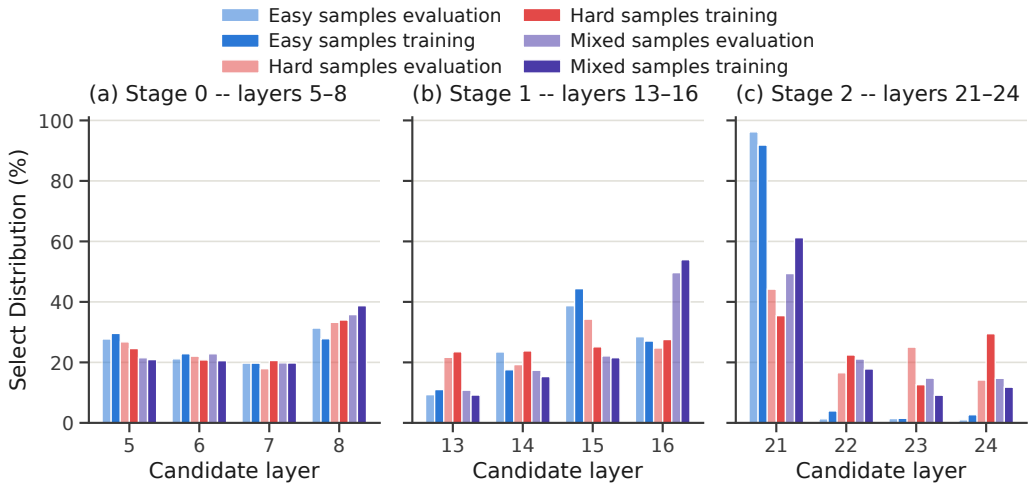


Figure 5: Layer-selection distributions across pruning stages for EASY-, HARD-, and MIXED-trained routers.

## B Theoretical Analysis

In this section, we analyze token pruning from a theoretical perspective. Our goal is to compare fixed and adaptive pruning policies along two dimensions: the number of kept tokens (pruning ratio) and the layer at which pruning is applied (pruning depth). Under the same average budget, we show that adaptive policies achieve lower representation error than fixed policies by allocating different computations according to the different inputs.

### B.1 Problem Setup

Let $f_\theta$ be a vision-language model processing a unified sequence of $T$ tokens:

$$\mathbf{x} = [\underbrace{t_1, \ldots, t_{n_{\text{pre}}}}_{\text{prefix text}}, \underbrace{v_1, \ldots, v_n}_{\text{visual tokens}}, \underbrace{t_{n_{\text{pre}}+n+1}, \ldots, t_T}_{\text{instruction + answer}}].$$

Table 7: Training configuration used for the DAPrune model built on top of LLaVA.

| Argument | Configuration |
|---|---|
| Architecture | LLaVA-based LVLM with lightweight ratio and router modules |
| Base model | LLaVA-v1.5-7B |
| Vision encoder | CLIP ViT-L/14-336 |
| Multimodal projector | Pretrained MLP-2×-GELU projector |
| Training data | 10% LLaVA instruction-tuning mixture (665K) |
| Image aspect ratio | Pad |
| Precision | BF16 with TF32 enabled |
| Number of epochs | 1 |
| Learning rate (both modules) | $2 \times 10^{-5}$ |
| Warmup ratio | 0.03 |
| LR scheduler | Cosine |
| Per-device batch size | 4 |
| Gradient accumulation steps | 1 |
| Effective batch size | 4 |
| Max sequence length | 2048 |
| Backbone frozen | Yes |
| Stage split layers | $[8, 16, 24]$ |
| Default token retention ratios | $[0.5, 0.25, 0.125]$ |
| Router candidate layers | $[[5, 6, 7, 8], [13, 14, 15, 16], [21, 22, 23, 24]]$ |
| Number of experts ($K$) | 4 |
| Router loss weight ($\lambda_{\text{r}}$) | 0.01 |
| Router temperature | 1.0 |
| Sparsity loss weight ($\lambda_{\text{s}}$) | 0.15 |
| Budget loss weight ($\lambda_{\text{b}}$) | 1.0 |

