LANGUAGE-AWARE EFFICIENT ADAPTIVE FILTERING FOR VISUAL TOKEN SELECTION VIA TRUNCATED PROBING AND HIERARCHIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Large vision-language models (VLMs) process hundreds or thousands of visual tokens, many of which are redundant for a given question. Existing training-free reduction methods often apply uniform transformations or rank patches independently using a single importance signal, making it difficult to preserve both spatial coverage and task-relevant detail under aggressive compression. We introduce LEAF, a training-free framework that jointly determines *how* the image should be partitioned, *which* patches are relevant, and *how much detail* each region requires. LEAF combines a saliency-guided adaptive hierarchy with question-aware relevance extracted through truncated language-model probing, then allocates the token budget using a coverage–detail objective that balances regional coverage, patch relevance, and feature heterogeneity. Across four frozen VLMs and nine benchmarks, LEAF achieves the highest normalized average performance at all evaluated model–budget settings. On Qwen2.5-VL, it preserves 96.08% of full-token performance at 10% retention, compared with 90.76% for the strongest competing method, and achieves a BD-Token of -75.94% relative to random retention across the evaluated budget range, demonstrating effective visual-token reduction without training or backbone modification.

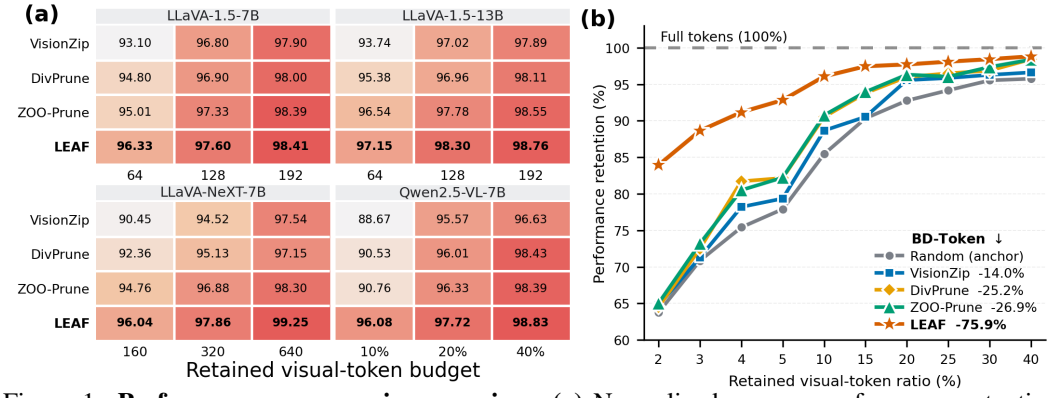

Figure 1: **Performance–compression overview.** **(a)** Normalized average performance retention across four VLMs at different visual-token budgets. Values are percentages of the corresponding full-token performance. **(b)** Qwen2.5-VL performance across visual-token retention ratios from 2% to 40%. The dashed line denotes full-token performance. BD-Token is computed relative to random retention, with more negative values indicating greater token savings at matched performance.

1 INTRODUCTION

Large vision-language models (VLMs) have demonstrated strong capabilities across a wide range of multimodal understanding and reasoning tasks, including visual question answering, image captioning, document understanding, and instruction following (Alayrac et al., 2022; Li et al., 2023a; Liu et al., 2023; Wang et al., 2024). These models typically regard an input image as a dense sequence of patch-level visual tokens, which are projected into the language-model embedding space and processed together with textual tokens. Preserving a large number of visual tokens enables fine-grained

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.

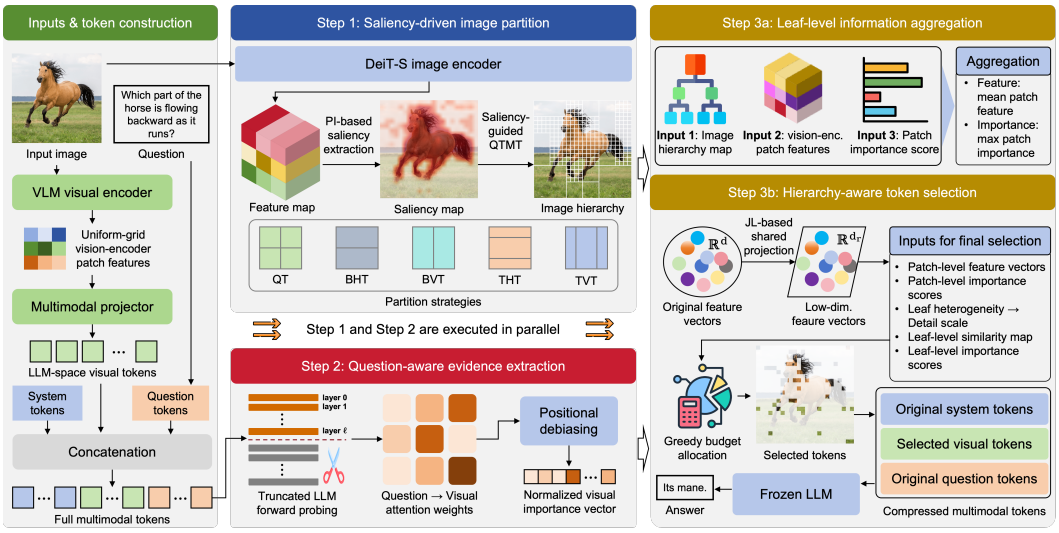

Figure 2: **LEAF** constructs a saliency-driven adaptive hierarchy and extracts question-aware patch importance through truncated language-model probing in parallel. It then performs hierarchy-aware visual-token selection. The selected original visual tokens are concatenated with the original system and question tokens and processed through a complete forward pass of the frozen language model.

perception, particularly for high-resolution images and visually complex scenes. However, it also greatly increases the sequence length processed by the language model, resulting in considerable computation, memory consumption, and inference latency (Zhang et al., 2026). This cost grows as VLMs adopt higher input resolutions and longer visual-token sequences.

Recent work has investigated visual-token reduction for efficient VLM inference. Existing approaches can be broadly categorized into transformation-based, similarity-based, attention-based, and query-based methods. Transformation-based approaches reduce the spatial token grid using operations such as pooling or convolution (Chu et al., 2023; Wang et al., 2024). However, these operations commonly apply a predetermined transformation at a uniform spatial granularity, preventing the token budget from adapting to the different levels of complexity across image regions. Similarity-based approaches merge visually redundant tokens into representative features (Bolya et al., 2023; Yang et al., 2025). Although merging preserves more information than direct removal, it modifies the original token representations and may collapse the spatial organization of visually similar but spatially distinct content. Attention-based approaches retain tokens that receive high attention from a visual summary token or the language model (Chen et al., 2024; Ye et al., 2025a). Such methods may require either a learned routing module or a substantial language-model forward pass before reliable attention scores become available, reducing the inference savings that token pruning is intended to provide. Query-based approaches explicitly use the textual query to identify task-relevant visual evidence (Zhang et al., 2025b). While more responsive to the current task, they generally score tokens over a flat patch grid and prioritize individually relevant patches without explicitly determining whether different semantic regions are adequately covered or whether a complex region requires additional local detail. These limitations motivate an approach that adapts spatial granularity to image content, preserves native visual representations, obtains language guidance efficiently, and balances semantic coverage with local detail.

To address these limitations, we introduce **LEAF** (Language-Aware Efficient Adaptive Filtering), a training-free framework for language-conditioned, hierarchy-aware visual-token selection (Fig. 2). LEAF comprises two parallel evidence-extraction branches followed by token selection. The image-driven branch uses a lightweight frozen image encoder to construct a saliency-guided quadtree-plus-multi-type-tree hierarchy over the native patch grid, producing content-adaptive leaves. In parallel, the question-driven branch executes only an early portion of the frozen language backbone and replays its native question-to-visual attention, followed by positional debiasing, to estimate patch-level relevance. LEAF then aggregates features and relevance within each leaf and greedily allocates the token budget according to regional importance, inter-leaf similarity, and within-leaf heterogeneity. The hierarchy is used only to organize budget allocation; the final sequence retains the selected original VLM visual tokens without modifying their representations.

Effective visual-token reduction requires three decisions: *how the image should be partitioned*, *which regions contain question-relevant evidence*, and *how much detail each region requires*. The adaptive hierarchy organizes the native patch grid into content-dependent regions, question-conditioned relevance identifies evidence needed for the current question, and the hierarchy-aware selector balances regional coverage with within-region detail. It therefore allocates fewer tokens to homogeneous regions and more to structurally complex ones, avoiding the uniform granularity of conventional downsampling. Truncated probing provides language guidance without a complete full-token forward pass devoted solely to importance estimation, while retaining original tokens avoids the representation shift introduced by merging or reprojection. These design choices improve performance under aggressive compression while maintaining competitive inference latency. As summarized in Fig. 1, LEAF achieves the strongest normalized performance across the evaluated frozen VLMs and token budgets, with larger gains at lower token budgets. LEAF also achieves the lowest BD-Token among the evaluated methods, indicating the lowest token requirement relative to random retention at matched performance.

Our main contributions are summarized as follows:

- We introduce a content-adaptive visual hierarchy that partitions the native patch grid at different spatial scales, enabling LEAF to retain more tokens in complex regions and fewer in homogeneous regions.
- We propose a training-free selection strategy that combines truncated question-conditioned relevance probing with hierarchy-aware allocation, jointly balancing task relevance, regional coverage, and within-region detail while retaining the original visual tokens.
- We evaluate LEAF across four frozen VLMs and nine benchmarks and adapt the BD-Rate metric to visual-token compression as BD-Token [1]. LEAF achieves the strongest normalized performance across the evaluated model–budget settings, with larger gains at lower token budgets.

2 Related Work

Large Vision-Language Models (VLMs) VLMs combine pretrained visual encoders with large language models for multimodal understanding and reasoning (Wu et al., 2023; Caffagni et al., 2024). Existing architectures connect modalities through gated cross-attention (Alayrac et al., 2022), query-based resampling (Li et al., 2023a), or lightweight visual projectors (Liu et al., 2023; 2024a), while models such as Qwen2-VL support dynamic-resolution inputs (Wang et al., 2024). Although dense visual tokens benefit fine-grained perception, processing them alongside text increases computational cost (Zhang et al., 2026). For example, LLaVA-1.5 represents an image with 576 tokens (Liu et al., 2024a), while dynamic-resolution models may produce longer sequences (Wu et al., 2024; Wang et al., 2024). Uniformly processing patches of varying complexity and query relevance therefore motivates visual-token compression that keeps task-relevant evidence with fewer tokens.

Visual Token Compression for VLMs Visual-token compression methods broadly follow transformation-, similarity-, attention-, or query-based strategies (Shao et al., 2025). Transformation-based methods reduce spatial grids through pooling, convolution, or learned projectors (Chu et al., 2023; Wang et al., 2024), but typically impose predetermined compression with uniform local granularity. Similarity-based methods, including ToMe, VisionZip, and FOLDER (Bolya et al., 2023; Yang et al., 2025; Wang et al., 2025), merge redundant features into representative tokens, modifying native representations and potentially weakening spatial organization. Attention-based methods prune tokens using importance estimated by the vision encoder or early LLM layers, as in FastV and ATP-LLaVA (Chen et al., 2024; Ye et al., 2025a), but may require substantial backbone computation or additional training. Query-based methods use language guidance to retain relevant tokens, as in SparseVLM (Zhang et al., 2025b), or distill visual information into specialized tokens, as in VoCo-LLaMA (Ye et al., 2025b). Despite improving efficiency, existing methods generally operate on flat patch sequences or replace native features, without jointly modeling semantic-region coverage and the detail required within each region.

Content-Adaptive and Hierarchical Visual Representation Content-adaptive representations address variations in regional information density: TokenLearner summarizes images using learned

[1] Appendix B.4 provides the formal definition and calculation procedure.

adaptive tokens (Ryoo et al., 2021), while A-ViT dynamically halts spatial-token processing according to image content (Yin et al., 2022). QuadTree Attention progressively refines regions receiving high coarse-level attention (Tang et al., 2022), whereas the quadtree-plus-multi-type-tree partition in Versatile Video Coding supports flexible spatial decomposition at multiple granularities (Bross et al., 2021). More closely related, QLIP replaces uniform CLIP patchification with a gradient-guided image quadtree and learned positional interpolation, improving MLLM perception without retraining the vision encoder or language model (Chickering et al., 2026). However, QLIP partitions images before visual encoding using image content alone and converts its leaves into new encoder inputs. In contrast, LEAF constructs a saliency-guided QTMT hierarchy over the VLM's native patch grid and combines it with language relevance, regional coverage, and within-region heterogeneity to allocate a budget while preserving the original visual representations without additional training.

3 METHOD

3.1 OVERVIEW OF LEAF

Given an image and question, a VLM produces N_v visual tokens. Under a budget $B < N_v$, LEAF selects B original tokens without training, merging, or modifying their representations. As shown in Fig. 2, LEAF comprises three stages. *S1* constructs a content-adaptive hierarchy through saliency-driven image partitioning, while the parallel *S2* estimates question-conditioned patch relevance through truncated language-model probing. *S3* combines both signals to allocate the budget according to regional coverage, question relevance, and within-leaf heterogeneity. The hierarchy guides only budget allocation; the selected tokens remain the original host-VLM visual tokens. Together with the unchanged textual tokens, they undergo a separate complete forward pass through the frozen language model. The two evidence branches can execute in parallel, after which the frozen language model performs ordinary inference on the selected visual sequence. Appendix A.1 summarizes the complete procedure.

3.2 SALIENCY-DRIVEN IMAGE PARTITION

As shown in Fig. 2, *S1* constructs a question-independent hierarchy whose granularity follows the image structure. A lightweight frozen DeiT-S encoder (Touvron et al., 2021) produces an auxiliary feature map. We subtract its spatial mean and estimate its leading principal direction using K power iterations (Xu et al., 2018). Center-prior orientation, percentile normalization, and resizing then produce a saliency map $S \in [0,1]^{G_h \times G_w}$, where G_h and G_w are the operational grid dimensions. This branch only estimates image structure.

LEAF recursively partitions $\mathbf{S}$ using saliency-guided quadtree plus multi-type tree (QTMT) partitioning (Bross et al., 2021). For a current region R, it computes

$$\bar{S}_R = \frac{1}{|R|} \sum_{u \in R} S_u, \qquad a_R = \frac{\max(\mathrm{Var}_{\mathrm{row}}(R), \mathrm{Var}_{\mathrm{col}}(R))}{\min(\mathrm{Var}_{\mathrm{row}}(R), \mathrm{Var}_{\mathrm{col}}(R)) + \epsilon}. \tag{1}$$

Here, $\bar{S}_R$ is the mean of saliency values S_u over the $|R|$ locations in R, and a_R is the anisotropy ratio of its row- and column-projection variances, with $\epsilon > 0$ ensuring numerical stability.

LEAF makes three successive partitioning decisions. First, if $\bar{S}_R \leq \alpha$, where α is the flatness threshold, R becomes a terminal leaf. Otherwise, R is split. Second, if $a_R \leq \rho$, where ρ is the anisotropy threshold, LEAF applies a quadtree (QT) split; if $a_R > \rho$, the dominant row or column variance determines a horizontal or vertical split. Third, for a directional split, κ_R measures the fraction of regional saliency within the central half along the split direction. LEAF uses a 1:2:1 ternary split when $\kappa_R > \tau$, where τ is the center-concentration threshold, and an equal binary split otherwise. These choices yield a binary horizontal tree (BHT), binary vertical tree (BVT), ternary horizontal tree (THT), or ternary vertical tree (TVT).

Recursion produces N non-overlapping leaves $\mathcal{L} = \{C_j\}_{j=1}^{N}$ covering all N_v host-VLM patches, where C_j contains the original patch indices assigned to leaf j. Full details appear in Appendix A.2.