Table 8: Effect of sparsity loss weight $\lambda_s$ on DAPrune. A larger $\lambda_s$ encourages more aggressive pruning, trading performance for efficiency. All models use the same budget ratios $[0.5, 0.25, 0.125]$, default layers $[7, 15, 23]$, and $K = 4$ experts.

| $\lambda_{\text{s}}$ | Avg. Tokens ↓ | FLOPs Red.↑ | GQA | MMB | MME | POPE | SEED | SQA | MMB$^{CN}$ | TextVQA | Avg. Retention↑ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.05 | 226 | 60.7 | 59.9 | 62.5 | 1476.2 | 84.5 | 64.2 | 68.8 | 57.9 | 57.1 | 97.6% |
| 0.10 | 215 | 62.8 | 59.6 | 63.4 | 1500.1 | 84.8 | 64.2 | 69.3 | 57.9 | 57.3 | 98.1% |
| 0.2 | 189 | 67.2 | 58.8 | 62.5 | 1468.3 | 84.3 | 63.0 | 68.9 | 57.0 | 57.1 | 96.9% |

Then all tokens, including both text and visual, are embedded in the same $d$-dimensional space and processed together by self-attention layers. At each transformer layer $l$, the forward passes hidden states $H^l$ and outputs:

$$H^{l+1} = \text{Layer}^l(H^l),$$

where each layer $l$ consists of multi-head self-attention followed by an MLP with residual connections. In self-attention, the query, key, and value projections all come from the same hidden states $H^l \in \mathbb{R}^{T\times d}$:

$$Q^{h,l} = H^l W_Q^{h,l}, \quad K^{h,l} = H^l W_K^{h,l}, \quad V^{h,l} = H^l W_V^{h,l}.$$

Then the output for each token $t$ in head $h$ is a weighted sum over all token value vectors:

$$\text{out}_t^{h,l} = \sum_{j=1}^{T} A_{t,j}^{h,l} V_j^{h,l}, \quad A_{t,j}^{h,l} = \text{softmax}\Big(\frac{Q_t^{h,l}(K^{h,l})^\top}{\sqrt{d_h}} + M\Big)_j, \tag{6}$$

where $d_h = d/|\mathcal{H}|$ is the per-head dimension and $M$ represents the causal mask. Every token's output representation mixes information from all preceding tokens' value vectors. Then we have the following definitions.

Table 9: Effect of budget ratios on DAPrune. Lower retention ratios enforce more aggressive token reduction across stages. All models use the same layers $[7, 15, 23]$, $\lambda_s = 0.2$, and $K = 4$ experts.

| Budget Ratios | Avg. Tokens ↓ | FLOPs Red.↑ | GQA | MMB | MME | POPE | SEED | SQA | MMB$^{CN}$ | TextVQA | Avg. Retention↑ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $[0.5, 0.25, 0.125]$ | 189.0 | 67.2 | 58.8 | 62.5 | 1468.3 | 84.3 | 63.0 | 68.9 | 57.0 | 57.1 | 96.9% |
| $[0.5, 0.20, 0.10]$ | 180.0 | 69.0 | 58.4 | 62.0 | 1482.8 | 84.2 | 63.2 | 68.7 | 57.6 | 56.7 | 96.8% |
| $[0.4, 0.15, 0.05]$ | 158.0 | 72.7 | 57.3 | 61.8 | 1450.8 | 83.6 | 61.5 | 68.0 | 56.0 | 55.5 | 95.2% |

**Definition 1** (Token importance)**.** *We define the importance of the visual token $i$ at the layer $l$ as the average attention weight it receives from all query positions across all heads:*

$$w_i^l(x) = \frac{1}{T \cdot |\mathcal{H}|} \sum_{t=1}^{T} \sum_{h=1}^{\mathcal{H}} A_{t,n_{pre}+i}^{h,l}. \tag{7}$$