3.3 QUESTION-AWARE EVIDENCE EXTRACTION

In parallel with the image-driven branch, LEAF obtains question-conditioned patch importance by running the frozen language model only up to a chosen probe layer ℓ, where it reuses the native

query and key projections to replay question-to-visual attention:

$$\mathbf{A}_{\mathrm{q}\to\mathrm{v}}^{(\ell,h)} = \left[\mathrm{softmax}\left(\frac{\widehat{\mathbf{Q}}^{(\ell,h)}[\mathcal{I}_{\mathrm{q}}]\widehat{\mathbf{K}}^{(\ell,h)\top}}{\sqrt{d_h}} + \mathbf{M}_{\mathrm{q}}\right)\right]_{:,\mathcal{I}_{\mathrm{v}}}. \tag{2}$$

Here, $\widehat{\mathbf{Q}}^{(\ell,h)}$ and $\widehat{\mathbf{K}}^{(\ell,h)}$ are the native position-encoded query and key tensors for head h, with head dimension d_h. The sets $\mathcal{I}_{\mathrm{q}}$ and $\mathcal{I}_{\mathrm{v}}$ index question and visual positions, and $\mathbf{M}_{\mathrm{q}}$ is the corresponding slice of the causal attention mask. Softmax spans all valid keys before the visual columns are extracted, yielding $\mathbf{A}_{\mathrm{q}\to\mathrm{v}}^{(\ell,h)} \in \mathbb{R}^{T_q\times N_v}$. Averaging over the H heads and T_q question tokens gives:

$$a_i = \frac{1}{HT_q}\sum_{h=1}^{H}\sum_{t=1}^{T_q}\mathbf{A}_{\mathrm{q}\to\mathrm{v}}^{(\ell,h)}[t,i], \tag{3}$$

where a_i is the raw importance of visual patch i, with $i \in \{1,\dots,N_v\}$. LEAF then suppresses spatial bias using a fixed positional basis $\mathbf{b}$. For a square grid, b is the normalized row-coordinate ramp flattened in visual-token order; otherwise, it is a normalized one-dimensional token-order ramp:

$$\mathbf{a}' = \mathbf{a}_c - \beta\,\langle \mathbf{a}_c, \mathbf{b}\rangle\,\mathbf{b} + \bar{a}\mathbf{1}, \qquad \mathbf{a}_c = \mathbf{a} - \bar{a}\mathbf{1}. \tag{4}$$

Here, $\mathbf{a} = [a_1,\dots,a_{N_v}]^\top$, $\bar{a}$ is its mean, $\mathbf{a}_c$ is the centered form, $\mathbf{1} \in \mathbb{R}^{N_v}$ is the all-ones vector, and $\langle\cdot,\cdot\rangle$ is the Euclidean inner product. The parameter $\beta \geq 0$ controls removal of the position-aligned component, while the final term restores the original mean, producing the debiased scores $\mathbf{a}'$. Min–max normalization yields the final patch importance scores $\omega_i \in [0,1]$ used by leaf aggregation and selection. Appendix A.3 provides implementation details.

3.4 Leaf-Level Information Aggregation

After both parallel branches finish, LEAF aggregates the host-VLM patch features and question-conditioned importance scores within each hierarchy leaf. Let $\mathcal{C}_j$ be the set of original patch indices assigned to leaf j, $\mathbf{f}_i \in \mathbb{R}^d$ be the pre-projector vision-encoder feature of patch i, and $\omega_i \in [0,1]$ be its importance score. For the N hierarchy leaves, LEAF computes

$$\boldsymbol{\phi}_j = \frac{1}{|\mathcal{C}_j|}\sum_{i\in\mathcal{C}_j}\mathbf{f}_i, \qquad \mu_j = \max_{i\in\mathcal{C}_j}\omega_i, \qquad j = 1,\dots,N. \tag{5}$$

Here, $\boldsymbol{\phi}_j \in \mathbb{R}^d$ is the mean feature of leaf j,, and $\mu_j \in [0,1]$ is its maximum patch importance. Mean aggregation summarizes the leaf's visual content, whereas maximum aggregation prevents its strongest question-relevant evidence from being diluted by less relevant patches. These leaf-level statistics guide hierarchy-aware selection without replacing or altering the original patch features or their corresponding language-model-space visual tokens.

3.5 Hierarchy-Aware Token Selection

LEAF allocates a budget of B visual tokens by jointly considering regional coverage, question relevance, and within-leaf detail. To reduce the cost of repeated feature comparisons, it applies the same fixed, seeded, signed Rademacher projection to every leaf and patch feature:

$$\widetilde{\boldsymbol{\phi}}_j = \mathbf{R}^\top\boldsymbol{\phi}_j, \qquad \widetilde{\mathbf{f}}_i = \mathbf{R}^\top\mathbf{f}_i, \qquad \mathbf{R} \in \mathbb{R}^{d\times d_r}. \tag{6}$$

Here, the entries of $\mathbf{R}$ are sampled once from $\{-1/\sqrt{d_r}, +1/\sqrt{d_r}\}$ and fixed during inference, yielding $\widetilde{\boldsymbol{\phi}}_j, \widetilde{\mathbf{f}}_i \in \mathbb{R}^{d_r}$. Motivated by the Johnson–Lindenstrauss (JL) lemma (Johnson & Lindenstrauss, 1984) and its Rademacher realization (Achlioptas, 2003), this training-free projection approximately preserves pairwise geometry after dimensionality reduction.

In the shared projected space, LEAF computes the inter-leaf similarity s_{ju}, within-leaf heterogeneity h_j, and heterogeneity-derived detail scale d_j:

$$s_{ju} = \cos\left(\widetilde{\boldsymbol{\phi}}_j, \widetilde{\boldsymbol{\phi}}_u\right), \qquad h_j = \frac{1}{|\mathcal{C}_j|}\sum_{i\in\mathcal{C}_j}\left[1 - \cos\left(\widetilde{\mathbf{f}}_i, \widetilde{\boldsymbol{\phi}}_j\right)\right], \qquad d_j = \max\left(h_j^{\sigma}, d_{\min}\right). \tag{7}$$

Here, $s_{ju} \in [-1,1]$ measures how well leaf j represents leaf u, while h_j is the average cosine distance between the leaf's patches and mean feature; a larger value indicates more diverse within-leaf content. The detail scale d_j, controlled by exponent $\sigma > 0$ and minimum value $d_{\min} \geq 0$, allows more heterogeneous leaves to receive additional patches.

Before selection, the patches within each leaf are sorted in descending order of their importance ω_i. At each iteration, the highest-ranked unselected patch i_j becomes the candidate of leaf j, and t_j denotes the number of patches already selected from that leaf. A leaf is active when $t_j > 0$, and the current set of active leaves is $\mathcal{A} = \{j : t_j > 0\}$.

Inspired by weighted facility-location selection (Lin & Bilmes, 2011), LEAF measures how well the active leaves collectively represent all hierarchy leaves:

$$\begin{aligned} \mathcal{F}(\mathcal{A}) &= \sum_{u=1}^{N} \mu_u \max_{k \in \mathcal{A}} s_{ku}, \qquad c_u = \max_{k \in \mathcal{A}} s_{ku}, \\ \Delta C_j &= \mathcal{F}(\mathcal{A} \cup \{j\}) - \mathcal{F}(\mathcal{A}) = \sum_{u=1}^{N} \mu_u \max\left(0, s_{ju} - c_u\right). \end{aligned} \tag{8}$$

Here, u indexes a target leaf, μ_u weights its question-conditioned importance, and k indexes the active leaves available to represent it. Thus, $\mathcal{F}(\mathcal{A})$ is the total importance-weighted coverage provided by the active leaves, while c_u stores the best current coverage of target leaf u. For an inactive candidate leaf j, ΔC_j is the marginal coverage gained by adding j to $\mathcal{A}$; it is positive only where s_{ju} improves upon c_u. Eq. 8 applies once at least one leaf is active. For the first iteration, where $A = \varnothing$, we directly compute the gain of activating leaf j as $\Delta C_j^{(0)} = \sum_{u=1}^{N} \mu_u s_{ju}$. This initialization equals the coverage obtained by activating leaf j when no other leaf is active. Although the coverage term follows a facility-location objective (Nemhauser et al., 1978), the complete gain in Eq. 8 is not claimed to be submodular. Repeated allocation within an active leaf is controlled by its detail scale and a diminishing-return factor:

$$\delta_j = \frac{d_j}{t_j^{\gamma}},\ t_j > 0, \qquad G_j = \begin{cases} \Delta C_j + \lambda \omega_{i_j}, & t_j = 0, \\ \lambda \delta_j \omega_{i_j}, & t_j > 0. \end{cases} \tag{9}$$

Here, δ_j is the current detail factor of leaf j, G_j is the gain of selecting its candidate patch i_j, $\gamma > 0$ controls repeated-allocation decay, and $\lambda \geq 0$ balances relevance and coverage. The first selection from a leaf combines coverage and relevance; subsequent selections combine relevance with progressively discounted, heterogeneity-derived detail.

At each iteration, LEAF selects $j^* = \arg\max_j G_j$ over leaves with remaining candidates and adds i_{j^*} to the selected index set. If j^* was inactive before selection, it is added to $\mathcal{A}$, and the coverage state is updated as $c_u \leftarrow \max(c_u, s_{j^*u})$ for every target leaf u. LEAF then increments t_{j^*}. If the winning leaf was already active, the coverage state remains unchanged. The procedure continues until B patches have been selected.

Finally, the selected indices are restored to their original order to gather the corresponding language-model-space visual tokens. Together with the unchanged system and question tokens, they form a compressed sequence that undergoes a separate full forward pass through the frozen language model. Complete implementation details are provided in Appendix A.4.

4 Experiments

We evaluate LEAF on LLaVA-1.5-7B/13B (Liu et al., 2024a), LLaVA-NeXT-7B (Liu et al., 2024b), and Qwen2.5-VL-7B-Instruct (Bai et al., 2025). Both LLaVA-1.5 models use fixed sequences of 576 visual tokens; LLaVA-NeXT produces up to 2,880 tokens from high-resolution inputs; and Qwen2.5-VL uses dynamic-resolution tokenization. At matched token budgets, we compare LEAF with VisionZip (Yang et al., 2025), DivPrune (Alvar et al., 2025), and ZOO-Prune (Kim et al., 2026). We report normalized performance retention over nine benchmarks for the LLaVA models and four for Qwen2.5-VL, together with Qwen2.5-VL BD-Token relative to random retention. All methods are training-free and evaluated on a single A800 GPU. Appendix B provides complete configurations, protocols, and metric definitions.

4.1 Performance Comparison Across VLMs and Token Budgets

Overall comparison. Figure 1 summarizes performance across four frozen VLMs, while Tables 1, 2, and 3 provide benchmark-level comparisons for three representative backbones. LEAF achieves

Table 1: **Performance comparison on LLaVA-1.5-7B at aggressive token budgets.** Avg. denotes mean benchmark-wise performance relative to the full-token model. Bold indicates the best compressed result at each budget.

Method	GQA Acc.↑	MMB Acc.↑	MME P+C↑	POPE F1↑	SQA Acc.↑	VQA^{v2} Acc.↑	VQA^{Text} Acc.↑	MMMU Acc.↑	SEED-I Acc.↑	Avg. ↑
Total 576 Visual Tokens										
LLaVA-1.5-7B	61.90	64.70	1862.00	85.90	69.50	78.50	58.20	36.30	58.60	100.00%
Retain 128 Tokens ↓ 77.8%										
VisionZip	57.60	62.00	1761.70	83.20	68.90	75.60	56.80	**37.90**	54.90	96.80%
DivPrune	59.25	62.03	1718.22	86.72	**68.96**	**75.96**	56.06	35.56	56.98	96.90%
ZOO-Prune	59.32	61.77	1745.44	**87.24**	68.86	75.84	**56.91**	35.56	57.40	97.33%
LEAF (Ours)	**59.33**	**63.14**	**1764.96**	86.37	68.32	74.58	56.81	36.11	**58.17**	**97.60%**
Retain 64 Tokens ↓ 88.9%										
VisionZip	55.10	60.10	1690.00	77.00	69.00	72.40	55.50	**36.20**	52.20	93.10%
DivPrune	57.78	59.28	1674.40	85.56	68.17	**74.11**	54.69	35.56	55.13	94.80%
ZOO-Prune	**58.32**	60.39	1675.05	85.80	68.12	73.01	55.47	35.11	55.48	95.01%
LEAF (Ours)	58.24	**62.11**	**1703.62**	**86.21**	**69.06**	73.48	**56.27**	35.44	**57.28**	**96.33%**

Table 2: **Performance comparison on LLaVA-NeXT-7B at aggressive token budgets.**

Method	GQA Acc.↑	MMB Acc.↑	MME P+C↑	POPE F1↑	SQA Acc.↑	VQA^{v2} Acc.↑	VQA^{Text} Acc.↑	MMMU Acc.↑	SEED-I Acc.↑	Avg. ↑
Total up to 2,880 Visual Tokens										
LLaVA-NeXT-7B	64.20	67.90	1842.00	86.40	70.20	80.10	61.30	35.10	70.20	100.00%
Retain 320 Tokens ↓ 88.9%										
VisionZip	59.30	63.10	1702.00	82.10	67.30	76.20	58.90	35.30	63.40	94.52%
DivPrune	59.63	63.66	1731.04	83.47	67.82	76.64	53.84	**37.11**	65.35	95.13%
ZOO-Prune	61.07	64.86	1762.67	85.27	67.18	**78.08**	57.51	37.00	66.32	96.88%
LEAF (Ours)	**61.16**	**65.55**	**1779.27**	**86.98**	**68.42**	76.82	**59.05**	**37.11**	**67.54**	**97.86%**
Retain 160 Tokens ↓ 94.4%										
VisionZip	55.50	60.10	1630.00	74.80	**68.30**	71.40	56.20	36.10	58.30	90.45%
DivPrune	57.79	62.29	1658.25	79.36	68.02	73.92	52.42	**36.44**	62.54	92.36%
ZOO-Prune	**59.66**	63.83	1722.01	83.04	67.92	**76.12**	55.55	35.89	64.32	94.76%
LEAF (Ours)	59.42	**65.55**	**1752.12**	**86.69**	68.17	75.18	**56.73**	36.11	**65.56**	**96.04%**

the highest normalized average performance at every evaluated model–budget setting. Its advantage generally increases as the token budget decreases, indicating that jointly preserving regional coverage and question-relevant detail is particularly valuable under aggressive compression. Although no method dominates every individual benchmark, LEAF achieves the highest normalized average across the evaluated models and budgets. On Qwen2.5-VL, its stronger performance–compression curve yields a BD-Token of -75.94% relative to random retention, corresponding to lower token requirements at matched performance.

Across model architectures. On LLaVA-1.5-7B, LEAF retains 96.33% of full-token performance at 64 tokens, exceeding the strongest baseline by 1.32 percentage points. LEAF also ranks first at all three budgets on LLaVA-1.5-13B, showing that the same method remains effective with the larger language backbone. On high-resolution LLaVA-NeXT, LEAF retains 97.86% and 96.04% of full-token performance at 320 and 160 tokens, respectively, demonstrating its effectiveness for longer visual sequences. The advantage is most pronounced on dynamic-resolution Qwen2.5-VL: LEAF retains 97.72% and 96.08% of full-token performance at 20% and 10% retention, respectively, and surpasses the next strongest evaluated baseline by 5.32 percentage points at 10%, where it also performs best on every evaluated benchmark. Complete results across all budgets, including the extended Qwen2.5-VL retention ratios, and the LLaVA-1.5-13B comparison are provided in Appendices C.1 and C.2.

Computational efficiency. LEAF uses more FLOPs than the compression baselines because of truncated probing. However, its CPU hierarchy branch overlaps with GPU probing, and its measured prefill and end-to-end latencies fall between those of the fastest and slowest baselines at all

Table 3: **Qwen2.5-VL-7B performance at the primary visual-token retention ratios.** Random retention is averaged over three seeds.

Method	*Retain 20% of Visual Tokens* ↓ 80.0%					*Retain 10% of Visual Tokens* ↓ 90.0%				
	GQA	MMB	MME	POPE	Avg.	GQA	MMB	MME	POPE	Avg.
Qwen2.5-VL-7B	60.84	84.10	2310	86.30	100.00	60.84	84.10	2310	86.30	100.00
Random retention	57.12	76.06	2144	81.07	92.77	53.51	70.25	1917	75.36	85.45
VisionZip	57.27	79.72	2221	83.89	95.57	54.09	76.03	1937	78.97	88.67
DivPrune	**60.05**	79.55	2173	83.42	96.01	55.49	76.03	2054	79.05	90.53
ZOO-Prune	58.81	80.60	2201	84.17	96.33	55.45	76.28	2018	80.99	90.76
LEAF (Ours)	58.44	**81.27**	**2257**	**86.71**	**97.72**	**57.22**	**78.52**	**2219**	**87.02**	**96.08**

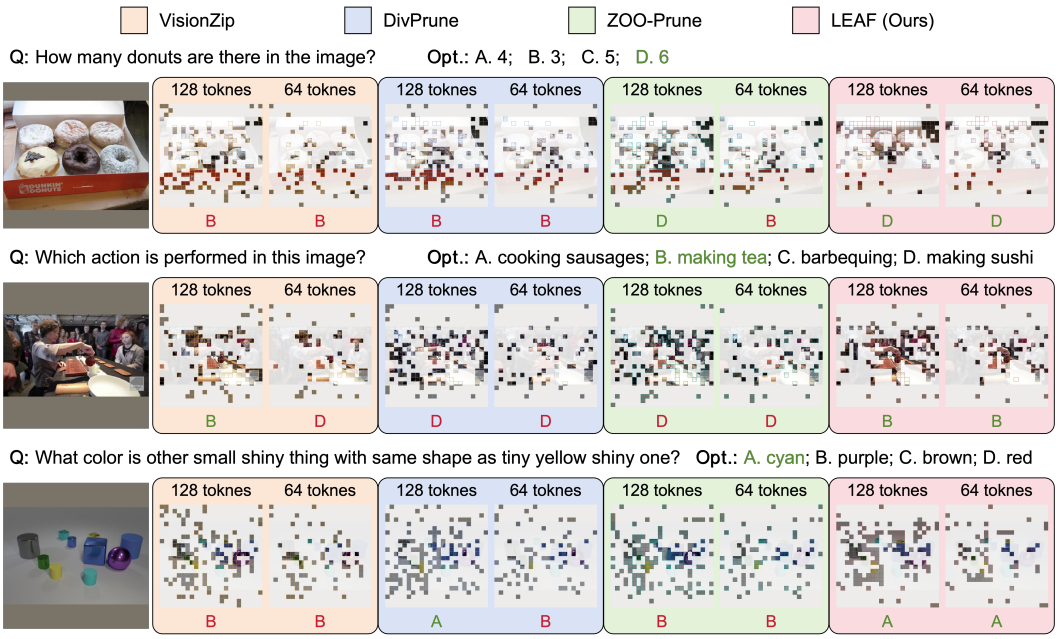

Figure 3: **Qualitative comparison with LLaVA-1.5-7B.** Each method is evaluated with 128 and 64 retained visual tokens. Correct and incorrect predictions are shown in green and red, respectively.

three evaluated budgets. Detailed FLOPs calculations and latency comparisons are provided in Appendix C.3.

Qualitative comparison. Figure 3 presents MMBench examples involving counting, action understanding, and color recognition. LEAF answers all three correctly at both 128 and 64 tokens, whereas competing methods become less reliable at the lower budget. In these examples, LEAF retains the visual evidence needed for the queried count, action, or color at both budgets. Additional qualitative examples from the remaining eight benchmarks are provided in Appendix C.4.

4.2 Ablation and Component Analysis

We conduct ablation studies using LLaVA-1.5-7B with 128 retained visual tokens and evaluate on GQA, MMBench, MME, and POPE. Detailed configurations can be found in Appendix B.5.

Necessity of each stage. The left block of Table 4 evaluates LEAF's three stages. "w/o S1" replaces the adaptive hierarchy with singleton patches, while "w/o S2" replaces question-aware relevance with uniform scores. "w/o S3" retains the hierarchy and relevance scores but replaces the complete selector with relevance-only leaf-wise selection, removing coverage, inter-leaf similarity, heterogeneity, and repeated-allocation decay. Removing either *S1* or *S2* lowers all four benchmark scores, indicating that adaptive spatial organization and question conditioning contribute complementary information. Replacing *S3* causes a much larger drop, showing that the leaf-wise relevance baseline overconcentrates the budget without LEAF's coverage and detail terms.

Table 4: **Component ablation on LLaVA-1.5-7B at the 128 token budget.** The left block removes the image-driven hierarchy (S1), question-aware evidence extraction (S2), or hierarchy-aware selection (S3), while the right block examines coverage weighting, detail decay, and heterogeneity modeling within the selector. "Full" denotes complete LEAF.

Benchmark	Necessity of each step			Selector mechanism				Full
	w/o S1	w/o S2	w/o S3	Coverage only	Source-weighted	No detail decay	No hetero-geneity	
GQA	58.74	58.65	50.06	57.25	58.95	59.03	58.99	**59.33**
MMB	62.12	61.94	47.51	60.54	62.65	62.85	62.55	**63.14**
MME	1739	1731	1408	1736	1762	1764	1761	**1765**
POPE	85.08	84.63	69.59	83.92	86.21	86.27	86.18	**86.37**

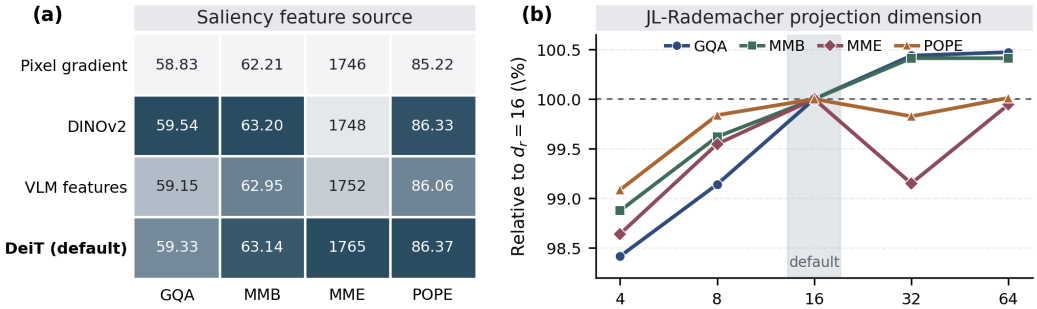

Figure 4: **Ablations of the saliency feature source and projection dimension.** All experiments use LLaVA-1.5-7B with 128 retained visual tokens. **(a)** Performance using different sources for constructing the saliency map. Cells report exact benchmark scores, with shading scaled within each benchmark. **(b)** Sensitivity to the JL-Rademacher projection dimension. Each benchmark is normalized to the default $d_r = 16$, indicated by the shaded region.

Selector mechanism. The right block of Table 4 examines the components of hierarchy-aware selection. "Coverage only" retains only marginal regional coverage, while "source-weighted coverage" weights coverage by candidate-leaf rather than target-leaf relevance. The remaining variants remove detail decay or heterogeneity-aware scaling. Coverage only performs worst on all four benchmarks, indicating that opening diverse regions is insufficient without allocating additional patches within them. Source weighting, removal of detail decay, and removal of heterogeneity cause smaller losses, but each is below full LEAF on all four benchmarks.

Saliency feature source. We replace the DeiT features used by the image-driven branch with pixel gradients, DINOv2 features, or the host VLM's visual features. As shown in Figure 4(a), DINOv2 performs better on GQA and MMBench, whereas DeiT achieves the highest MME and POPE scores, with competitive GQA and MMBench performance. Pixel gradients perform consistently worse, indicating that feature-level representations provide more reliable signal than low-level image gradients. Overall, DeiT is a robust source for constructing the saliency-guided hierarchy.

Projection dimension. We vary the dimension of the fixed JL-Rademacher projection over $d_r \in \{4, 8, 16, 32, 64\}$, leaving all other selector settings unchanged. Figure 4(b) shows that performance improves consistently from $d_r = 4$ to $d_r = 16$, after which the gains largely saturate. Dimensions 32 and 64 provide small improvements on GQA and MMBench but do not consistently improve MME or POPE. We therefore use $d_r = 16$ by default.

Additional ablations in Appendix C.5 examine the QTMT split families, attention-probe depth, and selector feature reduction.

5 Conclusion

We presented LEAF, a training-free framework that combines a saliency-guided adaptive hierarchy, truncated question-aware probing, and hierarchy-aware allocation to preserve both regional coverage and task-relevant detail. Across four frozen VLMs and nine benchmarks, LEAF achieves the strongest normalized performance at all evaluated model–budget settings, with particularly clear advantages under aggressive compression. It achieves a BD-Token of -75.94% relative to random

retention, corresponding to lower token requirements at matched performance, while extensive ablations validate its principal design choices. LEAF currently uses a prespecified token budget and incurs additional computation from full-token probing; these limitations and corresponding directions toward adaptive budgeting and in-model token selection are discussed in Appendix D.

AI USE STATEMENT

We used generative AI tools, including ChatGPT and OpenAI Codex, to assist with software implementation and debugging; provide feedback on experimental and ablation design; support the interpretation and presentation of quantitative and qualitative results; and express parts of the author-developed implementation in mathematical notation. We also used these tools to search and summarize relevant literature, draft and edit manuscript text, improve readability, assist with figure and table design, format references, and suggest manuscript structure.

We did not use generative AI to generate synthetic datasets, produce experimental measurements or benchmark outputs, develop theoretical models or conceptual frameworks, formulate theoretical claims, provide critical ingredients for or assist in writing mathematical proofs, propose or refine research hypotheses, translate content, clean or reformat evaluation datasets, or conduct thematic data analysis. These activities were performed independently by the authors or were not applicable to this work.

All AI-assisted material was reviewed by the authors. Code was inspected and tested, mathematical descriptions were checked against the implementation, reported values were traced to evaluation logs, and literature claims and citations were verified against their original sources. The authors made all final methodological, experimental, and editorial decisions and take responsibility for the content and claims of this work.

ETHICS STATEMENT

This work introduces a training-free inference method and does not involve human participants, private data, or the collection or release of new personal information. Our experiments use publicly available pretrained models and established benchmarks subject to their respective licenses and usage conditions. Because LEAF operates on frozen VLMs, it inherits their existing biases, inaccuracies, and potential for harmful outputs. Visual-token reduction may also discard evidence unevenly across inputs or tasks, particularly under aggressive compression; therefore, the reported benchmark performance should not be interpreted as establishing fairness, robustness, or safety in sensitive applications. Deployments in high-stakes settings should independently evaluate compression-induced failure modes and retain appropriate human oversight or full-token fallback mechanisms.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of LEAF's hierarchy construction, question-aware probing, and hierarchy-aware selection in Appendices A.2–A.4, including the corresponding equations, algorithmic procedures, and implementation settings. Appendix B specifies the model checkpoints, token budgets, benchmark splits, evaluation protocol, baseline implementations, and hyperparameters used in our experiments. The complete BD-Token and computational-efficiency calculations are documented in Appendices B.4 and C.3, respectively, and complete results and ablation configurations are also provided in the appendix. The original implementation is currently undergoing institutional compliance review and is therefore not publicly available at the time of submission. Subject to institutional approval, the implementation may be made available upon reasonable request and released publicly at a later date.

REFERENCES

Dimitris Achlioptas. Database-Friendly Random Projections: Johnson–Lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671–687, 2003. doi: 10.1016/S0022-0000(03)00025-4.

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, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: A visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 23716–23736, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/960a172bc7fbf0177ccccbb411a7d800-Abstract-Conference.html.

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

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*, 2025.

Gisle Bjontegaard. Calculation of average PSNR differences between RD-curves. *ITU SG16 Doc. VCEG-M33*, 2001.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token Merging: Your ViT but faster. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=JroZRaRw7Eu.

Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J. Sullivan, and Jens-Rainer Ohm. Overview of the versatile video coding standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10):3736–3764, 2021. doi: 10.1109/TCSVT.2021.3101953.

Davide Caffagni, Federico Cocchi, Luca Barsellotti, Nicholas Moratelli, Sara Sarto, Lorenzo Baraldi, Marcella Cornia, and Rita Cucchiara. The revolution of multimodal large language models: A survey. *Findings of the association for computational linguistics: ACL 2024*, pp. 13590–13618, 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*, volume 15139 of *Lecture Notes in Computer Science*, pp. 19–35. Springer, 2024. doi: 10.1007/978-3-031-73004-7_2. URL https://www.ecva.net/papers/eccv_2024/papers_ECCV/html/10478_ECCV_2024_paper.php.

Kyle R. Chickering, Bangzheng Li, and Muhao Chen. QLIP: A dynamic quadtree vision prior enhances MLLM performance without retraining. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=DSq3r8PjpQ.

Xiangxiang Chu, Limeng Qiao, Xinyang Lin, Shuang Xu, Yang Yang, Yiming Hu, Fei Wei, Xinyu Zhang, Bo Zhang, Xiaolin Wei, and Chunhua Shen. MobileVLM: A fast, strong and open vision language assistant for mobile devices. *arXiv preprint arXiv:2312.16886*, 2023. URL https://arxiv.org/abs/2312.16886.

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. Mme: A comprehensive evaluation benchmark for multimodal large language models. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi,

and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc., 2025. doi: 10.52202/085713-4899. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/d79a27cf2772fe00be7f341efc0eb517-Paper-Datasets_and_Benchmarks_Track.pdf.

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

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

William B. Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. In *Conference in Modern Analysis and Probability*, volume 26 of *Contemporary Mathematics*, pp. 189–206. American Mathematical Society, 1984. doi: 10.1090/conm/026/737400.

Youngeun Kim, Youjia Zhang, Huiling Liu, Aecheon Jung, Sunwoo Lee, and Sungeun Hong. Zooprune: Training-free token pruning via zeroth-order gradient estimation in vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 39572–39582, 2026.

Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. SEED-Bench: Benchmarking multimodal large language models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13299–13308. IEEE, 2024.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. 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*, pp. 19730–19742. PMLR, 2023a. URL https://proceedings.mlr.press/v202/li23q.html.

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

Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 510–520, Portland, Oregon, USA, 2011. Association for Computational Linguistics. URL https://aclanthology.org/P11-1052/.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34892–34916, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/6dcf277ea32ce3288914faf369fe6de0-Abstract-Conference.html.

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, 2024a. URL https://openaccess.thecvf.com/content/CVPR2024/html/Liu_Improved_Baselines_with_Visual_Instruction_Tuning_CVPR_2024_paper.html.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. LLaVA-NeXT: Improved reasoning, OCR, and world knowledge, January 2024b. URL https://llava-vl.github.io/blog/2024-01-30-llava-next/.

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

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.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14:265–294, 1978. doi: 10.1007/BF01588971.

Michael S. Ryoo, A. J. Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. TokenLearner: Adaptive space-time tokenization for videos. In *Advances in Neural Information Processing Systems*, volume 34, pp. 12786–12797, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/6a30e32e56fce5cf381895dfe6ca7b6f-Abstract.html.

Kele Shao, Keda Tao, Kejia Zhang, Sicheng Feng, Mu Cai, Yuzhang Shang, Haoxuan You, Can Qin, Yang Sui, and Huan Wang. A survey of token compression for efficient multimodal large language models. *arXiv preprint arXiv:2507.20198*, 2025.

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

Shitao Tang, Jiahui Zhang, Siyu Zhu, and Ping Tan. Quadtree attention for vision transformers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=fR-EnKWL_Zb.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 2021.

Haicheng Wang, Zhemeng Yu, Gabriele Spadaro, Chen Ju, Victor Quétu, Shuai Xiao, and Enzo Tartaglione. FOLDER: Accelerating multi-modal large language models with enhanced performance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 23614–23625, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Wang_FOLDER_Accelerating_Multi-Modal_Large_Language_Models_with_Enhanced_Performance_ICCV_2025_paper.html.

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. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024. URL https://arxiv.org/abs/2409.12191.

Jiayang Wu, Wensheng Gan, Zefeng Chen, Shicheng Wan, and Philip S Yu. Multimodal large language models: A survey. In *2023 IEEE international conference on big data (bigdata)*, pp. 2247–2256. IEEE, 2023.

Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, et al. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding. *arXiv preprint arXiv:2412.10302*, 2024.

Peng Xu, Bryan He, Christopher De Sa, Ioannis Mitliagkas, and Chris Re. Accelerated stochastic power iteration. In *International Conference on Artificial Intelligence and Statistics*, pp. 58–67. PMLR, 2018.

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. doi: 10.1109/CVPR52734.2025.01843. URL https://openaccess.thecvf.com/content/CVPR2025/html/Yang_VisionZip_Longer_is_Better_but_Not_Necessary_in_Vision_Language_CVPR_2025_paper.html.

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, 2025a.

Xubing Ye, Yukang Gan, Xiaoke Huang, Yixiao Ge, and Yansong Tang. VoCo-LLaMA: Towards vision compression with large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 29836–29846, 2025b. URL https://openaccess.thecvf.com/content/CVPR2025/html/Ye_VoCo-LLaMA_Towards_Vision_Compression_with_Large_Language_Models_CVPR_2025_paper.html.

Hongxu Yin, Arash Vahdat, Jose M. Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-ViT: Adaptive tokens for efficient vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10799–10808, 2022. doi: 10.1109/CVPR52688.2022.01054.

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

Jun Zhang, Yicheng Ji, Feiyang Ren, Yihang Li, Bowen Zeng, Zonghao Chen, Ke Chen, Lidan Shou, Gang Chen, and Huan Li. Efficient inference for large vision-language models: Bottlenecks, techniques, and prospects. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 21036–21066, 2026.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, et al. Lmms-eval: Reality check on the evaluation of large multimodal models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 881–916, 2025a.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis A. Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. SparseVLM: Visual token sparsification for efficient vision-language model inference. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 74840–74857. PMLR, 2025b. URL https://proceedings.mlr.press/v267/zhang25s.html.

A Additional Method Details

A.1 Overall LEAF Procedure

Algorithm 1 describes the overall procedure of LEAF in detail.