*This is the column-average of the full attention matrix over all heads, outputting a scalar in $[0, 1]$ for each visual token $i$. It captures how much the entire sequence collectively attends to visual token $i$. The more all tokens attend to it, the more harm its removal causes to all token representations.*

Token pruning typically follows a top-$k$ selection: given a non-negative keep count $k \in \mathbb{Z}$, retain the $k$ visual tokens with the highest importance scores (Chen et al., 2024a; Xing et al., 2024; Ye et al., 2025). The main difference across pruning policies lies in how $k$ is chosen. A fixed policy applies the same keep count to every input, while an adaptive policy predicts an input-dependent keep count. In either case, once $k$ is determined, the selected subset follows directly from the top-$k$ ranking. Based on this, we define the following pruning distortion to characterize the information loss.

**Definition 2** (Pruning distortion)**.** *Suppose we have the sorted importance for all visual tokens. Denote the set of retained visual tokens as $\mathcal{R}$ and the set of pruned visual token positions as $\bar{\mathcal{R}}$. The distortion of keeping $k$ tokens is:*

$$\mathcal{D}^l(k, x) = \sum_{i \notin \mathcal{R}} w_i^l(x) = \sum_{i=k+1}^{n} w_i^l(x). \tag{8}$$

*By construction, $\mathcal{D}^l(k, x)$ is non-increasing in $k$. Increasing the keep count from $k$ to $k + 1$ removes one fewer token from the sequence, and eliminates the contribution of the $(k + 1)$-th ranked token from the distortion:*

$$\mathcal{D}^l(k, x) - \mathcal{D}^l(k + 1, x) = w_{k+1}^l(x) \geq 0.$$

A key challenge in analyzing token pruning is the absence of a unified metric that directly captures its effect on model performance. Though the downstream metrics, such as accuracy or generation quality, are ultimately of interest, they are difficult to relate directly to the pruning operation. Naturally, pruning has an immediate and explicit effect on the attention update: deleting tokens removes their contributions from the attention aggregation and changes the resulting hidden representations. This motivates a representation-level analysis, and we define the following representation error.

**Definition 3** (Representation error)**.** *Let $\mathcal{R}$ denote the set of retained visual tokens, $out_t^{h,l}$ and $out_t^{h,l,\mathcal{R}}$ denote the output of token $t$ at layer $l$, head $h$, without and with pruning, respectively. The representation error induced by pruning is defined as the average perturbation over all tokens and heads:*

$$\mathcal{E}(\mathcal{R}, x) = \frac{1}{T \cdot |\mathcal{H}|} \sum_{t=1}^{T} \sum_{h=1}^{|\mathcal{H}|} \|out_t^{h,l} - out_t^{h,l,\mathcal{R}}\|. \tag{9}$$

*The representation error $\mathcal{E}(\mathcal{R}, x)$ quantifies the change in the self-attention outputs after removing a subset of tokens. The averaging over tokens and heads provides a global measure of the perturbation across the entire layer.*

Then we show that the representation error is bounded by the pruning distortion up to a constant factor.

**Lemma 1.** *Under the Definition 1-Definition 3, the following bound holds for any retained visual token set $\mathcal{R}$:*

$$\mathcal{E}(\mathcal{R}, x) = \frac{1}{T \cdot |\mathcal{H}|} \sum_{t=1}^{T} \sum_{h=1}^{|\mathcal{H}|} \|out_t^{h,l} - out_t^{h,l,\mathcal{R}}\| \leq C \cdot \mathcal{D}^l(k, x), \tag{10}$$

*where* $C := \max_j \|V_j^{h,l}\|$.