Algorithm 1 Overall LEAF procedure

Require: Input image x, question q, frozen host VLM, token budget B, and probe layer ℓ
Ensure: Selected indices $\mathcal{I}_{\text{sel}}$ and model output y

Obtain pre-projector patch features $\{\mathbf{f}_i\}_{i=1}^{N_v}$, original visual tokens $\mathbf{H}_{\text{vis}}$, system tokens $\mathbf{H}_{\text{sys}}$, and question tokens $\mathbf{H}_q$
Construct the complete multimodal sequence $\mathbf{H}^{(0)} = [\mathbf{H}_{\text{sys}}; \mathbf{H}_{\text{vis}}; \mathbf{H}_q]$

Parallel image- and question-driven branches:
    Image branch: extract auxiliary features, construct the saliency map $\mathbf{S}$, and apply saliency-guided QTMT partitioning
    Obtain the leaves $\mathcal{L} = \{\mathcal{C}_j\}_{j=1}^{N}$ and their original patch-index memberships
    Question branch: execute the frozen language model only to probe layer ℓ
    Replay question-to-visual attention, remove positional bias, and obtain patch importance $\boldsymbol{\omega} = [\omega_1, \ldots, \omega_{N_v}]^\top$

Leaf aggregation and pre-computation:
for $j = 1, \ldots, N$ **do**
    Compute the mean leaf feature $\boldsymbol{\phi}_j = |\mathcal{C}_j|^{-1} \sum_{i \in \mathcal{C}_j} \mathbf{f}_i$
    Compute the leaf importance $\mu_j = \max_{i \in \mathcal{C}_j} \omega_i$
end for
Apply the shared fixed Rademacher projection to all leaf and patch features
Compute leaf similarities s_{ju}, heterogeneity values h_j, and detail scales d_j
Sort the patches in every leaf by decreasing ω_i, yielding the rankings π_j

Hierarchy-aware greedy selection:
Initialize $\mathcal{I}^{(0)} \leftarrow \varnothing$, $\mathcal{A} \leftarrow \varnothing$, $t_j \leftarrow 0$, and $c_u \leftarrow -\infty$
for $b = 1, \ldots, B$ **do**
    Let $\mathcal{J}_b = \{j : t_j < |\mathcal{C}_j|\}$ be the leaves with remaining patches
    for all $j \in \mathcal{J}_b$ **do**
        Set the candidate patch $i_j \leftarrow \pi_j(t_j + 1)$
        if $\mathcal{A} = \varnothing$ **then**
            Compute $\Delta C_j \leftarrow \sum_{u=1}^{N} \mu_u s_{ju}$
        else if $j \notin \mathcal{A}$ **then**
            Compute $\Delta C_j \leftarrow \sum_{u=1}^{N} \mu_u \max(0, s_{ju} - c_u)$
        end if
        Compute the detail factor δ_j and candidate gain G_j
    end for
    Select $j^* \leftarrow \arg\max_{j \in \mathcal{J}_b} G_j$ and $i^* \leftarrow i_{j^*}$
    $\mathcal{I}^{(b)} \leftarrow \mathcal{I}^{(b-1)} \cup \{i^*\}$
    if $j^* \notin \mathcal{A}$ **then**
        $\mathcal{A} \leftarrow \mathcal{A} \cup \{j^*\}$
        $c_u \leftarrow \max(c_u, s_{j^*u})$ for all $u = 1, \ldots, N$
    end if
    $t_{j^*} \leftarrow t_{j^*} + 1$
end for
Restore the original visual order: $\mathcal{I}_{\text{sel}} \leftarrow \text{sort}(\mathcal{I}^{(B)})$
Gather the corresponding original visual tokens: $\mathbf{H}_{\text{vis}}^{\text{sel}} \leftarrow \mathbf{H}_{\text{vis}}[\mathcal{I}_{\text{sel}}]$
Construct $\mathbf{H}_{\text{comp}} = [\mathbf{H}_{\text{sys}}; \mathbf{H}_{\text{vis}}^{\text{sel}}; \mathbf{H}_q]$
Run a separate complete forward pass: $y \leftarrow \text{VLM}(\mathbf{H}_{\text{comp}})$
return $\mathcal{I}_{\text{sel}}, y$

A.2 Detailed Saliency-Driven Image Partition

Figure 5 shows the detailed saliency-extraction and hierarchy-construction process used in Sec. 3.2.

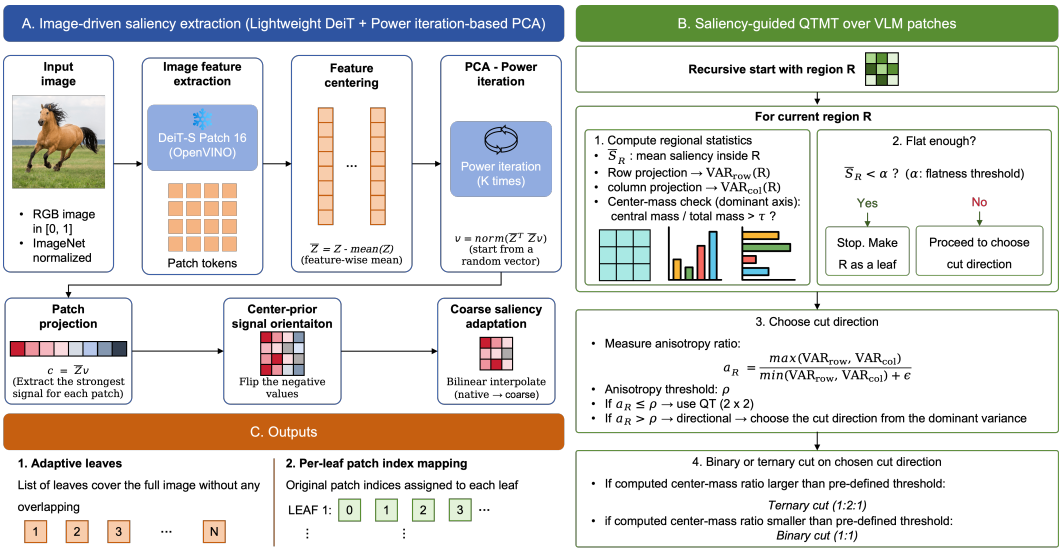

Figure 5: **Detailed workflow of saliency-driven image partition. A.** A frozen DeiT-S encoder produces auxiliary patch features. Feature centering and power-iteration-based PCA extract a spatial saliency map. **B.** Saliency-guided quadtree plus multi-type tree (QTMT) partitioning recursively selects among quadtree (QT), binary horizontal tree (BHT), binary vertical tree (BVT), ternary horizontal tree (THT), and ternary vertical tree (TVT) partitions. **C.** The procedure outputs non-overlapping adaptive leaves and the original VLM patch indices assigned to each leaf.

Auxiliary feature centering. The normalized hierarchy input is processed by a frozen DeiT-S/16 encoder, producing a feature grid of resolution $H_e \times W_e$. After flattening the spatial dimensions, the resulting matrix is $\mathbf{Z} \in \mathbb{R}^{M \times d_s}$, where $M = H_e W_e$. LEAF centers these features as

$$\bar{\mathbf{z}} = \frac{1}{M} \sum_{u=1}^{M} \mathbf{z}_u, \qquad \widetilde{\mathbf{Z}} = \mathbf{Z} - \mathbf{1}\bar{\mathbf{z}}^\top. \tag{10}$$

Here, $\mathbf{z}_u \in \mathbb{R}^{d_s}$ is the auxiliary feature at spatial location u, $\bar{\mathbf{z}} \in \mathbb{R}^{d_s}$ is the spatial mean, $\mathbf{1} \in \mathbb{R}^M$ is an all-ones vector, and $\widetilde{\mathbf{Z}}$ is the centered feature matrix.

Power-iteration saliency. Starting from a seeded random unit vector $\mathbf{v}^{(0)}$, the dominant direction of feature variation is estimated using

$$\mathbf{v}^{(k+1)} = (\widetilde{\mathbf{Z}}^\top \widetilde{\mathbf{Z}} \mathbf{v}^{(k)}) / (\left\| \widetilde{\mathbf{Z}}^\top \widetilde{\mathbf{Z}} \mathbf{v}^{(k)} \right\|_2), \qquad k = 0, \ldots, K-1. \tag{11}$$

Here, $\mathbf{v}^{(k)} \in \mathbb{R}^{d_s}$ is the estimated principal direction at iteration k, K is the number of iterations, and $\|\cdot\|_2$ denotes the Euclidean norm. This procedure estimates the leading principal direction without explicitly performing a complete eigendecomposition. The centered features are then projected onto the final direction: $\mathbf{r} = \widetilde{\mathbf{Z}} \mathbf{v}^{(K)} \in \mathbb{R}^M$. Here, $\mathbf{r} = [r_1, \ldots, r_M]^\top$ contains one raw response for each auxiliary patch.

Center-prior orientation. Because both $\mathbf{v}^{(K)}$ and $-\mathbf{v}^{(K)}$ are valid principal directions, LEAF determines the response orientation using a weak center prior:

$$\eta = \begin{cases} +1, & \dfrac{1}{|\mathcal{I}_{\mathrm{ctr}}|} \displaystyle\sum_{u \in \mathcal{I}_{\mathrm{ctr}}} r_u \geq \frac{1}{|\mathcal{I}_{\mathrm{bnd}}|} \sum_{u \in \mathcal{I}_{\mathrm{bnd}}} r_u, \\ -1, & \text{otherwise}, \end{cases} \qquad \mathbf{c} = \eta \mathbf{r}. \tag{12}$$

Here, $\mathcal{I}_{\mathrm{ctr}}$ and $\mathcal{I}_{\mathrm{bnd}}$ denote the central and boundary locations of the auxiliary feature grid, respectively; $\eta \in \{-1, +1\}$ is the selected orientation; and $\mathbf{c}$ is the oriented response vector. The center prior only resolves the sign ambiguity and does not directly alter the magnitude or spatial distribution of the responses.

Normalization and grid adaptation. The oriented responses are robustly normalized and sharpened according to

$$\widehat{c}_u = \mathrm{clip}\left(\frac{c_u - Q_{q_l}(\mathbf{c})}{Q_{q_h}(\mathbf{c}) - Q_{q_l}(\mathbf{c}) + \epsilon}, 0, 1\right), \qquad \mathbf{S} = \mathrm{Resize}\left(\widehat{\mathbf{c}}^{\,\gamma_s}; G_h, G_w\right). \tag{13}$$

Here, $Q_{q_l}(\mathbf{c})$ and $Q_{q_h}(\mathbf{c})$ are the lower and upper percentiles of $\mathbf{c}$, $\widehat{c}_u \in [0,1]$ is the normalized response at location u, $\gamma_s > 0$ is the sharpening exponent, and $\mathrm{Resize}(\cdot; G_h, G_w)$ denotes bilinear interpolation to the $G_h \times G_w$ operational grid. The resulting saliency map is denoted by $\mathbf{S}$.

Regional QTMT statistics. For every current rectangular region R, LEAF computes

$$\bar{S}_R = \frac{1}{|R|}\sum_{u \in R} S_u, \qquad v_R^{\mathrm{row}} = \mathrm{Var}\left(\mathrm{RowProj}(R)\right), \qquad v_R^{\mathrm{col}} = \mathrm{Var}\left(\mathrm{ColProj}(R)\right). \tag{14}$$

Here, $\bar{S}_R$ is the mean saliency inside R; $\mathrm{RowProj}(R)$ and $\mathrm{ColProj}(R)$ are its row-wise and column-wise saliency projections; and v_R^{row} and v_R^{col} are their respective variances. The corresponding anisotropy ratio is

$$a_R = \frac{\max\left(v_R^{\mathrm{row}}, v_R^{\mathrm{col}}\right)}{\min\left(v_R^{\mathrm{row}}, v_R^{\mathrm{col}}\right) + \epsilon}. \tag{15}$$

Here, a_R measures the relative dominance of directional saliency variation, and $\epsilon > 0$ prevents division by zero.

For a directional partition, the center-concentration ratio is

$$\kappa_R = \frac{\sum_{u \in R_{\mathrm{center}}} S_u}{\sum_{u \in R} S_u + \epsilon}. \tag{16}$$

Here, R_{center} is the central half of R along the selected cut direction, and κ_R measures how strongly the regional saliency is concentrated around that central interval.

Complete partition rule. Combining the above statistics, the partition decision is

$$\mathcal{D}(R) = \begin{cases} \mathrm{Leaf}, & \bar{S}_R \le \alpha, \\ \mathrm{QT}, & \bar{S}_R > \alpha,\ a_R \le \rho, \\ \mathrm{THT}, & \bar{S}_R > \alpha,\ a_R > \rho,\ v_R^{\mathrm{row}} > v_R^{\mathrm{col}},\ \kappa_R > \tau, \\ \mathrm{BHT}, & \bar{S}_R > \alpha,\ a_R > \rho,\ v_R^{\mathrm{row}} > v_R^{\mathrm{col}},\ \kappa_R \le \tau, \\ \mathrm{TVT}, & \bar{S}_R > \alpha,\ a_R > \rho,\ v_R^{\mathrm{col}} > v_R^{\mathrm{row}},\ \kappa_R > \tau, \\ \mathrm{BVT}, & \bar{S}_R > \alpha,\ a_R > \rho,\ v_R^{\mathrm{col}} > v_R^{\mathrm{row}},\ \kappa_R \le \tau. \end{cases} \tag{17}$$

Here, $\mathcal{D}(R)$ is the decision applied to R, α is the flatness threshold, ρ is the anisotropy threshold, and τ is the center-concentration threshold. QT divides the region into four equal children. BHT and BVT apply equal binary cuts in the horizontal and vertical directions, respectively, whereas THT and TVT apply 1:2:1 ternary cuts. The procedure recurses over each child until the child satisfies the leaf condition or admits no valid split. Each partition decision is applied only when the corresponding split is geometrically valid. If the preferred split is invalid, LEAF falls back to a valid axis-aligned split; recursion terminates when no valid split remains. One operational-grid cell is the smallest partitioning unit.

Host-grid alignment and output. Each terminal region is spatially mapped from the $G_h \times G_w$ operational saliency grid to the $H_v \times W_v$ patch grid of the host VLM. The resulting membership sets satisfy

$$\bigcup_{j=1}^{N} \mathcal{C}_j = \{1, \ldots, N_v\}, \qquad \mathcal{C}_j \cap \mathcal{C}_u = \varnothing \quad \text{for } j \neq u. \tag{18}$$

Here, N is the number of terminal leaves, $N_v = H_v W_v$ is the number of host-VLM patches, and $\mathcal{C}_j$ contains the original patch indices assigned to leaf j. These sets are passed directly to the leaf-aggregation and token-selection stages.

A.3 Detailed Question-Aware Evidence Extraction

This section provides the additional details of the truncated probing process introduced in Sec. 3.3.

Token ranges and truncated execution. The complete multimodal input is

$$\mathbf{H}^{(0)} = [\mathbf{H}_{\text{sys}}; \mathbf{H}_{\text{vis}}; \mathbf{H}_{\text{q}}] \in \mathbb{R}^{T \times d_l}. \tag{19}$$

Here, $\mathbf{H}_{\text{sys}}$ contains T_{sys} system or prefix tokens, $\mathbf{H}_{\text{vis}}$ contains N_v visual tokens, and $\mathbf{H}_{\text{q}}$ contains T_q question tokens. Thus, $T = T_{\text{sys}} + N_v + T_q$, while d_l denotes the language-model embedding dimension. Their sequence-index sets are denoted by $\mathcal{I}_{\text{sys}}$, $\mathcal{I}_{\text{v}}$, and $\mathcal{I}_{\text{q}}$, respectively.

The frozen backbone is executed only until the hidden states required by probe layer ℓ are obtained:

$$\mathbf{H}^{(r+1)} = \mathcal{F}_r\left(\mathbf{H}^{(r)}\right), \qquad r = 0, \ldots, \ell - 1. \tag{20}$$

Here, $\mathcal{F}_r$ is the transformation performed by frozen backbone layer r, and $\mathbf{H}^{(r)}$ is its input hidden sequence. Only layer ℓ is used to derive patch importance; the preceding layers provide its input representations but do not independently contribute importance scores.

Query–key replay. At probe layer ℓ, LEAF constructs the native position-encoded query and key tensors and computes question-to-visual attention as in Eqs. 2–3. Importantly, softmax is evaluated over all valid key positions using the native causal mask before the visual columns are extracted. The resulting attention is averaged over all retained heads and question tokens.

Positional-basis normalization. Let $\mathbf{p} \in \mathbb{R}^{N_v}$ denote the predefined raw spatial-position template used by the implementation. Its zero-mean unit-normalized basis is constructed as

$$\mathbf{p}_c = \mathbf{p} - \left(\frac{1}{N_v}\mathbf{1}^\top \mathbf{p}\right)\mathbf{1}, \qquad \mathbf{b} = \frac{\mathbf{p}_c}{\max(\|\mathbf{p}_c\|_2, \epsilon)}. \tag{21}$$

Here, $\mathbf{p}_c$ is the centered positional template, $\mathbf{b} \in \mathbb{R}^{N_v}$ is the resulting positional basis, $\mathbf{1} \in \mathbb{R}^{N_v}$ is an all-ones vector, and $\epsilon > 0$ prevents division by zero. For fixed-resolution models with an inferred square $H_v \times W_v$ grid, we use the row-coordinate template $p_{r,c} = 2r/(H_v - 1) - 1$, flattened in visual-token order, where $r \in \{0, \ldots, H_v - 1\}$ and $c \in \{0, \ldots, W_v - 1\}$. For dynamic-resolution models, we use a one-dimensional token-order ramp from -1 to 1. Both templates are centered and ℓ_2-normalized.

Projection-based debiasing. Using this basis, LEAF applies the projection-based debiasing operation defined in Eq. 4. Finally, the scores are normalized according to

$$\omega_i = \frac{a'_i - \min_k a'_k}{\max_k a'_k - \min_k a'_k + \epsilon}, \qquad i = 1, \ldots, N_v. \tag{22}$$

Here, $\omega_i \in [0, 1]$ is the final importance score of patch i, and k ranges over all visual patches. The normalized vector $\boldsymbol{\omega} \in \mathbb{R}^{N_v}$ is passed to the leaf-aggregation and hierarchy-aware selection stages. All hidden states produced by this probe are discarded after importance estimation; final answer generation uses a separate complete forward pass over the compressed multimodal sequence.

A.4 Detailed Leaf Aggregation and Hierarchy-Aware Token Selection

Figure 6 presents the complete leaf-aggregation, pre-computation, and greedy token-selection procedure introduced in Secs. 3.4 and 3.5.

Selector inputs and pre-computation. The selector receives the original pre-projector patch features $\{\mathbf{f}_i\}_{i=1}^{N_v}$, normalized patch-importance scores $\{\omega_i\}_{i=1}^{N_v}$, and hierarchy memberships $\{\mathcal{C}_j\}_{j=1}^{N}$. Leaf features $\{\boldsymbol{\phi}_j\}_{j=1}^{N}$ and leaf importance scores $\{\mu_j\}_{j=1}^{N}$ are obtained using Eq. 5. The patch and leaf features are then projected using the same fixed Rademacher matrix as defined in Eq. 6. In the shared projected space, LEAF pre-computes the complete $N \times N$ leaf-similarity matrix, the within-leaf heterogeneity values, and the corresponding detail scales according to Eq. 7. These quantities remain fixed throughout greedy selection.

Within-leaf candidate frontier. The patches in each leaf are sorted once in descending order of their question-conditioned importance:

$$\omega_{\pi_j(1)} \geq \omega_{\pi_j(2)} \geq \cdots \geq \omega_{\pi_j(|\mathcal{C}_j|)}, \qquad \pi_j(r) \in \mathcal{C}_j, \tag{23}$$

where $\pi_j(r)$ is the original index of the r-th highest-ranked patch in leaf j. If t_j patches have already been selected from that leaf, its current candidate is

$$i_j = \pi_j(t_j + 1), \qquad t_j < |\mathcal{C}_j|. \tag{24}$$

Thus, each nonempty leaf contributes at most one candidate per greedy iteration and leaves with no remaining patches are removed from the candidate frontier.

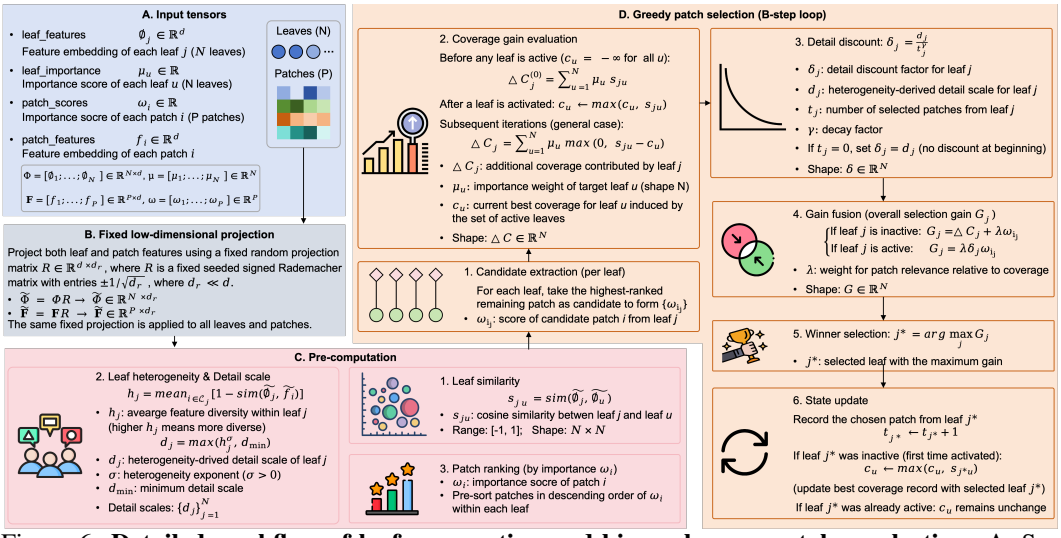

Figure 6: **Detailed workflow of leaf aggregation and hierarchy-aware token selection. A.** Selector inputs and leaf aggregation. **B.** Shared fixed low-dimensional projection. **C.** Pre-computation of leaf similarity, within-leaf heterogeneity, detail scales, and patch rankings. **D.** Greedy allocation using regional coverage, patch relevance, and heterogeneity-derived detail.

Initialization and candidate evaluation. Before selection, LEAF initializes

$$\mathcal{I}^{(0)} = \varnothing, \qquad \mathcal{A}^{(0)} = \varnothing, \qquad t_j^{(0)} = 0, \qquad c_u^{(0)} = -\infty, \tag{25}$$

where $\mathcal{I}^{(0)}$ is the selected-patch set, $\mathcal{A}^{(0)}$ is the active-leaf set, t_j records the number of patches selected from leaf j, and c_u stores the best current coverage of target leaf u.

Candidate gains follow Eqs. 8 and 9. Because no coverage state exists before the first activation, the initial coverage gain is evaluated directly as

$$\Delta C_j^{(0)} = \sum_{u=1}^{N} \mu_u s_{ju}. \tag{26}$$

Once a leaf is active, an inactive candidate receives only the coverage that it adds beyond the cached values $\{c_u\}_{u=1}^N$. Its total gain also includes the relevance of its current patch candidate, as defined in Eq. 9. For an active leaf, no additional regional coverage is credited; its gain depends on the relevance of its next candidate, scaled by its heterogeneity-derived detail factor and the repeated-allocation discount. All gains are evaluated using the pre-update value of t_j.

Winner selection and state update. At each iteration, LEAF selects the highest-gain candidate among leaves with remaining patches and adds its original patch index to the selected set. If the winning leaf j^* was previously inactive, it is added to $\mathcal{A}$ and the coverage cache is updated as

$$c_u \leftarrow \max(c_u, s_{j^* u}), \qquad u = 1, \ldots, N.$$

The coverage cache remains unchanged when the winning leaf is already active. LEAF then increments t_{j^*}, advances that leaf to its next pre-sorted candidate when available, and repeats until $|\mathcal{I}^{(B)}| = B$.

Compressed-sequence construction. After B iterations, the selected patch indices are restored to their original visual-token order and used to gather the corresponding original language-model-space visual tokens. No projected patch or leaf feature is inserted into the multimodal sequence; the low-dimensional representations are used only during selection. The selected visual tokens are concatenated with the unchanged system and question tokens, and the resulting compressed sequence is processed through a separate complete forward pass of the frozen language model.

Implementation efficiency. All leaf similarities, heterogeneity values, detail scales, and within-leaf rankings are computed before the greedy loop. During selection, LEAF maintains only one candidate per nonempty leaf and updates the coverage cache only when a previously inactive leaf is selected, allowing candidate gains to be evaluated using vectorized tensor operations.

B Additional Experimental Details

B.1 Models and Token Budgets

We use the official pretrained checkpoints listed in Table 5.

Table 5: Model checkpoints used in our experiments.

Model	Hugging Face identifier
LLaVA-1.5-7B	`liuhaotian/llava-v1.5-7b`
LLaVA-1.5-13B	`liuhaotian/llava-v1.5-13b`
LLaVA-NeXT-7B	`liuhaotian/llava-v1.6-vicuna-7b`
Qwen2.5-VL-7B	`Qwen/Qwen2.5-VL-7B-Instruct`

LLaVA-1.5 processes each 336×336 image into 576 visual tokens. LLaVA-NeXT uses an any-resolution configuration containing up to five image crops and produces up to 2,880 visual tokens. Qwen2.5-VL employs dynamic-resolution tokenization; therefore, its budgets are specified relative to the number of post-merger visual tokens produced for each input.

For LLaVA-1.5, we retain 192, 128, or 64 visual tokens, corresponding to pruning ratios of 66.7%, 77.8%, and 88.9%, respectively. For LLaVA-NeXT, we retain 640, 320, or 160 visual tokens, corresponding to pruning ratios of 77.8%, 88.9%, and 94.4%. For Qwen2.5-VL, the main comparison uses retention ratios of 40%, 20%, and 10%, while the extended budget curve additionally includes 30%, 25%, 15%, 5%, 4%, 3%, and 2%. All budgets apply exclusively to visual tokens; textual and special tokens remain unchanged.

B.2 LEAF Implementation Settings

The hierarchy construction, question-aware probing, and selection procedures are detailed in Appendices A.2–A.4. Here, we report the implementation settings used in the experiments.

Image-driven hierarchy. We use the frozen `deit_small_distilled_patch16_224` checkpoint and execute its converted OpenVINO model on the CPU. The encoder receives a 336×336 image and produces a 21×21 grid of 384-dimensional patch features from the normalized output of its final transformer block. The class and distillation tokens are discarded. We use $K = 5$ PCA power iterations, the 5th and 95th percentiles for normalization, sharpening exponent $\gamma_s = 1.5$, and a 6×6 operational saliency grid. QTMT partitioning operates on this grid with $\alpha = 0.2$, $\rho = 4$, and $\tau = 0.55$. Each terminal region is spatially mapped to the corresponding host-VLM patch grid to obtain its original patch-index membership. The minimum leaf dimension is one operational-grid cell.

Question-aware probing. We use zero-based transformer-block index $\ell = 8$, corresponding to the ninth language-model block. Attention replay retains all attention heads and question queries and performs the softmax operation in FP32. Positional debiasing uses $\beta = 1.5$, and score normalization uses $\epsilon = 10^{-8}$. The positional template, debiasing operation, and normalization procedure are defined in Appendix A.3.

Leaf aggregation and selection. We use a shared signed Rademacher projection with dimension $d_r = 16$ and random seed 0. The same cached projection matrix is used for patch and leaf features, and projection and selection are executed on the CPU. Unless otherwise stated, we use heterogeneity exponent $\sigma = 1$, minimum detail scale $d_{\min} = 0$, relevance weight $\lambda = 1$, and repeated-allocation decay exponent $\gamma = 1$. Target-weighted marginal coverage is enabled. Leaf-level ties are resolved by the first maximal entry, while within-leaf candidates are ranked by descending patch relevance.

B.3 Evaluation Protocol, Baselines, and Datasets

Evaluator and inference configuration. The LLaVA experiments use a locally vendored LMMS-Eval implementation (Zhang et al., 2025a) whose original package version and Git revision were not retained. The Qwen2.5-VL experiments use LMMS-Eval v0.7.2 at commit `047ec52003590fd8f9540b8e9aabb82e889159cf`. All experiments use batch size 1, FP16 inference, KV caching, and deterministic greedy decoding with temperature 0, one beam, and sampling disabled. LLaVA uses the `vicuna_v1` conversation template and scaled dot-product attention when supported, with eager attention as a fallback. Qwen2.5-VL uses scaled dot-product attention and the system prompt "You are a helpful assistant."

Baseline implementations. We use the authors' official implementations of VisionZip (Yang et al., 2025), DivPrune (Alvar et al., 2025), and ZOO-Prune (Kim et al., 2026) with their recommended settings. We modify only the backbone interface and target visual-token budget when necessary. Each method uses the same frozen VLM backbone and evaluation configuration at a given budget.

Datasets and metrics. We evaluate the LLaVA models on nine benchmarks covering complementary multimodal capabilities. GQA (Hudson & Manning, 2019) evaluates compositional reasoning over real-world images, while VQAv2 (Goyal et al., 2017) measures general visual question answering. MMBench (Liu et al., 2024c) assesses broad visual perception and reasoning through multiple-choice questions, and MME (Fu et al., 2025) evaluates both perception and cognition. POPE (Li et al., 2023b) measures object hallucination, whereas TextVQA (Singh et al., 2019) focuses on understanding and reasoning about text embedded in images. ScienceQA-IMG (Lu et al., 2022) evaluates multimodal scientific reasoning, and MMMU (Yue et al., 2024) targets multidisciplinary reasoning over knowledge-intensive visual materials. Finally, SEED-I (Li et al., 2024) evaluates broad image comprehension across diverse visual domains. Qwen2.5-VL is evaluated on the GQA, MMBench, MME, and POPE subset. Table 6 summarizes the evaluated splits, primary metrics, and maximum generation lengths.

Table 6: Evaluation splits, primary metrics, and maximum generation lengths. P+C denotes the sum of perception and cognition scores.

Benchmark	Split	Metric	Max tokens
GQA	test-dev	Acc.	16
MMBench	dev	Acc.	1024
MME	test	P+C	16
POPE	test	F1	128
ScienceQA-IMG	test	Acc.	16
VQAv2	validation	Acc.	16
TextVQA	validation	Acc.	1024
MMMU	validation	Acc.	128
SEED-I	test	Acc.	1024

We use the testdev balanced-instructions configuration for GQA, the English configuration of MMBench, and the image-only configuration of ScienceQA. SEED-I denotes the image subset of the locally stored SEED-Bench dataset. The remaining benchmarks use their default LMMS-Eval configurations. Exact dataset identifiers, cached builder hashes, and evaluation settings are provided in the supplementary evaluation configuration files.

For GQA, MME, and VQAv2, the evaluator requests a single word or short phrase. MMBench, ScienceQA-IMG, and SEED-I use their formatted multiple-choice prompts and request the option letter directly. POPE uses the original binary question without an additional suffix. TextVQA includes the available OCR tokens and requests a single word or phrase. MMMU uses its official prompt construction, with image markers converted to evaluator image tokens. Generation terminates earlier when an end-of-sequence condition is reached.

Normalized average performance. We report accuracy except for POPE, which uses F1, and MME, which uses the sum of its perception and cognition scores. To aggregate benchmarks with different metric scales, we report normalized performance retention:

$$\text{Avg.} = \frac{100}{M} \sum_{m=1}^{M} \frac{p_m^{\text{comp}}}{p_m^{\text{full}}}, \tag{27}$$

where M is the number of evaluated benchmarks, and p_m^{comp} and p_m^{full} denote the compressed and full-token scores on benchmark m, respectively.

Random-retention reference. We construct the random-retention reference using the original frozen Qwen2.5-VL-7B model without importance scoring or a pruning method. After the vision encoder's spatial merger and before the visual tokens enter the language model, we uniformly sample visual tokens without replacement while preserving all textual and special tokens. For each nominal retention ratio $r \in \{2\%, 3\%, 4\%, 5\%, 10\%, 15\%, 20\%, 25\%, 30\%, 40\%\}$, the retained-token budget is

$$B_r = \text{round}(rN_v), \tag{28}$$

where N_v is the post-merger visual-token count. Consequently, the realized retention ratio may differ slightly from r. We independently evaluate random seeds 0, 1, and 42 on every benchmark and report the mean performance across the three runs. The resulting curve serves as the Qwen2.5-VL reference in the BD-Token calculation.

B.4 BD-TOKEN CALCULATION

BD-Token adapts the conventional Bjøntegaard-Delta rate (BD-Rate) metric (Bjontegaard, 2001) by replacing bitrate with the retained visual-token ratio and reconstruction quality with normalized VLM performance. For an evaluated method M and reference method R, we define

$$\text{BD-Token}(M;R)=\left[\exp\left(\frac{1}{A_H-A_L}\int_{A_L}^{A_H}[\log T_M(A)-\log T_R(A)]\,\mathrm{d}A\right)-1\right]\times 100\%, \tag{29}$$

where A denotes normalized average performance retention, $T_M(A)$ and $T_R(A)$ are the retained visual-token ratios required by M and R, respectively, to achieve performance A, and $[A_L, A_H]$ is the common performance interval covered by both curves. We use Qwen2.5-VL with random visual-token retention as R. A negative BD-Token value indicates that M requires fewer visual tokens than random retention at matched performance, with a more negative value representing greater average token savings.

For each method and the random-retention reference, we first sort the observations by retained-token ratio and construct a monotonically non-decreasing performance curve by replacing every performance value with the cumulative maximum over all smaller or equal retention ratios. If this operation produces duplicate performance values, we retain the smallest token ratio achieving each value. For each method–reference pair, the integration interval is their overlapping performance range:

$$A_L=\max(\min\mathbf{A}_M,\min\mathbf{A}_R)\,,\qquad A_H=\min(\max\mathbf{A}_M,\max\mathbf{A}_R)\,, \tag{30}$$

where $\mathbf{A}_M$ and $\mathbf{A}_R$ denote the processed observed performance values of method M and reference R, respectively.