*Proof.* Recall that $\text{out}_t^{h,l} = \sum_{j=1}^{T} A_{t,j}^{h,l} V_j^{h,l}$, then $\text{out}_t^{h,l} - \text{out}_t^{h,l,\mathcal{R}} = \sum_{j \notin \mathcal{R}} A_{t,j}^{h,l} V_j^{h,l}$. Then we can obtain,

$$\|\text{out}_t^{h,l} - \text{out}_t^{h,l,\mathcal{R}}\| = \Big\| \sum_{j \notin \mathcal{R}} A_{t,j}^{h,l} V_j^{h,l} \Big\| \leq \sum_{j \notin \mathcal{R}} A_{t,j}^{h,l} \|V_j^{h,l}\|,$$

where the inequality follows from the triangle inequality and $A_{t,j}^{h,l} \geq 0$. At this point, $A_{t,j}^{h,l} \|V_j^{h,l}\|$ is a product of an attention weight and a value vector norm. On the one hand, tokens with small attention weights contribute little error. On the other hand, $\|V_j^{h,l}\|$ is bounded because LayerNorm is applied before the value projection at each layer, which normalizes the hidden states to have bounded magnitude. Therefore, there exists a constant $C = \max_j \|V_j^{h,l}\| \leq \infty$. Thus,

$$\|\text{out}_t^{h,l} - \text{out}_t^{h,l,\mathcal{R}}\| = \Big\| \sum_{j \notin \mathcal{R}} A_{t,j}^{h,l} V_j^{h,l} \Big\| \leq \sum_{j \notin \mathcal{R}} A_{t,j}^{h,l} \|V_j^{h,l}\| \leq C \cdot \sum_{j \notin \mathcal{R}} A_{t,j}^{h,l}. \tag{11}$$

Then we can obtain

$$\begin{aligned} \frac{1}{T \cdot |\mathcal{H}|} \sum_{t=1}^{T} \sum_{h=1}^{|\mathcal{H}|} \|\text{out}_t^{h,l} - \text{out}_t^{h,l,\mathcal{R}}\| \leq & \frac{C}{T \cdot |\mathcal{H}|} \sum_{t=1}^{T} \sum_{h=1}^{|\mathcal{H}|} \sum_{j \notin \mathcal{R}} A_{t,j}^{h,l} \\ = & C \sum_{j \notin \mathcal{R}} \Big( \frac{1}{T \cdot |\mathcal{H}|} \sum_{t=1}^{T} \sum_{h=1}^{|\mathcal{H}|} A_{t,j}^{h,l} \Big) = C \sum_{j \notin \mathcal{R}} w_j^l(x) \\ = & C \cdot \mathcal{D}^l(k, x). \end{aligned} \tag{12}$$

The proof is complete. □

Therefore, the objective becomes reducing the pruning distortion for different pruning policies.

We then make the following assumptions for comparing fixed and adaptive policies. For notation simplicity, we use $k_{\text{adaptive}}(x)$ and $l_{\text{adaptive}}(x)$ to denote the keep token count and the chosen pruning layer under adaptive policy, depending on the input $x$ while $k_{\text{fix}}$ and $l_{\text{fix}}$ are constants.

**Assumption 1.** *Both policies operate under the same average budget: $\mathbb{E}_x[k_{adaptive}(x)] = k_{fix}, \mathbb{E}_x[l_{adaptive}(x)] = l_{fix}$. The adaptive policy allocates fewer tokens and prunes earlier for simple inputs, while retaining more tokens and pruning later for hard inputs.*

**Assumption 2.** *The marginal benefit of delaying pruning by one layer, $\gamma^l(x) = \mathcal{D}^l(k, x) - \mathcal{D}^{l+1}(k, x)$, is non-negative and non-increasing with depth $l$.*

Intuitively, Assumption 1 ensures a fair comparison under the same average budget, while Assumption 2 captures the diminishing benefit of postponing pruning to deeper layers.

### B.2 Adaptive Ratio Analysis

For the adaptive policy, we assume that the keep count varies with input complexity: simple inputs are assigned a smaller keep count $k_{\text{adaptive}}$ (equivalently, a higher pruning ratio), whereas difficult inputs are assigned a larger keep count $k_{\text{adaptive}}$ (equivalently, a lower pruning ratio). By contrast, the fixed policy applies a uniform keep count $k_{\text{fix}}$ to all inputs. For a fair comparison, we require both policies to satisfy the same average budget on the keep count. Accordingly, we make the following assumption by solving the following objective.