We then fit separate piecewise cubic Hermite interpolants (PCHIP) to the natural logarithm of the retained-token ratio as a function of normalized performance. The two interpolants are evaluated at $2{,}000$ uniformly spaced points over $[A_L, A_H]$, and their difference is numerically integrated using Simpson's rule. The calculation uses retention ratios of 2%, 3%, 4%, 5%, 10%, 15%, 20%, 25%, 30%, and 40%. Each method is integrated over its own overlapping performance interval with the random-retention reference, and the resulting BD-Token value is reported to two decimal places.

B.5 ABLATION CONFIGURATIONS

The ablations reported in Table 4 use LLaVA-1.5-7B with 128 retained visual tokens and the evaluation protocol described in Appendix B. Each variant changes only the component specified below; all remaining applicable model, hierarchy, probing, projection, selection, and evaluation settings are identical to those of full LEAF.

Necessity of each stage. For *w/o S1*, we bypass the auxiliary saliency encoder and QTMT construction and replace the adaptive hierarchy with the native 24×24 LLaVA patch grid:

$$\mathcal{L}=\big\{\{1\},\{2\},\ldots,\{576\}\big\}. \tag{31}$$

Each visual patch therefore forms a singleton leaf, while question-aware relevance and the complete hierarchy-aware selector remain active. This variant removes adaptive regional organization without replacing it with uniform coarse regions or a single global region.

For *w/o S2*, we skip question-aware probing and assign identical relevance to every visual patch:

$$q_i=1,\qquad i=1,\ldots,576. \tag{32}$$

Max aggregation consequently gives every leaf the same relevance. The QTMT hierarchy, projected leaf similarity, marginal coverage, heterogeneity, and detail decay remain unchanged, making selection independent of the input question.

For *w/o S3*, we retain both the QTMT hierarchy and question-aware patch scores but replace the complete selector with grouped leaf-importance greedy selection. We assign each leaf L_j the score

$$r_j=\max_{i\in L_j} q_i, \tag{33}$$

sort leaves by decreasing r_j, and rank patches within each leaf by decreasing q_i. The algorithm exhausts the patches of each higher-ranked leaf before proceeding to the next leaf and stops after selecting 128 patches. This is a grouped leaf-relevance baseline: it exhausts higher-ranked leaves before lower-ranked ones and therefore differs from global patch-level top-k selection.

Selector mechanism. The selector ablations retain the same QTMT hierarchy, question-aware relevance, projected features, token budget, and deterministic candidate ordering as full LEAF. We use the notation defined in Appendix A.4.

The *Coverage only* variant sets the patch-detail weight to $\lambda = 0$. Its inactive-leaf gain therefore contains only target-weighted marginal coverage, while the active-leaf gain becomes zero. Question relevance remains present through the target-leaf weights; only the patch-level detail term, and consequently its heterogeneity and decay factors, is disabled.

The *Source-weighted coverage* variant replaces the target-weighted coverage gain

$$\Delta C_j = \sum_u w_u \left[s_{ju} - c_u \right]_+ \tag{34}$$

with

$$\Delta C_j^{\mathrm{src}} = w_j \sum_u \left[s_{ju} - c_u \right]_+ , \tag{35}$$

where w_u is the relevance of target leaf u, w_j is the relevance of candidate leaf j, s_{ju} is their projected-feature similarity, and c_u is the current coverage of u. Full LEAF therefore weights each newly covered target by its relevance, whereas this variant weights the candidate leaf's total unweighted coverage by the candidate's own relevance. All other gain terms remain unchanged.

The *No detail decay* variant sets the repeated-allocation decay exponent to $\gamma = 0$, making its decay factor equal to one. Repeated selection from an active leaf is consequently not progressively discounted. The *No heterogeneity* variant sets the implemented heterogeneity exponent to $\sigma = 0$, producing the same detail scale $d_j = 1$ for every leaf. Heterogeneity is still computed but no longer affects allocation. All selected patch indices are restored to their original visual-token order before the compressed sequence is constructed.

JL-Rademacher projection dimension. We vary the projection dimension over $d_r \in \{4, 8, 16, 32, 64\}$, with $d_r = 16$ as the default. For every dimension, the projection is a fixed signed Rademacher matrix whose entries are independently drawn from $\{-1/\sqrt{d_r}, +1/\sqrt{d_r}\}$ using seed 0. The same matrix is applied to patch and leaf features, and all hierarchy, relevance, and selector settings remain unchanged.

Saliency feature source. We compare pixel gradients, DINOv2 features, host-VLM visual features, and the default DeiT features as the source for constructing the image-driven saliency map. For each variant, the resulting response is aligned to the same 6×6 operational grid and passed to the unchanged QTMT hierarchy construction. All question-aware probing and selection settings remain fixed.

C Additional Experimental Results

C.1 Complete Performance Comparisons

Tables 7, 8, and 9 report the complete results across all evaluated token budgets. The main paper presents the lower primary budgets to emphasize performance under aggressive compression.

Table 7: **Performance comparison on LLaVA-1.5-7B across visual-token budgets.** Avg. denotes mean benchmark-wise performance relative to the full-token model. Bold indicates the best compressed result at each budget.

Method	GQA Acc.↑	MMB Acc.↑	MME P+C↑	POPE F1↑	SQA Acc.↑	VQA^{v2} Acc.↑	VQA^{Text} Acc.↑	MMMU Acc.↑	SEED-I Acc.↑	Avg. ↑
Total 576 Visual Tokens										
LLaVA-1.5-7B	61.90	64.70	1862.00	85.90	69.50	78.50	58.20	36.30	58.60	100.00%
Retain 192 Tokens ↓ 66.7%										
VisionZip	59.30	63.00	1782.60	85.30	68.90	76.80	57.30	**36.60**	56.40	97.90%
DivPrune	59.97	62.54	1762.23	87.00	68.91	**76.87**	56.97	35.44	58.71	98.00%
ZOO-Prune	60.04	63.02	1781.83	**87.07**	**69.06**	76.83	57.26	35.56	58.89	98.39%
LEAF (Ours)	**60.22**	**63.75**	**1804.05**	86.49	68.12	75.26	**57.44**	35.89	**59.07**	**98.41%**
Retain 128 Tokens ↓ 77.8%										
VisionZip	57.60	62.00	1761.70	83.20	68.90	75.60	56.80	**37.90**	54.90	96.80%
DivPrune	59.25	62.03	1718.22	86.72	**68.96**	**75.96**	56.06	35.56	56.98	96.90%
ZOO-Prune	59.32	61.77	1745.44	**87.24**	68.86	75.84	**56.91**	35.56	57.40	97.33%
LEAF (Ours)	**59.33**	**63.14**	**1764.96**	86.37	68.32	74.58	56.81	36.11	**58.17**	**97.60%**
Retain 64 Tokens ↓ 88.9%										
VisionZip	55.10	60.10	1690.00	77.00	69.00	72.40	55.50	**36.20**	52.20	93.10%
DivPrune	57.78	59.28	1674.40	85.56	68.17	**74.11**	54.69	35.56	55.13	94.80%
ZOO-Prune	**58.32**	60.39	1675.05	85.80	68.12	73.01	55.47	35.11	55.48	95.01%
LEAF (Ours)	58.24	**62.11**	**1703.62**	**86.21**	**69.06**	73.48	**56.27**	35.44	**57.28**	**96.33%**

Table 8: **Performance comparison on LLaVA-NeXT-7B across visual-token budgets.** Bold indicates the best compressed result at each budget.

Method	GQA Acc.↑	MMB Acc.↑	MME P+C↑	POPE F1↑	SQA Acc.↑	VQA^{v2} Acc.↑	VQA^{Text} Acc.↑	MMMU Acc.↑	SEED-I Acc.↑	Avg. ↑
Total up to 2,880 Visual Tokens										
LLaVA-NeXT-7B	64.20	67.90	1842.00	86.40	70.20	80.10	61.30	35.10	70.20	100.00%
Retain 640 Tokens ↓ 77.8%										
VisionZip	61.30	**66.30**	1787.00	86.30	68.10	79.10	**60.20**	34.70	66.70	97.54%
DivPrune	61.58	65.38	1773.04	85.51	67.82	78.94	55.41	36.89	67.56	97.15%
ZOO-Prune	62.19	65.21	**1816.45**	86.75	68.02	**79.64**	57.98	36.89	67.95	98.30%
LEAF (Ours)	**62.97**	66.15	1811.21	**87.47**	**69.26**	78.09	59.07	**37.67**	**69.06**	**99.25%**
Retain 320 Tokens ↓ 88.9%										
VisionZip	59.30	63.10	1702.00	82.10	67.30	76.20	58.90	35.30	63.40	94.52%
DivPrune	59.63	63.66	1731.04	83.47	67.82	76.64	53.84	**37.11**	65.35	95.13%
ZOO-Prune	61.07	64.86	1762.67	85.27	67.18	**78.08**	57.51	37.00	66.32	96.88%
LEAF (Ours)	**61.16**	**65.55**	**1779.27**	**86.98**	**68.42**	76.82	**59.05**	**37.11**	**67.54**	**97.86%**
Retain 160 Tokens ↓ 94.4%										
VisionZip	55.50	60.10	1630.00	74.80	**68.30**	71.40	56.20	36.10	58.30	90.45%
DivPrune	57.79	62.29	1658.25	79.36	68.02	73.92	52.42	**36.44**	62.54	92.36%
ZOO-Prune	**59.66**	63.83	1722.01	83.04	67.92	**76.12**	55.55	35.89	64.32	94.76%
LEAF (Ours)	59.42	**65.55**	**1752.12**	**86.69**	68.17	75.18	**56.73**	36.11	**65.56**	**96.04%**

Table 9: **Detailed Qwen2.5-VL-7B comparison across visual-token retention ratios.** Random retention is averaged over three seeds.

Method	GQA Acc.↑	MMB Acc.↑	MME P+C↑	POPE F1↑	Avg. ↑
Full Dynamic-Resolution Visual Tokens					
Qwen2.5-VL-7B	60.84	84.10	2310	86.30	100.00%
Retain 40% of Visual Tokens ↓ 60.0%					
Random retention	57.25	79.63	2227	84.43	95.76%
VisionZip	57.31	79.92	2258	85.91	96.63%
DivPrune	60.03	**82.04**	2271	85.61	98.43%
ZOO-Prune	**60.36**	81.36	2253	86.38	98.39%
LEAF (Ours)	59.41	81.96	**2309**	**86.52**	**98.83%**
Retain 30% of Visual Tokens ↓ 70.0%					
Random retention	58.46	78.64	2226	83.02	95.54%
VisionZip	59.01	78.52	2234	84.67	96.29%
DivPrune	59.42	81.01	2199	84.85	96.88%
ZOO-Prune	**59.81**	80.15	2217	86.21	97.37%
LEAF (Ours)	59.06	**81.96**	**2282**	**86.67**	**98.44%**
Retain 25% of Visual Tokens ↓ 75.0%					
Random retention	57.97	77.43	2177	82.07	94.17%
VisionZip	58.48	79.98	2192	83.98	95.86%
DivPrune	59.02	80.33	2201	84.74	96.50%
ZOO-Prune	**59.58**	78.69	2151	85.94	96.05%
LEAF (Ours)	58.94	**81.96**	**2254**	**86.71**	**98.10%**
Retain 20% of Visual Tokens ↓ 80.0%					
Random retention	57.12	76.06	2144	81.07	92.77%
VisionZip	57.27	79.72	2221	83.89	95.57%
DivPrune	**60.05**	79.55	2173	83.42	96.01%
ZOO-Prune	58.81	80.60	2201	84.17	96.33%
LEAF (Ours)	58.44	**81.27**	**2257**	**86.71**	**97.72%**
Retain 15% of Visual Tokens ↓ 85.0%					
Random retention	55.79	74.23	2086	78.60	90.34%
VisionZip	57.70	74.31	1963	81.13	90.55%
DivPrune	57.58	78.26	2131	82.16	93.79%
ZOO-Prune	**58.28**	77.32	2068	85.07	93.96%
LEAF (Ours)	58.15	**80.93**	**2247**	**87.00**	**97.47%**

Method	GQA Acc.↑	MMB Acc.↑	MME P+C↑	POPE F1↑	Avg. ↑
Full Dynamic-Resolution Visual Tokens					
Qwen2.5-VL-7B	60.84	84.10	2310	86.30	100.00%
Retain 10% of Visual Tokens ↓ 90.0%					
Random retention	53.51	70.25	1917	75.36	85.45%
VisionZip	54.09	76.03	1937	78.97	88.67%
DivPrune	55.49	76.03	2054	79.05	90.53%
ZOO-Prune	55.45	76.28	2018	80.99	90.76%
LEAF (Ours)	**57.22**	**78.52**	**2219**	**87.02**	**96.08%**
Retain 5% of Visual Tokens ↓ 95.0%					
Random retention	49.00	63.80	1773	67.72	77.91%
VisionZip	49.69	65.81	1798	68.78	79.36%
DivPrune	51.16	67.70	1891	70.95	82.17%
ZOO-Prune	50.71	67.10	1775	76.76	82.23%
LEAF (Ours)	**55.18**	**75.26**	**2107**	**86.28**	**92.84%**
Retain 4% of Visual Tokens ↓ 96.0%					
Random retention	47.49	60.77	1769	64.51	75.42%
VisionZip	46.84	65.27	1658	74.61	78.21%
DivPrune	49.56	67.97	1756	76.45	81.72%
ZOO-Prune	48.21	66.57	1783	74.61	80.51%
LEAF (Ours)	**54.08**	**73.97**	**2043**	**85.82**	**91.18%**
Retain 3% of Visual Tokens ↓ 97.0%					
Random retention	45.53	55.87	1682	59.83	70.86%
VisionZip	46.53	59.73	1559	60.55	71.29%
DivPrune	46.81	60.91	1652	59.65	72.50%
ZOO-Prune	45.49	58.85	1593	68.27	73.20%
LEAF (Ours)	**52.43**	**72.16**	**1959**	**84.38**	**88.64%**
Retain 2% of Visual Tokens ↓ 98.0%					
Random retention	42.22	49.08	1574	51.06	63.77%
VisionZip	40.17	50.38	1562	56.43	64.73%
DivPrune	43.71	52.66	1538	49.19	64.51%
ZOO-Prune	41.23	52.41	1509	55.83	65.03%
LEAF (Ours)	**49.46**	**68.91**	**1852**	**79.71**	**83.94%**

C.2 Detailed Results on LLaVA-1.5-13B

Table 10 reports the complete benchmark-level comparison on LLaVA-1.5-13B. LEAF achieves the highest normalized average performance at all three evaluated budgets, retaining 98.76%, 98.30%, and 97.15% of full-token performance with 192, 128, and 64 visual tokens, respectively. These results are consistent with those obtained using LLaVA-1.5-7B and demonstrate that LEAF's effectiveness is preserved when scaling the language-model backbone from 7B to 13B parameters.

Table 10: **Performance comparison on LLaVA-1.5-13B across visual-token budgets.** Avg. denotes mean benchmark-wise performance relative to the full-token model. Bold indicates the best compressed result at each budget.

Method	GQA Acc.↑	MMB Acc.↑	MME P+C↑	POPE F1↑	SQA Acc.↑	VQA^{v2} Acc.↑	VQA^{Text} Acc.↑	MMMU Acc.↑	SEED-I Acc.↑	Avg. ↑
Total 576 Visual Tokens										
LLaVA-1.5-13B	63.20	67.70	1818.00	85.90	72.80	80.00	61.30	36.40	66.90	100.00%
Retain 192 Tokens ↓ 66.7%										
VisionZip	59.10	66.90	1754.00	85.10	73.50	78.10	**59.50**	36.40	65.20	97.89%
DivPrune	59.42	66.58	**1781.50**	**86.76**	72.88	77.98	58.46	36.56	65.72	98.11%
ZOO-Prune	59.95	66.67	1762.41	86.73	73.12	**78.65**	59.11	**37.33**	65.56	98.55%
LEAF (Ours)	**60.29**	**68.13**	1776.72	86.71	**73.53**	76.61	59.14	37.11	**66.17**	**98.76%**
Retain 128 Tokens ↓ 77.8%										
VisionZip	57.90	66.70	1743.00	85.20	**74.00**	76.80	58.70	36.10	63.80	97.02%
DivPrune	58.89	66.07	1748.56	86.53	72.83	77.10	58.17	35.56	64.22	96.96%
ZOO-Prune	58.89	67.01	**1791.10**	**86.95**	73.38	**77.83**	**58.80**	35.56	64.50	97.78%
LEAF (Ours)	**59.97**	**67.87**	1777.80	86.68	73.57	75.93	58.41	**37.22**	**65.15**	**98.30%**
Retain 64 Tokens ↓ 88.9%										
VisionZip	56.20	64.90	1676.00	76.00	**74.40**	73.70	57.40	**36.40**	60.40	93.74%
DivPrune	57.66	64.60	1777.93	84.80	71.34	75.20	57.11	35.22	62.44	95.38%
ZOO-Prune	58.58	64.78	**1780.03**	85.34	72.09	**76.39**	**58.59**	36.00	63.02	96.54%
LEAF (Ours)	**59.14**	**66.58**	1774.69	**86.55**	72.63	75.14	57.81	36.22	**64.57**	**97.15%**

C.3 Efficiency Measurement Details

We evaluate the efficiency–performance trade-off on LLaVA-1.5-7B using MMBench. Table 11 reports the exact measurements underlying Figure 7, including MMBench accuracy, average prefill latency, end-to-end evaluation time, and estimated computation at each visual-token budget.