**Objective function.** To formulate the trade-off between pruning quality and computational efficiency, we consider the following objective with $\lambda > 0$ being a threshold on marginal importance,

$$\Phi(k, x) = \mathcal{D}^l(k, x) + \lambda k. \tag{13}$$

The first term $\mathcal{D}^l(k, x)$ measures the distortion induced by pruning and, by Lemma 1 controls the representation error of the model. Minimizing this term encourages retaining tokens that are important for preserving the model's hidden states. The second term $\lambda k$ penalizes the number of retained tokens and serves as a proxy for computational cost, since the complexity of self-attention scales with the sequence length. Further, we define the oracle keep count $k^*(x) = \arg\min_k \mathcal{D}^l(k, x) + \lambda k$. The threshold on marginal importance $\lambda$ requires that $k^*(x) \in (1, n),\ \forall x$ and we have the following lemma.

**Lemma 2.** *Suppose that $\lambda$ is a suitable threshold on marginal importance that $k^*(x) \in (1, n),\ \forall x$. We can obtain the following relation:*

$$w_{k^*(x)+1} \leq \lambda \leq w_{k^*(x)}. \tag{14}$$

*Proof.* Recall that $k^*(x) = \arg\min_k \Phi(k, x)$ and the distortion is the summation of sorted token importance of pruned tokens $\mathcal{D}^l(k, x) = \sum_{i=k+1}^{n} w_i^l(x)$, where $w_{k+1}^l(x) \geq w_{k+2}^l(x) \geq \cdots \geq w_n^l(x)$. Then we have

$$\mathcal{D}^l(k^*(x)+1, x) + \lambda(k^*(x)+1) - \mathcal{D}^l(k^*(x), x) - \lambda k^*(x) = -w_{k^*(x)+1} + \lambda \geq 0.$$

Furthermore,

$$\mathcal{D}^l(k^*(x)-1, x) + \lambda(k^*(x)-1) - \mathcal{D}^l(k^*(x), x) - \lambda k^*(x) = w_{k^*(x)} - \lambda \geq 0.$$

Therefore, we can obtain $w_{k^*(x)+1} \leq \lambda \leq w_{k^*(x)}$. The proof is complete. □

In a word, the objective in eq. (13) keeps tokens until their importance drops below $\lambda$.

The adaptive policy should have the same average budget on the keep count as that in the fixed policy. Then we make the following assumption.

**Assumption 3.** *Let $k_{fix}$ denote the keep count used by the fixed pruning policy, and let $k_{adaptive}(x)$ denote the input-dependent keep count used by the adaptive pruning policy for input $x$. We assume that the two policies operate under the same average budget on the keep count,*

$$\mathbb{E}_x[k_{adaptive}(x)] = k_{fix}.$$

*Moreover, let $x_s$ denote a simple input and $x_h$ denote a hard input. We also assume that the adaptive policy allocates fewer kept tokens to simple inputs and more kept tokens to hard inputs,*

$$k^*(x_s) \leq k_{adaptive}(x_s) \leq k_{fix}, \qquad k^*(x_h) \geq k_{adaptive}(x_h) \geq k_{fix}.$$

Then we show that the distortion of the adaptive policy is smaller than that of the fixed policy in the following theorem.

**Theorem 1.** *Suppose that $\lambda$ is a suitable threshold on marginal importance that $k^*(x) \in (1, n),\ \forall x$ and Assumption 3 is satisfied. We have*

$$\mathbb{E}_x[\mathcal{D}^l(k_{adaptive}(x), x)] \leq \mathbb{E}_x[\mathcal{D}^l(k_{fix}, x)]. \tag{15}$$

*Proof.* For any simple input $x_s$, we have $k^*(x_s) \leq k_{\text{adaptive}}(x_s) \leq k_{\text{fix}}$ according to Assumption 3. Then we have,

$$\mathcal{D}^l(k_{\text{adaptive}}(x_s), x_s) - \mathcal{D}^l(k_{\text{fix}}, x_s) = \sum_{j=k_{\text{adaptive}}(x_s)+1}^{k_{\text{fix}}} w_j^l(x_s).$$

Recall that $w_j^l(x_s)$ is non-increasing with index $j$, and $w_{k^*(x_s)+1}^l \leq \lambda$ according to Lemma 2. Thus, we can derive

$$k^*(x_s) + 1 \leq k_{\text{adaptive}}(x_s) + 1,$$

$$w^l_{k_{\text{adaptive}}(x_s)+1} \leq w^l_{k^*(x_s)+1} \leq \lambda.$$

Then,

$$\mathcal{D}^l(k_{\text{adaptive}}(x_s), x_s) - \mathcal{D}^l(k_{\text{fix}}, x_s) = \sum_{j=k_{\text{adaptive}}(x_s)+1}^{k_{\text{fix}}} w^l_j(x_s) \leq (k_{\text{fix}} - k_{\text{adaptive}}(x_s))\lambda. \tag{16}$$

Similarly, with any hard input $x_h$, we have $k^*(x_h) \geq k_{\text{adaptive}}(x_h) \geq k_{\text{fix}}$ according to Assumption 3.

$$\mathcal{D}^l(k_{\text{adaptive}}(x_h), x_h) - \mathcal{D}^l(k_{\text{fix}}, x_h) = -\sum_{j=k_{\text{fix}}+1}^{k_{\text{adaptive}}(x_h)} w^l_j(x_h).$$

Since we have $k^*(x_h) \geq k_{\text{adaptive}}(x_h)$ and $w^l_{k^*(x_h)} \geq \lambda$, we can obatin that $w^l_{k_{\text{adaptive}}(x_h)} \geq w^l_{k^*(x_h)} \geq \lambda$. Then we can derive

$$\mathcal{D}^l(k_{\text{adaptive}}(x_h), x_h) - \mathcal{D}^l(k_{\text{fix}}, x_h) = -\sum_{j=k_{\text{fix}}+1}^{k_{\text{adaptive}}(x_h)} w^l_j(x_h) \leq -(k_{\text{adaptive}}(x_h) - k_{\text{fix}})\lambda. \tag{17}$$

Combining two cases and taking the expectation of the distortion gap over all inputs $x$, we have

$$\mathbb{E}_x[\mathcal{D}^l(k_{\text{adaptive}}(x), x)] - \mathbb{E}_x[\mathcal{D}^l(k_{\text{fix}}, x)] \leq \mathbb{E}_x[(k_{\text{fix}} - k_{\text{adaptive}}(x))\lambda] = 0, \tag{18}$$

where the last equality follows from $\mathbb{E}_x[k_{\text{adaptive}}(x)] = k_{\text{fix}}$. The proof is complete. □

### B.3 Adaptive Layer Analysis

Beyond **how many** tokens to prune, our method also learns **when** to prune, meaning which layer to apply pruning at. We first define the objective function below.

**Objective function.** To formulate the trade-off between pruning quality and computational efficiency along the depth dimension, we consider the following objective with $\mu > 0$ being a threshold on marginal benefit,

$$J(l, x) = \mathcal{D}^l(k, x) + \mu l. \tag{19}$$

The first term $\mathcal{D}^l(k, x)$ measures the distortion induced by pruning at layer $l$ and, by Lemma 1, controls the representation error of the model. Minimizing this term encourages delaying pruning to obtain more accurate token importance estimates. The second term $\mu l$ penalizes the depth at which pruning is applied and serves as a proxy for computational cost, since deeper layers incur additional computation before pruning. Further, we define the marginal benefit of delaying pruning by one layer as $\gamma^l(x) = \mathcal{D}^l(k, x) - \mathcal{D}^{l+1}(k, x)$. We then define the oracle pruning layer$l^*(x) = \arg\min_l \mathcal{D}^l(k, x) + \mu l$. The threshold on marginal benefit $\mu$ requires that $l^*(x) \in (1, L),\ \forall x$.