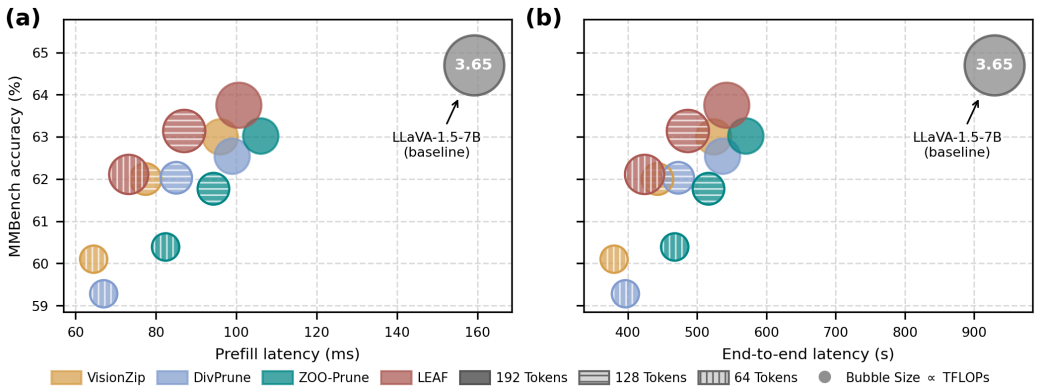

Figure 7: **Efficiency–performance trade-off on LLaVA-1.5-7B.** MMBench accuracy versus **(a)** average prefill latency and **(b)** end-to-end evaluation time at three retained-token budgets. Colors identify methods, hatching identifies budgets, and marker area is proportional to TFLOPs. The gray marker denotes the full-token backbone.

Table 11: **Detailed efficiency comparison on LLaVA-1.5-7B.** Prefill latency is averaged over MMBench examples, whereas end-to-end time measures the complete benchmark evaluation. Bold denotes the best compressed result for each metric at a given budget.

Method	Retained tokens	MMBench Acc. (↑)	Prefill latency (ms, ↓)	End-to-end time (s, ↓)	TFLOPs (↓)
LLaVA-1.5-7B	576	64.70	159.14	929.08	3.65
VisionZip	192	63.00	96.02	523.93	1.62
DivPrune	192	62.54	98.98	536.71	1.62
ZOO-Prune	192	63.02	106.09	570.34	1.63
LEAF	192	63.75	100.64	542.81	2.65
VisionZip	128	62.00	77.43	442.83	1.29
DivPrune	128	62.03	85.07	472.82	1.29
ZOO-Prune	128	61.77	94.32	516.45	1.30
LEAF	128	63.14	87.05	486.69	2.31
VisionZip	64	60.10	64.51	380.21	0.96
DivPrune	64	59.28	67.01	396.53	0.96
ZOO-Prune	64	60.39	82.41	467.67	0.97
LEAF	64	62.11	73.24	424.17	1.99

Timing protocol. All methods are measured using the same LLaVA-1.5-7B checkpoint on MMBench, with a batch size of 1, FP16 inference, KV caching, and greedy decoding, as described in Appendix B.3. Measurements are conducted on a single NVIDIA A800-SXM4-80GB GPU with an Intel Xeon Platinum 8336C CPU at 2.30 GHz. Each method is evaluated using one complete pass over the 4,329 MMBench samples, without explicit warm-up iterations. Prefill latency is measured using CUDA events around the model generation call with `max_new_tokens=1`, covering multimodal input preparation on the GPU, the prefilling forward pass, and the generation of the first output token. We report the mean latency across all MMBench examples. CUDA synchronization is applied before reading the elapsed CUDA-event time. End-to-end latency is measured as the wall-clock duration of the complete evaluation run. Dataset loading and metric aggregation are excluded from the reported prefilling latency but included in the end-to-end latency.

Parallel LEAF execution. For LEAF, timing includes image-driven hierarchy construction, question-aware probing, leaf-level aggregation and selection, and the separate full forward pass over the compressed multimodal sequence. The image-driven hierarchy runs on the CPU while the question-aware probe runs on the GPU. The branches run concurrently and synchronize before leaf-level selection, so their execution times partially overlap.

FLOPs calculation. Let N_v and B denote the original and retained visual-token counts, respectively, and let N_t denote the unchanged number of textual and special tokens. The full and compressed multimodal sequence lengths are therefore

$$N_{\text{full}} = N_t + N_v, \qquad N_B = N_t + B. \tag{36}$$

Following prior visual-token pruning work (Kim et al., 2026), the prefilling cost of an L-layer language model with sequence length N, hidden dimension d, and feed-forward intermediate dimension m is approximated as

$$F_{\text{LM}}(N; L) = L\left(4Nd^2 + 2N^2d + 2Ndm\right), \tag{37}$$

where the three terms correspond to attention projections, attention-map computation, and feed-forward layers, respectively. The cost of the uncompressed model is

$$F_{\text{full}} = F_{\text{vis}} + F_{\text{proj}} + F_{\text{LM}}(N_{\text{full}}; L), \tag{38}$$

where F_{vis} and F_{proj} denote the costs of the host vision encoder and multimodal projector.

For a conventional compression method M, the total cost at budget B is

$$F_M(B) = F_{\text{vis}} + F_{\text{proj}} + F^M_{\text{score}} + F^M_{\text{select}} + F_{\text{LM}}(N_B; L), \tag{39}$$

where F^M_{score} and F^M_{select} account for method-specific token scoring and selection.

LEAF additionally constructs an image-driven hierarchy and executes the full multimodal sequence through the probing blocks. Because ℓ is a zero-based probe-block index, probing through block ℓ evaluates $\ell + 1$ language-model blocks:

$$F_{\text{probe}} = (\ell + 1)\left(4N_{\text{full}}d^2 + 2N^2_{\text{full}}d + 2N_{\text{full}}dm\right). \tag{40}$$

The total LEAF cost is therefore

$$F_{\text{LEAF}}(B) = F_{\text{vis}} + F_{\text{proj}} + F_{\text{hier}} + F_{\text{probe}} + F_{\text{replay}} + F^{\text{LEAF}}_{\text{select}} + F_{\text{LM}}(N_B; L), \tag{41}$$

where F_{hier} covers the auxiliary DeiT-S encoder, PCA-based saliency estimation, and QTMT construction; F_{replay} covers question-to-visual attention replay; and $F^{\text{LEAF}}_{\text{select}}$ covers feature projection, leaf aggregation, and greedy allocation. The final term contains all L language-model layers because the selected sequence undergoes a separate complete forward pass rather than resuming from the probing block.

FLOPs sum all executed operations and therefore do not account for concurrent execution. In wall-clock time, LEAF's image-driven and question-aware branches run on the CPU and GPU, respectively, so their combined evidence-extraction latency is approximately

$$T_{\text{evidence}} \approx \max(T_{\text{hier}}, T_{\text{probe}}) + T_{\text{select}}, \tag{42}$$

rather than $T_{\text{hier}} + T_{\text{probe}} + T_{\text{select}}$. This explains why LEAF's higher operation count does not produce a proportional increase in measured latency.

TFLOPs estimate the total arithmetic cost per evaluated input, including the shared visual processing, method-specific scoring or probing, and language-model computation over the retained sequence. Unlike measured latency, this operation count does not represent the execution schedule or CPU–GPU overlap. LEAF therefore has a higher operation count, primarily because of its question-aware probe. Because the CPU and GPU branches overlap, the increase in measured latency is smaller than the increase in FLOPs.

C.4 Additional Qualitative Results

We provide additional qualitative comparisons on GQA (Figure 8), MME (Figure 9), POPE (Figure 10), ScienceQA-IMG (Figure 11), VQAv2 (Figure 12), TextVQA (Figure 13), MMMU (Figure 14), and SEED-I (Figure 15). Together with the MMBench examples in Figure 3, these results cover all nine benchmarks evaluated with LLaVA-1.5-7B. Each figure compares VisionZip, DivPrune, ZOO-Prune, and LEAF using 128 and 64 retained visual tokens. Correct and incorrect predictions are shown in green and red, respectively.

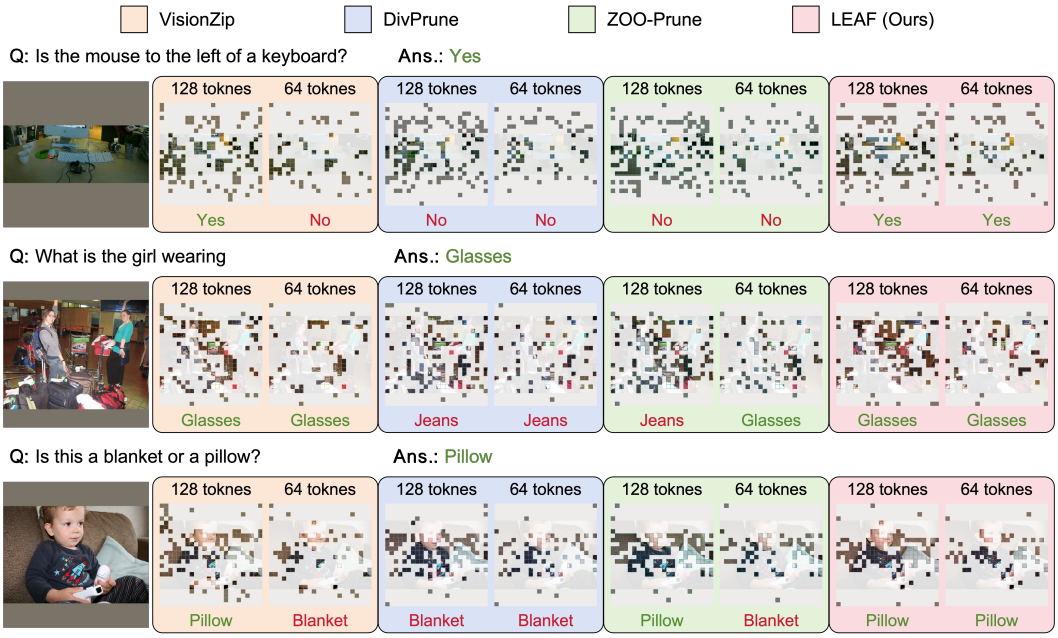

Figure 8: **Qualitative comparison on GQA with LLaVA-1.5-7B.** Results are shown with 128 and 64 retained visual tokens.

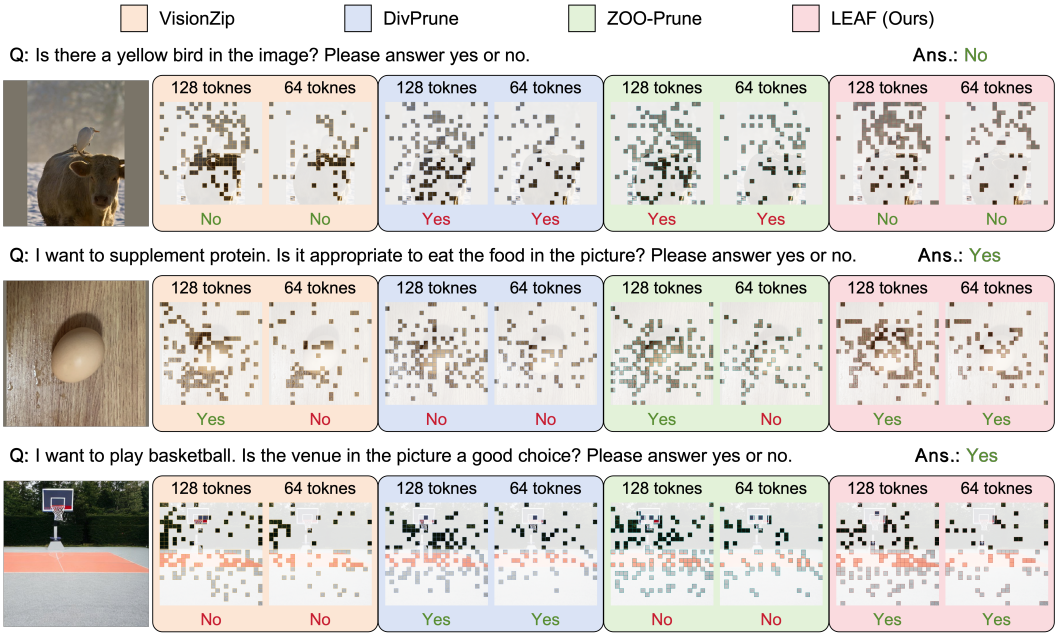

Figure 9: **Qualitative comparison on MME with LLaVA-1.5-7B.** Results are shown with 128 and 64 retained visual tokens.

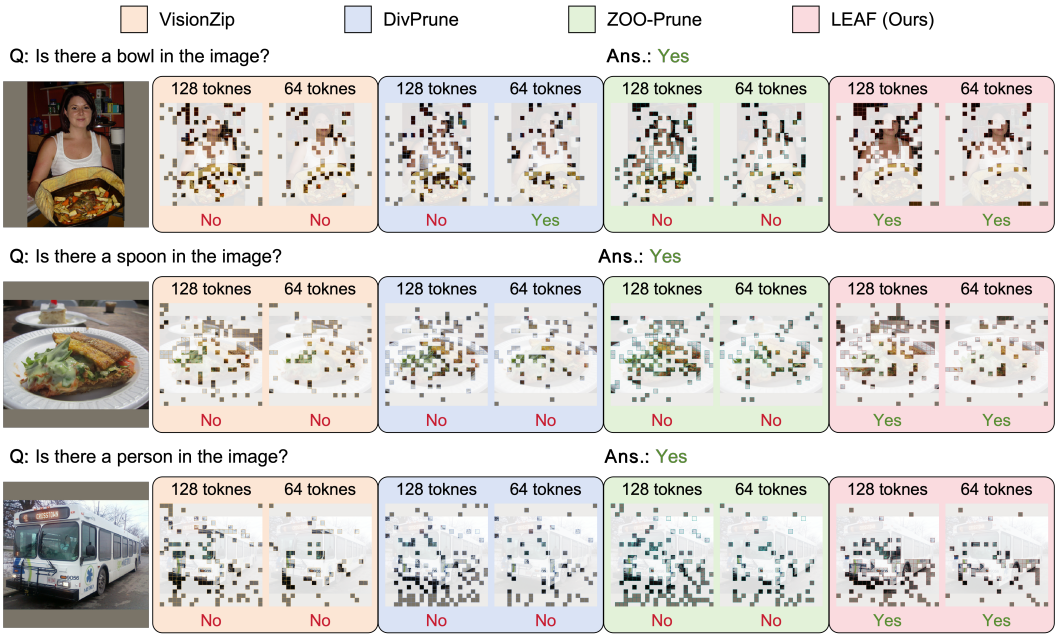

Figure 10: **Qualitative comparison on POPE with LLaVA-1.5-7B.** Results are shown with 128 and 64 retained visual tokens.

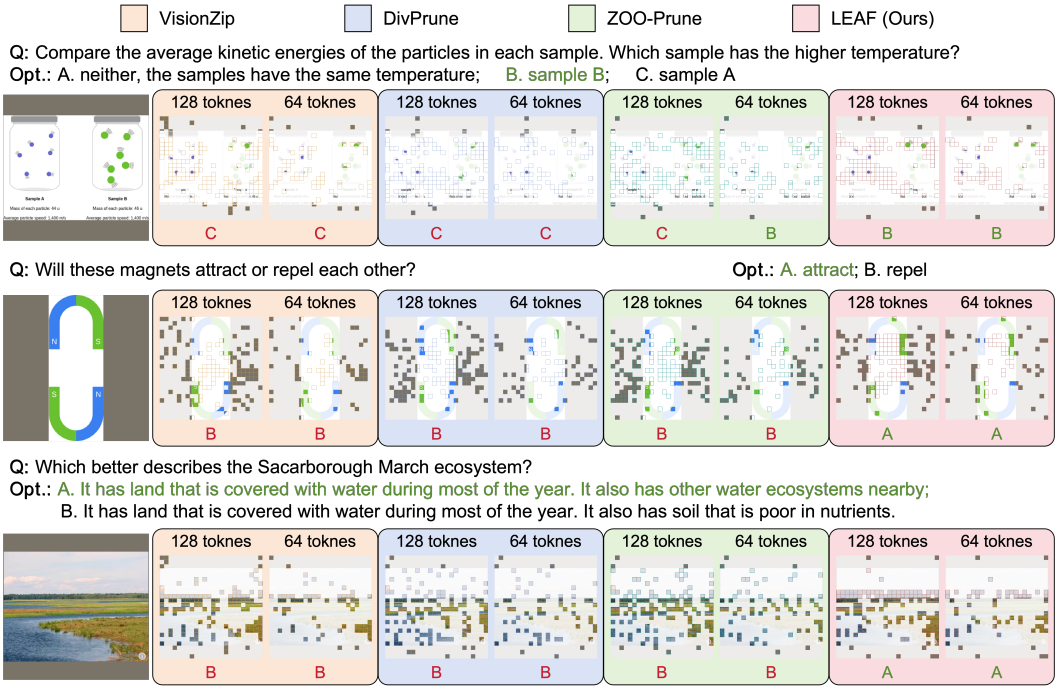

Figure 11: **Qualitative comparison on ScienceQA-IMG with LLaVA-1.5-7B.** Results are shown with 128 and 64 retained visual tokens.