**Lemma 3.** *Suppose that $\mu$ is a suitable threshold on marginal benefit such that $l^*(x) \in (1, L),\ \forall x$. Then we have the following relation:*

$$\gamma^{l^*(x)}(x) \leq \mu \leq \gamma^{l^*(x)-1}(x). \tag{20}$$

*Proof.* Recall that $l^*(x) = \arg\min_l J(l, x) = \arg\min_l \big(\mathcal{D}^l(k, x) + \mu l\big)$, and the marginal benefit of delaying pruning from layer $l$ to layer $l+1$ is defined as $\gamma^l(x) = \mathcal{D}^l(k, x) - \mathcal{D}^{l+1}(k, x)$.

Since $l^*(x)$ minimizes $J(l, x)$, comparing layer $l^*(x)+1$ with layer $l^*(x)$ gives

$$\begin{aligned} J(l^*(x)+1, x) - J(l^*(x), x) &= \mathcal{D}^{l^*(x)+1}(k, x) + \mu(l^*(x)+1) - \mathcal{D}^{l^*(x)}(k, x) - \mu l^*(x) \\ &= -\gamma^{l^*(x)}(x) + \mu \geq 0. \end{aligned}$$

Furthermore, comparing layer $l^*(x)-1$ with layer $l^*(x)$ gives

$$\begin{aligned} J(l^*(x)-1, x) - J(l^*(x), x) &= \mathcal{D}^{l^*(x)-1}(k, x) + \mu(l^*(x)-1) - \mathcal{D}^{l^*(x)}(k, x) - \mu l^*(x) \\ &= \gamma^{l^*(x)-1}(x) - \mu \geq 0. \end{aligned}$$

Combining the two inequalities above, we obtain

$$\gamma^{l^*(x)}(x) \leq \mu \leq \gamma^{l^*(x)-1}(x). \tag{21}$$

The proof is complete. □

Similarly, we assume the same average budget on the layer indices.

**Assumption 4.** *Let $l_{fix}$ denote the pruning layer used by the fixed policy, and let $l_{adaptive}(x)$ denote the input-dependent pruning layer used by the adaptive policy for input $x$. We assume that the two policies operate under the same average budget on the layer indices,*

$$\mathbb{E}_x[l_{adaptive}(x)] = l_{fix}.$$

*Moreover, let $x_s$ denote a simple input and $x_h$ denote a hard input. We assume that the adaptive policy prunes earlier for simple inputs and later for hard inputs,*

$$l^*(x_s) \le l_{adaptive}(x_s) \le l_{fix}, \qquad l^*(x_h) \ge l_{adaptive}(x_h) \ge l_{fix}.$$

Moreover, the quality of importance scores $w_i^l(x)$ depends on layer depth, so we need to first characterize how importance scores evolve across layers.

**Empirical motivation.** We are motivated by empirical support from prior analyses of layer-wise attention in vision-language models. FastV (Chen et al., 2024a) and LLaVA-Mini (Zhang et al., 2025c) observe that attention over visual tokens is relatively uniform in early layers and becomes increasingly concentrated in deeper layers. As a result, with the same fixed keep count $k$, the top-$k$ tokens at deeper layers retain more of the total attention mass. In our notation, this corresponds to a decreasing pruning distortion with depth that $\mathcal{D}^l(k,x) \ge \mathcal{D}^{l+1}(k,x)$. In addition, we assume that the marginal benefit $\gamma^l(k,x)$ diminishes with depth. This captures a diminishing-return behavior: deeper layers can reserve original representation, but the improvement becomes smaller with depth. This property is important because it makes early pruning desirable whenever the additional reduction in distortion from going deeper is no longer significant. Therefore, we make the following assumption.