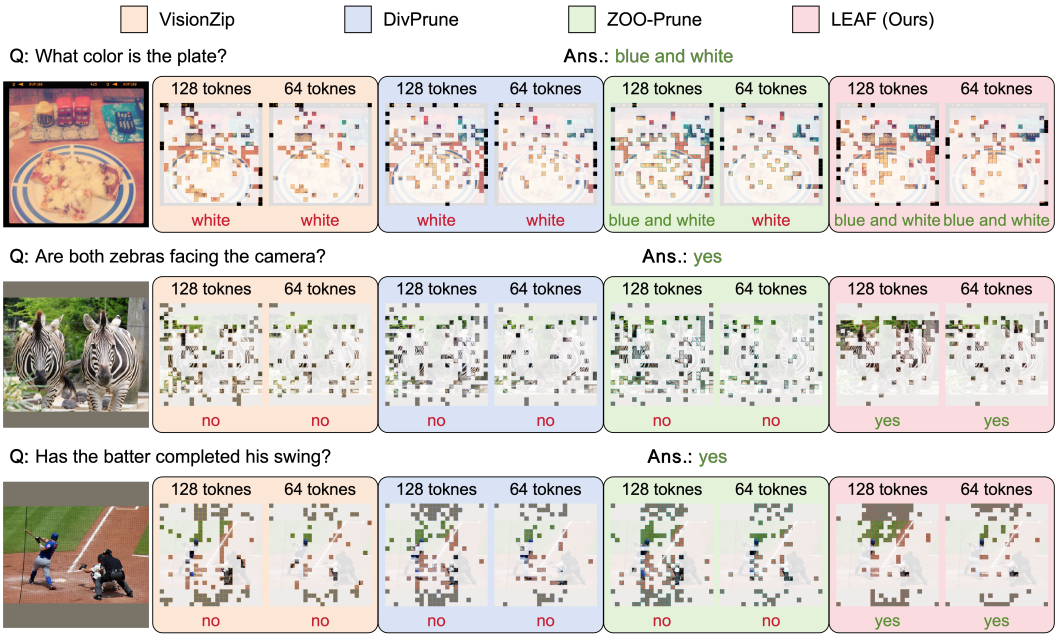

Figure 12: **Qualitative comparison on VQAv2 with LLaVA-1.5-7B.** Results are shown with 128 and 64 retained visual tokens.

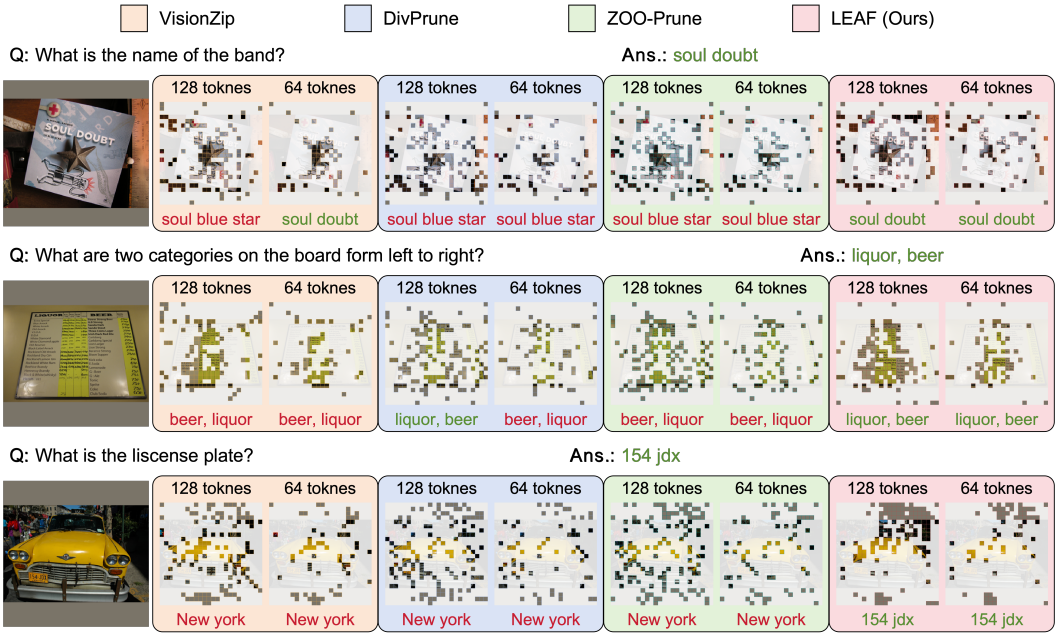

Figure 13: **Qualitative comparison on TextVQA with LLaVA-1.5-7B.** Results are shown with 128 and 64 retained visual tokens.

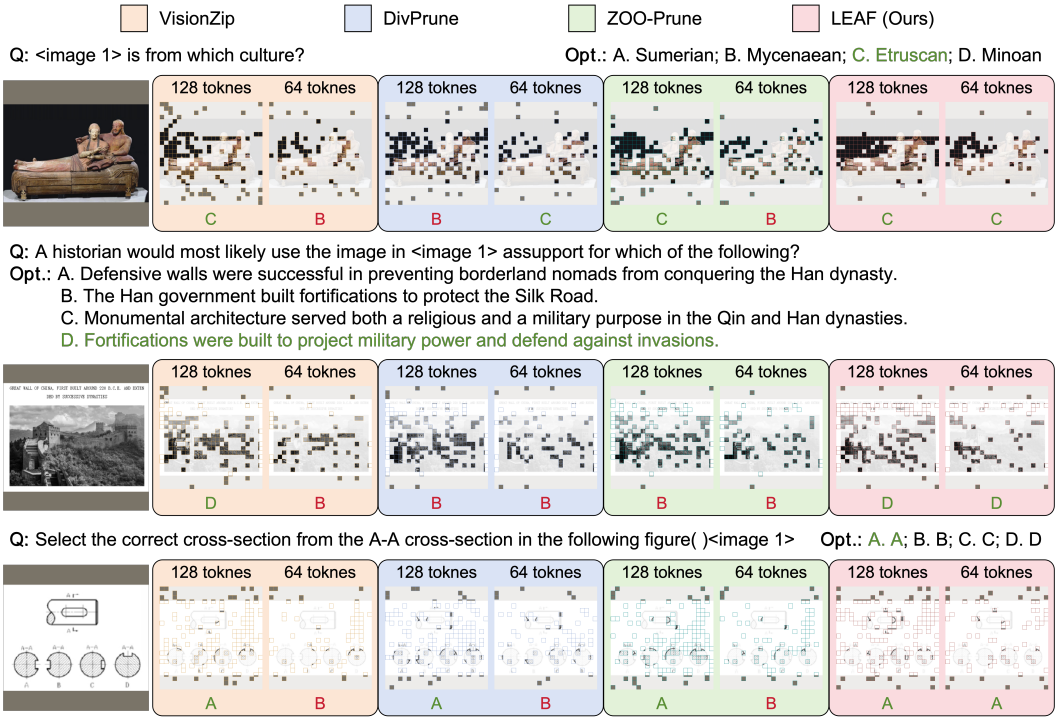

Figure 14: **Qualitative comparison on MMMU with LLaVA-1.5-7B.** Results are shown with 128 and 64 retained visual tokens.

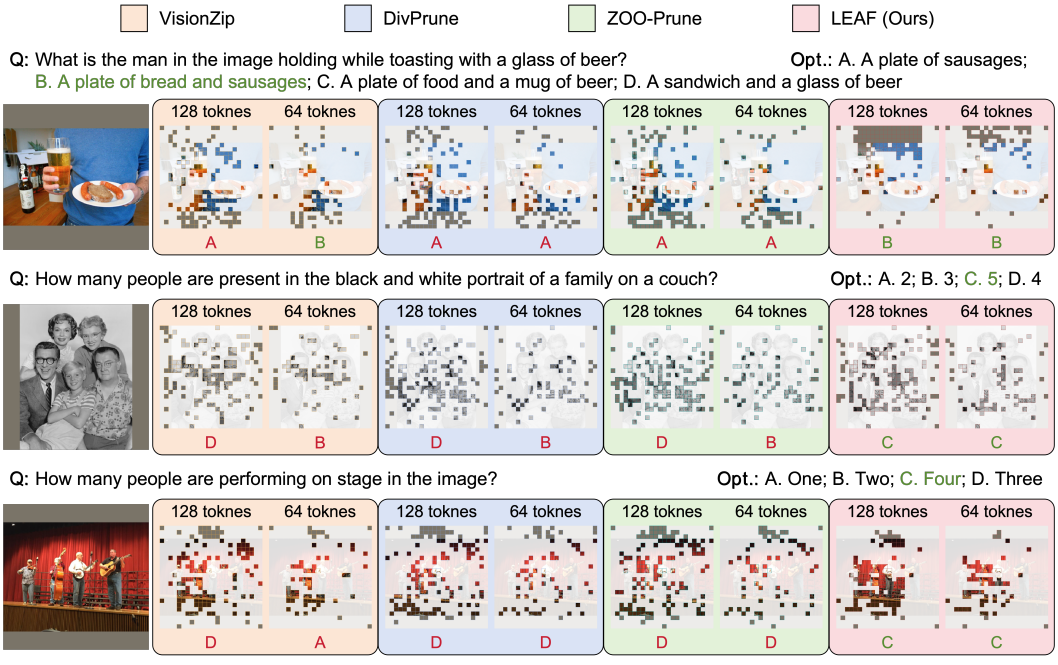

Figure 15: **Qualitative comparison on SEED-I with LLaVA-1.5-7B.** Results are shown with 128 and 64 retained visual tokens.

C.5 Additional Ablation Results

QTMT split types. To assess whether the different partition types in QTMT are necessary, we restrict the hierarchy constructor to progressively different split families. QT only permits equal four-way partitioning; QT + binary and QT + ternary additionally permit directional two-way and 1:2:1 three-way splits, respectively; full QTMT adaptively chooses among all three families according to regional saliency statistics. As shown in Table 12, QTMT achieves the best GQA and POPE results and ties QT + binary on MMBench and MME. Relative to QT only, it improves GQA by 0.34, MMBench by 0.77, MME by 12.83, and POPE by 0.55 points. Binary splitting is particularly effective on MMBench and MME, whereas adding ternary splitting produces a stronger POPE result than binary splitting alone. The results favor adaptive selection among quadtree, binary, and ternary partitions over a fixed split geometry.

Table 12: **Ablation of QTMT split types on LLaVA-1.5-7B with 128 retained visual tokens.** Binary and ternary denote the corresponding directional split families. All other settings remain unchanged. † marks the default configuration, and the best result in each column is bolded.

Variant	GQA	MMB	MME	POPE
QT only	58.99	62.37	1752.13	85.82
QT + binary	59.23	**63.14**	**1764.96**	85.88
QT + ternary	59.17	62.61	1755.19	86.11
QTMT†	**59.33**	**63.14**	**1764.96**	**86.37**

Attention probe layer. We evaluate single-layer probing at layers 4, 8, 16, and 24, together with a multi-layer variant that combines evidence from layers 8, 16, and 24. For a single probe layer ℓ, we average question-to-visual attention over all heads and question-token positions to obtain a patch score vector $\mathbf{a}^{(\ell)} \in \mathbb{R}^{576}$. The multi-layer variant independently extracts these vectors at layers 8, 16, and 24 and combines them by elementwise arithmetic averaging:

$$\mathbf{a}_{\text{multi}} = \frac{1}{3}\left(\mathbf{a}^{(8)} + \mathbf{a}^{(16)} + \mathbf{a}^{(24)}\right). \tag{43}$$

Positional debiasing is applied after this cross-layer aggregation. All probe variants retain every attention head and otherwise use the same question-aware scoring procedure. As shown in Table 13, probing at layer 4 produces the weakest results, indicating that early hidden representations provide insufficient question-conditioned visual evidence. Moving to layer 8 yields substantial improvements, including 33.74 points on MME and 4.02 points on POPE. Later probing further improves GQA, MMBench, and MME, with layer 24 achieving the best scores on these three benchmarks. The multi-layer variant obtains the highest POPE score but does not consistently outperform layer 24, despite using evidence from multiple depths. We use layer 8 because it improves all four benchmarks substantially over layer 4 while avoiding the additional full-sequence blocks required by layers 16 and 24. The later layers improve several scores, but incur correspondingly greater probing cost.

Table 13: **Ablation of the question-aware probe layer on LLaVA-1.5-7B with 128 retained visual tokens.** Layer indices are zero-based, and the multi-layer variant uses evidence from layers 8, 16, and 24. All other settings remain unchanged. † marks the default configuration, and the best result in each column is bolded.

Probe configuration	GQA	MMB	MME	POPE
Layer 4	57.96	62.63	1731.22	82.35
Layer 8†	59.33	63.14	1764.96	86.37
Layer 16	59.75	63.57	1771.72	86.27
Layer 24	**60.03**	**63.83**	**1802.89**	86.29
Layers 8+16+24	59.98	63.23	1798.77	**86.49**

Selector feature reduction. We compare the fixed JL-Rademacher projection with coordinate subsampling, PCA, variance-based coordinate selection, max pooling, and selection in the original

full-dimensional feature space. The first four alternatives represent different ways of constructing a 16-dimensional selector representation, whereas Full dimension disables reduction entirely. All reduced variants map patch and leaf features from dimension D to $d_r = 16$, applying the same transformation to both feature types. Random coordinate uses a seed-0 permutation of the original feature dimensions and retains its first 16 coordinates; the resulting coordinate subset is fixed across evaluation samples. PCA is fitted independently for each sample using its 576 patch features. After centering by the sample mean, we retain the first 16 right-singular vectors and apply the same sample-specific center and basis to the leaf features. Top variance similarly operates per sample but selects the 16 original coordinates with the largest population variance across the patch features. Max pooling treats each feature vector as a one-dimensional sequence and applies adaptive one-dimensional max pooling to produce exactly 16 outputs. JL-Rademacher uses the fixed seeded signed projection defined in Appendix A.4, while Full dimension performs similarity and heterogeneity calculations without feature reduction. As shown in Table 14, random coordinate selection performs worst on all four benchmarks, indicating that arbitrary feature subsampling does not adequately preserve the geometry required for leaf similarity and heterogeneity estimation. Full-dimensional features achieve the highest GQA and MMBench scores, while PCA performs best on POPE. At $d_r = 16$, JL-Rademacher is 0.23 and 0.26 points below full-dimensional selection on GQA and MMBench, respectively, while achieving the best MME score. It also avoids both the per-sample basis construction required by PCA and full-dimensional similarity computation. We therefore use the 16-dimensional JL-Rademacher projection by default.

Table 14: **Ablation of selector feature reduction on LLaVA-1.5-7B with** 128 **retained visual tokens.** Reduced representations use $d_r = 16$, while Full dimension performs selection in the original feature space. All other settings remain unchanged. † marks the default configuration, and the best result in each column is bolded.

Feature reduction	GQA	MMB	MME	POPE
Random coordinate	57.71	62.23	1743.87	84.48
PCA	59.53	63.06	1756.22	**86.52**
Top variance	59.02	63.23	1737.61	86.23
Max pooling	59.10	62.89	1745.75	86.08
JL-Rademacher†	59.33	63.14	**1764.96**	86.37
Full dimension	**59.56**	**63.40**	1753.46	86.24

D Limitations and Future Directions

Adaptive token budgeting. Performance is locally non-monotonic across token budgets on several benchmarks: retaining more visual tokens does not always improve and can occasionally reduce task performance. LEAF nevertheless operates under a prespecified budget shared across inputs. An input- and question-adaptive budgeting mechanism could instead estimate how much visual evidence each sample requires, potentially improving both accuracy and efficiency.

In-model token selection. Truncated question-aware probing requires a partial forward pass over the full visual-token sequence, increasing theoretical computation even though parallel execution limits its wall-clock impact. Future work could instead integrate selection directly into the language model. After the probe layers, unselected visual states could be discarded and the retained states processed by the remaining layers. Reusing the probe computation in this manner would avoid the separate complete forward pass currently applied to the compressed multimodal sequence.