**Assumption 5.** *For any input $x$, the marginal benefit of delaying pruning by one layer satisfies*

$$\gamma^l(x) \ge 0, \qquad \gamma^l(x) \ge \gamma^{l+1}(x), \quad \forall l.$$

*That is, delaying pruning to a deeper layer does not increase distortion, and the marginal benefit of going one additional layer deeper is non-increasing with depth.*

Then we show that the distortion of the adaptive layer policy is smaller than that of the fixed layer policy in the following theorem.

**Theorem 2.** *Suppose that $\mu$ is a suitable threshold on marginal benefit such that $l^*(x) \in (1, L),\ \forall x$, and Assumption 4 and Assumption 5 hold. Then we have*

$$\mathbb{E}_x[\mathcal{D}^{l_{adaptive}(x)}(k,x)] \le \mathbb{E}_x[\mathcal{D}^{l_{fix}}(k,x)]. \tag{22}$$

*Proof.* For any simple input $x_s$, we have $l^*(x_s) \le l_{\text{adaptive}}(x_s) \le l_{\text{fix}}$ according to Assumption 4. Then

$$\mathcal{D}^{l_{\text{adaptive}}(x_s)}(k,x_s) - \mathcal{D}^{l_{\text{fix}}}(k,x_s) = \sum_{j=l_{\text{adaptive}}(x_s)}^{l_{\text{fix}}-1} \gamma^j(x_s).$$

Recall that $\gamma^j(x_s)$ is non-increasing with layer index $j$ by Assumption 5, and by Lemma 3, $\gamma^{l^*(x_s)}(x_s) \le \mu$. Since $l^*(x_s) \le l_{\text{adaptive}}(x_s)$, we have

$$\gamma^{l_{\text{adaptive}}(x_s)}(x_s) \le \gamma^{l^*(x_s)}(x_s) \le \mu.$$

Therefore,

$$\mathcal{D}^{l_{\text{adaptive}}(x_s)}(k,x_s) - \mathcal{D}^{l_{\text{fix}}}(k,x_s) = \sum_{j=l_{\text{adaptive}}(x_s)}^{l_{\text{fix}}-1} \gamma^j(x_s) \le (l_{\text{fix}} - l_{\text{adaptive}}(x_s))\mu. \tag{23}$$

Similarly, for any hard input $x_h$, we have $l^*(x_h) \ge l_{\text{adaptive}}(x_h) \ge l_{\text{fix}}$ according to Assumption 4. Then

$$\mathcal{D}^{l_{\text{adaptive}}(x_h)}(k,x_h) - \mathcal{D}^{l_{\text{fix}}}(k,x_h) = -\sum_{j=l_{\text{fix}}}^{l_{\text{adaptive}}(x_h)-1} \gamma^j(x_h).$$

Moreover, by Lemma 3, $\gamma^{l^*(x_h)-1}(x_h) \geq \mu$. Since $l_{\text{adaptive}}(x_h) - 1 \leq l^*(x_h) - 1$, and $\gamma^j(x_h)$ is non-increasing with $j$, we obtain

$$\gamma^{l_{\text{adaptive}}(x_h)-1}(x_h) \geq \gamma^{l^*(x_h)-1}(x_h) \geq \mu.$$

Hence,

$$\mathcal{D}^{l_{\text{adaptive}}(x_h)}(k, x_h) - \mathcal{D}^{l_{\text{fix}}}(k, x_h) = -\sum_{j=l_{\text{fix}}}^{l_{\text{adaptive}}(x_h)-1} \gamma^j(x_h) \leq -(l_{\text{adaptive}}(x_h) - l_{\text{fix}})\mu. \tag{24}$$

Combining the two cases and taking the expectation over $x$ yields

$$\mathbb{E}_x[\mathcal{D}^{l_{\text{adaptive}}(x)}(k, x)] - \mathbb{E}_x[\mathcal{D}^{l_{\text{fix}}}(k, x)] \leq \mathbb{E}_x[(l_{\text{fix}} - l_{\text{adaptive}}(x))\mu] = 0, \tag{25}$$

where the last equality follows from $\mathbb{E}_x[l_{\text{adaptive}}(x)] = l_{\text{fix}}$. The proof is complete. □